CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Classifying Short Clinical Notes:

An Unsupervised Approach

Master's thesis in Computer Science – algorithms, languages and logic

KEVIN CHEN TRIEU & LONG NGUYEN

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Master's thesis 2020:NN

Classifying Short Clinical Notes:

An Unsupervised Approach

KEVIN CHEN TRIEU & LONG NGUYEN

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2020

Classifying Short Clinical Notes:
An Unsupervised Approach
KEVIN CHEN TRIEU & LONG NGUYEN

Supervisor : Jacobo Rouces Gonzalez, Språkbanken, University of Gothenburg
Advisor : Thony Price, Appva AB
Examiner: Carl-Johan Seger , Department of Computer Science and Engineering

Master's Thesis 2020:NN
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2020

Classifying Short Clinical Notes:
An Unsupervised Approach
Kevin Chen Trieu & Long Nguyen
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

A mandatory task in Sweden is the reporting of clinical procedures with a specially assigned code based on the procedure. It is both time-consuming and troublesome for medical personnel since more than 10,000 codes exist. By automating this task, it is possible to both save time of the personnel and money within the healthcare industry.

This master thesis explores an alternative way of classifying short clinical notes through unsupervised methods when quality labelled data is not available. By combining advances within NLP, utilising word embeddings and incorporating additional knowledge into the data, a classifier which do not rely on labelled data is presented.

Instead of learning by examples as supervised methods, the classifier manages to find semantic similarities between clinical notes and the description of the different codes, making it intuitively similar to how we humans would classify a code.

Keywords: Natural language processing, text classification, unsupervised learning, word embedding, short text, self-supervised, information-retrieval, clinical text

Acknowledgements

First and foremost, we would like to thank our supervisor, Jacobo Rouces Gonzalez, for his guidance and expertise which were highly valuable and appreciated. We would also like to thank the team at Appva AB for providing the necessary resources as well as support, especially our advisor Thony Price. Finally, we would also like to thank our examiner Carl-Johan Seger for his insightful feedback during the progress of the thesis.

Kevin Chen Trieu & Long Nguyen, Gothenburg, June 2020

Contents

List of Figures

List of Tables

1
Introduction

Since 2007, it is mandatory for licensed personnel such as doctors and nurses to report clinical procedures involving patients in the form of short notes and a corresponding care-procedure code (KVÅ-codes) to the Swedish National Board of Health [1]. An extension introduced in 2019 included all medical personnel and social workers in both private organisations and municipalities [2]. This task requires manual labour and is massively time-consuming due to the number of daily executed procedures and the large number of KVÅ-codes available. Since there are more than 10 000 different KVÅ-codes divided into nine categories and 123 sub-categories, it is next to impossible for the personnel to memorise [3]. The classification of KVÅ-codes has turned into an exciting and unique challenge on how to automatically assign a KVÅ-code given a short and unstructured clinical note in Swedish written by the personnel. This is in its essence a problem of text classification and information retrieval [4][5].

Text classification is a field within the Natural Language Processing (NLP) domain that has garnered attraction in recent years. The purpose of text classification is the process of categorising text into a set of predefined categories. Despite its complexity, recent advances through machine learning are producing results capable of beating humans without requiring a large amount of manual labour.

Today, state-of-the-art solutions within text classification are majorly represented by supervised methods [6][7] utilising advanced neural networks architectures. Despite their excellent results, supervised methods require a large amount of labelled data which is hard to acquire when in reality the norm consists of an abundance of unstructured text such as emails, chat conversations and social media. Thus alternative approaches are sought after.

Alternative approaches include semi-supervised and unsupervised methods where the goal is to combat the problem with scarcely labelled data [8][9]. Unfortunately, due to their often poor results compared to supervised methods, they are overlooked and less studied. However, recent advances show promise utilising semi-supervised approaches where data with few labelled samples can match supervised methods. Unsupervised methods, on the other hand, are still rarely used for direct text classification and are instead, mainly used for feature learning or clustering of unlabelled data. Recently published papers where unsupervised methods can beat baseline models shows promise for this approach but indicates that there is still much to be investigated.

1.1 Purpose

The purpose of this thesis is to investigate whether an unsupervised approach is sufficient for classification of very short unstructured text to a large number of predefined categories (108 categories). Since descriptions of the labels exists and the task are to find the most relevant KVÅ-code, it falls between the areas of traditional text classification and information retrieval systems. By leveraging recent success from unsupervised learning and ideas from information retrieval, a new classifier for text classification is proposed. Methods and approaches from related works are explored to propose a suitable classifier for the domain.

1.2 Limitations

In this work, the focus is circumscribed to the medical domain. More specifically, the classification of short clinical notes in Swedish to their corresponding care procedure code. Since the data retrieved from customers of Appva consist mainly of elderly care organisations and municipalities, the data is in its nature limited.

The data is limited to 108 actual unique care procedure codes in use. Therefore, the classifier will be limited to the classification of 108 different categories in its current state. The reason is that executed care procedures are often monotonous where the procedures are often similar. Thus, different areas within healthcare use a certain number of specific KVÅ-codes. Furthermore, since the data is limited to the customers of Appva, it will not be representative of the complete healthcare given in Sweden.

Lastly, the work will solely focus on unsupervised methods, although other options exist such as supervised systems which are proven to work well on text classification. This is due to the limitation of the data provided, which only has a small subset of labelled data. Thus, building a supervised system is not feasible.

1.3 Outline

Chapter 2, *Theory*, summarises the necessary theory behind the methods which are required to understand this thesis..
Chapter 3, *Methods*, describes how the work has been made from data analysis to the complete classifier.
Chapter 4, *Evaluation and Results*, presents the results of the classifier and the evaluation of it.
Chapter 5, *Discussion*, analysis of the methods, results of this thesis and work which should be done in the future.
Chapter 6, *Conclusion*, summarises the project.

2
Theory

This chapter introduces the reader to the theory behind the methods used in this paper. A short introduction to machine learning and artificial neural networks are given. After that, Natural Language Processing is introduced, followed by different methods to model text in vector form and how they can be measured. The chapter concludes by presenting related works where inspiration have been taken.

2.1 Machine Learning

Machine Learning is the science of automated learning for computer systems. In short, machine learning allows computers to learn and improve from experience. Machine learning has currently broad applications within image processing, speech recognition and text analysis. Various machine learning models exist, from support vector machines to traditional Bayesian networks with the prevalent one today being artificial neural networks. While Machine Learning has a common purpose as a whole, it is typically divided into two kinds, supervised and unsupervised learning [10].

2.1.1 Supervised Learning

Supervised learning is the learning of a system using data with known outcomes, also known as labelled data. Each example consists of an input x and a known outcome y. Together, the examples make up the data set of input/output pairs (x, y). By feeding the examples to a machine learning model, it learns a mapping $f : X \to Y$ where f is dependent on a set of parameters, w [10]. Supervised learning has shown considerable success with state-of-the-art results within classification tasks, such as image recognition and sentiment analysis. However, it is held back by the requirement of a large amount of labelled data.

2.1.2 Unsupervised Learning

Unsupervised learning is the learning of a system by using data where the outcomes are not known. Compared to supervised methods which predict y given x, unsupervised systems aim to learn the underlying features of the data consisting of $x \in X$ without the knowledge of $y \in Y$. This makes unsupervised methods a practical approach compared to supervised methods which require labelled data. However, unsupervised methods are often great for tasks such as clustering and alternative

representations of data while it lacks in tasks where supervised methods excel.

Self-supervised learning is a method considered a sub-area within unsupervised learning that is gaining traction. Compared to the traditional unsupervised method, self-supervised learning is similar to supervised in the sense that there are known outcomes. The difference is that the outcomes are directly derived from the data itself. This means that it can benefit from the advantages of supervised learning while avoiding the obstacles of labelled data.

2.1.3 Semi-supervised Learning

A hybrid solution which tries to minimise the weakness of both supervised and unsupervised methods is semi-supervised learning. Although it still relies on labelled data, the requirements are lower where the majority of the data is unlabelled with a few amount that is labelled. This gives considerable improvements compared to unsupervised learning while matching the performance of supervised methods in some cases.

2.2 Artificial Neural Networks

An artificial neural network (ANN) is a computational model loosely inspired by the biological neural networks that make up the human brain. It consists of a group of interconnected neurons, where each neuron serves a purpose which is to calculate the output based on its internal state and the received input. By interconnecting the neurons, they form a directed graph known as a neural network that is capable of learning complex tasks which can solve different classification and regression problems [11].

The perceptron introduced in 1958 by Rosenblatt is widely considered the first modern neural network [12]. It serves the purpose of a binary classifier which given multiple inputs produces a single binary output. Weights were introduced, which expressed the importance of each respective input and a bias b. The output of the neuron, a binary number was calculated by the weighted sum of the input values with their respective weights. The output was dependent on the threshold which returned 1 when the weighted sum was larger than the threshold and otherwise 0.

From perceptrons, new neurons were born. Instead of being limited to binary inputs and outputs based on the step function, modern neurons utilise a wide variety of different functions known as activation functions. By interconnecting the neurons together, complex networks are built making them capable of solving a wide variety of problems.

A neural network consists of an *input layer* with an arbitrary number of inputs X_n and an *output layer* with an arbitrary number of neurons with outputs Y_n. Multi-layer networks are constructed by introducing an arbitrary amount of hidden layers with an arbitrary number of neurons. As noticed, neural networks can be

constructed in arbitrary size and form adjusted for each specific task. In Figure 2.1, a neural network with an arbitrary number of inputs, one hidden layer and two outputs is shown.

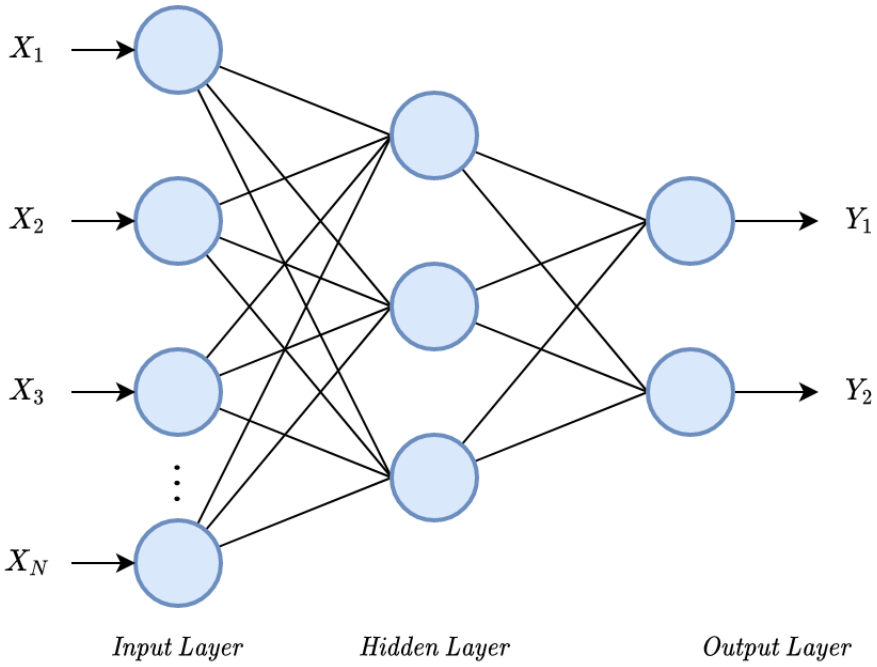

Figure 2.1: A neural network with one hidden layer.

2.2.1 Training of a Neural Network

The weights and biases of a neural network when constructed, is often initialised randomly. A newly constructed ANN with randomly initialised weights cannot make sensible predictions. Instead, the ANN learns from the data that is fed to it, which are divided into batches of size b. For every batch, the examples are fed through the ANN, which is known as the forward propagation. After that, the error is calculated, and the weights are fine-tuned through the backward propagation.

Forward propagation refers to the process of propagating the input values forward in the neural network until its resulting output in the output layer. The calculations are identical to the perceptrons, except that the outputs are propagated into the neurons in the next layer until it reaches the output layer. The weighted sum is calculated as seen in equation 2.1 which is then fed into the activation function in 2.2 resulting in the output y.

$$z = \sum_{i=1}^{n} X_i W_i \tag{2.1}$$

$$y = f(z) \tag{2.2}$$

The resulting output from the neural network is known as the *prediction*. The Loss function is the function used to evaluate how close the prediction is to the desired

outcome. A high loss score indicates that the prediction is far from the expected outcome, while a low score indicates the opposite. Stochastic gradient descent is used to minimise the loss function. The algorithm used to calculate the gradients is backward propagation. It calculates the gradient of the loss function with respect to the weights of the neural network.

2.2.2 Optimising an Artificial Neural Network

The training of a neural network needs to be optimised for it to reach its optimal performance. In order to do this, it is important that over and underfitting of the data is avoided. By overfitting, it means that the model has learnt the given training data too well, where it negatively impacts performance on new unseen data. Underfitting, on the other hand, refers to when the model has not learnt enough from the data. Thus it is not able to predict correctly and will generalise terribly to new data. In Figure 2.2 the over and underfitting is visualised.

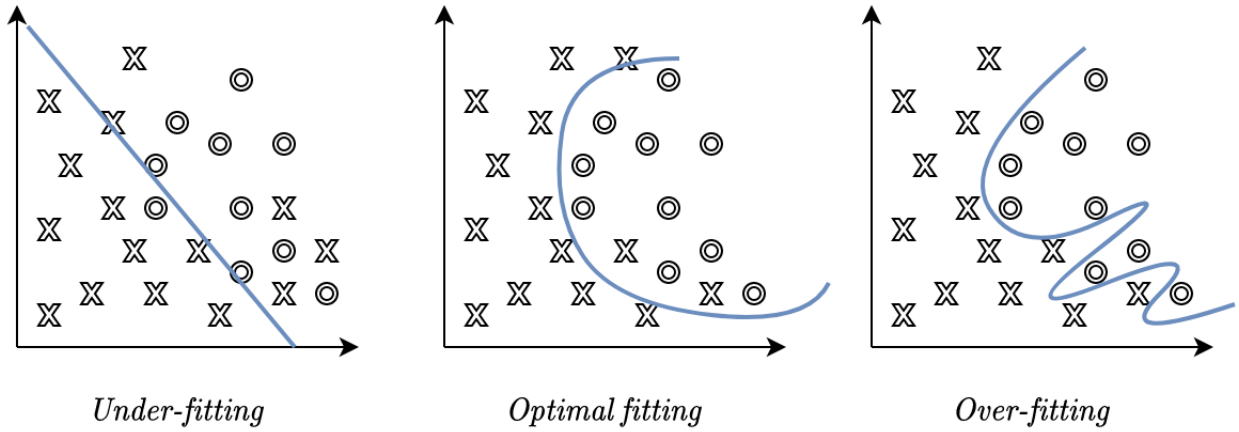

Figure 2.2: The difference between an underfitted, optimal and overfitted model in a binary classification problem.

An artificial neural network contains many parameters which can be tuned in order to improve the performance and get an optimal fitted network. From adjusting the architecture by the number of neurons and layers to tuning the hyperparameters.

Typical hyperparameters include *learning rate* which dictates how fast the network learns, the *epochs* which is the number of iteration the whole training data is fed to the network for training and the *batch size* which is the number of samples given to the network before the weights in the network is tuned though backward propagation.

2.3 Natural Language Processing

Natural language processing (NLP) is the area within linguistics and computer science that focuses on how computers can learn to analyse, understand and generate

human language. NLP has led to the development of many useful applications, including virtual assistants and chatbots.

Human language is complex, making NLP one of the most challenging research areas. Abstract notions within semantics and pragmatics, such as polysemy and sarcasm, are difficult for a computer to understand. In fact even humans can have a difficult time understanding sarcasm.

Research within NLP focuses on utilising rule-based algorithms to convert human language into a suitable format which computers can understand. Traditionally, the focus has been on syntax [13][14], where identifying the structure of texts and building collection of them were the main focus. Today, many excellent algorithms exist to solve these problems making it a trivial task for the computer.

Instead, the success of machine learning has made a shift within NLP towards automated learning. Large amounts of data are used to learn complex structures which were previously made manually. It has propelled a new way of understanding languages where semantics between sentences are captured, often with the help of neural networks without any additional domain knowledge than the data itself. This has opened up lots of opportunities, leading to an increased interest in word representation such as bag-of-words and word embeddings.

2.3.1 Document Classification

The availability of text data is rapidly growing [15], and text is often organised in documents of some sort. Document classification is the problem of assigning a document such as text to one or more pre-defined categories. Traditionally this has been done manually by humans, but it is highly time-consuming, so automatic classification has become a highly relevant task.

2.3.2 Information Retrieval

Information retrieval (IR) is the task of obtaining information resources relevant to a specific query from a collection of resources, often a database or storage [5]. The query may consist of a text string or a set of keywords. Typical applications of IR are search engines and databases. In general, IR works by finding similarities or relationships between the query and the documents, and a simple method would be to see how many terms are similar in the query and the corpus of documents.

2.4 Bag of Words

Bag-of-words (BOW) is a simplified representation of a text document in vector form. Using, BOW a text is modelled as a vector of length v, where v is the vocabulary size of the corpus. Every element in the vector represents a specific word in the vocabulary, and the value of this element determines if the word exists in the text or not. BOW is commonly used in classification tasks due to its simplicity and

good results but has its drawbacks.

Term frequency-inverse document frequency (TF-IDF) is a statistical method for calculating the importance of a word in a document or corpus. The weight is calculated by taking both the frequency of a word in the document (term frequency) and the inverse of the number of documents that a particular word appears. This approach results in reduced weight for common words and highlights the less frequent words in the corpus. The essential idea behind this approach is that the less frequent words usually have a crucial roll in determining the difference in semantic between documents.

In BOW, the simplified representation results in loss of information. Sentence structure is lost since word order is not taken into account. The vector length, which is directly proportional to the vocabulary results in a high dimensional vector.

2.5 Word Embeddings

Word embeddings is a self-supervised method for learning representations of words. Words are mapped from their text form to a numerical representation in the form of a vector of fixed length (embedded vector). Words that have similar semantic meaning will have similar embedded vectors and therefore be closely located in the vector space.

Word embedding models commonly used today are Word2Vec and Fasttext. Both are unsupervised methods using self-supervision to learn the semantic meaning of words using the corpus of texts. Compared to Word2Vec, Fasttext is capable of handling both infrequent and out-of-vocabulary words by dividing words into subwords (n-grams).

The neural network architecture behind both Word2Vec and Fasttext are based on Continuous Bag-of-Words (CBOW) and Continuous Skip-gram (skip-gram) [16]. The essential idea behind both of the architectures is that a word is characterised by its surrounding words.

2.5.1 Continuous Bag of Words and Skip-gram

In both ANN architectures, CBOW and skip-gram, labelled data is generated by separating a word and its surrounding words. The CBOW architecture takes a word as the label data and learns to predict this word using its surrounding words as the input. The illustrations of CBOW architecture can be seen in Figure 2.3.

This leads to a potential weakness of representing less frequent words using the CBOW architecture and can be tackled by switching to skip-gram architecture. The opposite approach, skip-gram architecture learns to predict the surrounding words using the word itself as the input. The illustrations of skip-gram architecture

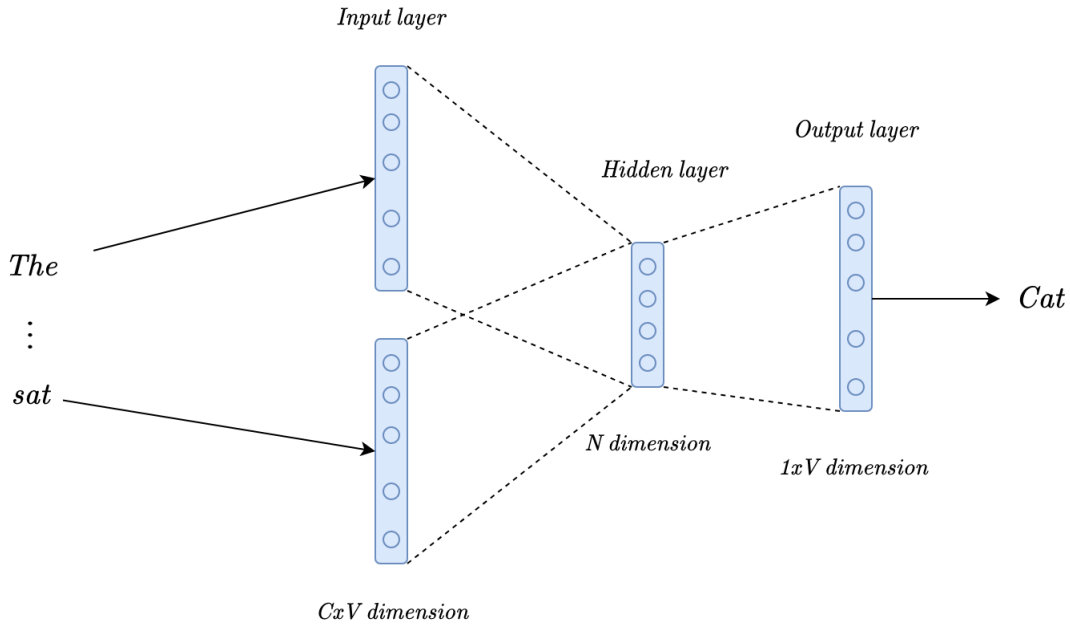

Figure 2.3: CBOW takes surrounding words as input and outputs a word

can be seen in Figure 2.4.

In both figures, C is the number of surrounding words taken into consideration, V is the size of the BOW vector. N is the dimension size of the word embedding. The input and output following an example corpus below:

- The dog sat in house.
- The cat sat on a bench.
- The cat sat behind the wall.

As mentioned, the CBOW architecture learns to predict the word by the surrounding words. In other words, it is maximising the probability of predicting the most frequent word given the surrounding words. This becomes a potential weakness as the rare words will get much less attention by the model as it is built to predict the most frequent words having the same surrounding words. Take the surrounding words "The ... sat" from the above example corpus, CBOW architecture learns to predict the word "cat", and the word "dog" gets less attention since it is less frequent. In the skip-gram model, the word "dog" is treated equally as the word "cat" since the pair "The ... sat" and "dog" will be considered as a new piece of information. Therefore, the skip-gram also generates more pairs of input and output to train on than the CBOW.

2.5.2 Word2vec

Word2Vec, published by Google in 2013, is a feed-forward neural network with two layers for word embedding [17]. Word2Vec is capable of utilising both CBOW and skip-gram architectures to produce an embedded vector representation of a word. In Word2vec, the first step is to prune the vocabulary by ignoring the words that appear

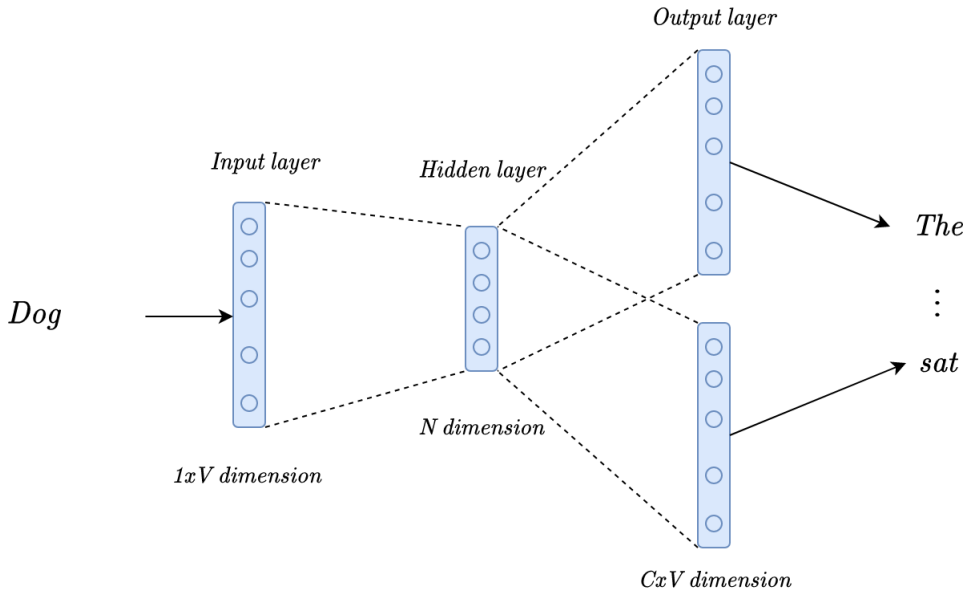

Figure 2.4: Skip-gram takes a word as input and outputs its surrounding words

less than *min_count* times (default value is 5) as they are considered to be typos or "noises". Furthermore, as they appear too infrequently, there are not enough data to train on. Therefore, the biggest drawback of Word2vec is that out-of-vocabulary and rare words cannot be represented as an embedded vector.

2.5.3 Fasttext

Fasttext was created by Facebook's AI Research lab as an extension to Word2Vec [18]. Instead of maintaining complete words as input, Fasttext divides a word into several sub-words of length n (n-gram). For example, the 4-gram of word *league* is '$<$ *lea*', '*leag*', '*eagu*', '*ague*' and '*gue* $>$' while the 4-gram of word '*hi*' is only '$<$ *hi* $>$'. The symbols '$<$' and '$>$' are special symbols which represent the start and end of a word. The embedding vector of a word is the sum of the vectors of all the n-grams. In Fasttext, the parameters *min_n* and *max_n* determine the maximum and minimum size of an n-gram. For rare and out-of-vocabulary words where the trained Fasttext embedding has embedded vectors of all its n-grams, Fasttext generate their embedded vectors through the same approach. However, for words that the trained Fasttext embedding does not have an embedded vector of each n-gram, they are divided into smaller n-grams of a minimum length of $\geq min_n$. However, a smaller n-gram leads to a less accurate representation as there are more words in the vocabulary sharing this n-gram. Thus its embedded vector is more generalised. Through this approach, the trained Word Embedding is, therefore, capable of handling out-of-vocabulary and rare words.

2.5.4 Creation of Sentence Embedding

Many tasks deal with sentences rather than single words. Being able to represent sentences in the shape of vectors are, therefore, of importance. Word vectors which are great for capturing semantic similarity are extended to represent sentences through various methods such as Doc2Vec [19] which is an extension of Word2vec.

2.5.5 SIF-embedding

Another approach for creating sentence embedding which has shown excellent results and is commonly used is the averaging of words vectors. Smooth Inverse Frequency Embedding (SIF) [20] takes this a step further. Instead of a simple average of the word vectors, a weighted average with respect to the term frequency is used, as shown in equation 2.3. This method has shown to beat sophisticated supervised methods.

$$\frac{a}{a + p(w)} \tag{2.3}$$

where: a: Hyperparameter, usually set to 0.0001
$p(w)$: The term frequency of a word w

2.6 Measuring Text Similarity

Text similarity is the measurement of determining how similar two pieces of text are. It is critical for the performance of several applications from information retrieval systems to text classification. The similarity is typically measured in two ways, either by semantic or lexical relevance. Semantic similarity is the similarity between words, sentences, or documents where they are conceptually similar, while lexicographical similarity refers to how similar they are in their form [21].

2.6.1 Levenshtein Distance

Levenshtein distance, also known as edit distance, is a measurement of the difference between two sequences of text. The distance is calculated as the minimum number of basic operations required to transform one sequence to the other. The available basic operations are character insertion, deletion and substitution. For example, the distance between the two words *three* and *tree* is one since the deletion of *h* from the first sequence would make them identical.

2.6.2 Cosine Similarity

Cosine similarity is the measurement of similarity between two non-zero vectors, namely the cosine of the angle between the two vectors. The cosine similarity score varies between minus one and one where two identical vectors would have the score

of one. The advantage of cosine similarity is due to its low complexity and can handle sparse-vectors well due to the non zero dimensions being ignored.

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{xy}}{\|\mathbf{x}\|\|\mathbf{y}\|} = \frac{\sum_{i=1}^{n} \mathbf{x}_i \mathbf{y}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{x}_i)^2}\sqrt{\sum_{i=1}^{n} (\mathbf{y}_i)^2}} \tag{2.4}$$

The formula of cosine similarity is shown in equation 2.4. x and y are the two input vectors. The numerator is the dot product of the two vectors, and the denominator is the product of the magnitude of each vector.

2.6.3 Word Mover Distance

With the rise of word embedding, traditional similarity measures that had been effective and efficient seemed too simple to capture the information in embedded word vectors. Word mover distance [22] proposes a measurement which suggests that the distance between words in the vector space are semantically meaningful. The distance between two documents is calculated by the minimum cumulative distance required for document A to match the text of document B.

2.7 Inter-Annotator Agreement Scores

Since linguistically related problems are partly determined by human judgement, the correctness of the labelled data cannot be measured directly. On the other hand, validity and reliability can be measured by comparing decisions made by different annotators on the same data set. This process is known as the Inter-Annotator Agreement [23]. Through the Inter-Annotator Agreement, the reliability of decisions made by an annotator is determined by similarity compared to decisions made by other annotators.

As one of the existing reliability measures, Cohen's Kappa score measures the similarity (agreement) between decisions made by two annotators [24]. However, Cohen's Kappa score also takes the probability of the occurrence of an agreement. Therefore, a category with more entries classified to it is equally important as a category with a lesser number of entries classified to it. This is important as the problem presented in this paper is a multi-label and classification problem with a non-uniform distributed data set.

Based on various papers [25] [26], Cohen's Kappa score can be interpreted differently based on the specific task. However, most of the papers agree on a threshold of 0.8 for an annotation to be considered reliable.

2.8 Related Work

This section presents different works with the same aim of solving text classification and information retrieval problems. Other methods which improve upon the presented solutions are also presented.

2.8.1 Classification of Text

Text classification is a widely researched domain with many practical applications, as mentioned previously. Many of these applications use neural networks, which usually outperform most other approaches today. Readily available plug-and-play services exist where classifiers can be built within minutes when labelled data are provided.

More domain-specific and advanced classifiers are also built. Hughes et al. [27] propose a supervised classifier for medical text classification. By using Word2vec to transform the text into a word vector which was then fed into a neural network, an accuracy of 68% has been achieved, beating many prior biomedical classifiers.

On the other spectrum of machine learning, a paper published in 2019 [9], proposes an unsupervised method for text classification on documents written in English. Instead of relying on labelled data, they decided to enrich the label of every category with the help of humans, dictionaries, and word embeddings by including relevant words. An example is that documents about sports should be assigned to the category *Sports* which has been enriched by including the words *Football* and *Basketball*. After that, they matched the documents to the enriched categories by comparing their similarities with Latent Semantic Analysis. This managed to achieve an F1-Score of 55.7 on the Yahoo-Answers data set, which consists of ten different categories making it on par with simple supervised approaches.

2.8.2 Alternative Word Representations

An autoencoder is an ANN architecture that uses a self-supervised approach, which can be divided into two parts, *encoder* and *decoder*. The text input is first transformed into a vector representation using an approach such as BOW. After that, the vector representation is encoded by the encoder to a denser representation by reducing the number of dimensions. The decoder is used to reconstruct the corresponding vector representation from the encoded representation. The encoder becomes better at extracting essential features and the decoder at expanding these features to its original input through extensive training on data. After the training phase, the encoder can be used to create a low dimensional vector representation of a text where only essential features are kept.

The latent space vector can be retrieved from the encoded representation, as shown in 2.5. This vector can be used as a word representation in the same way word embedding is used. Advanced autoencoders have shown to perform on par with word embedding, albeit being more complex [28].

The rise of popularity in word embedding has led to a trend of pre-trained word vectors. Some of this is due to the large amount of data required to build useful, versatile word embeddings. In 2018 google published a universal sentence encoder [29] to make universal embeddings that could be used for many downstream tasks. The initial support for English was quickly extended to 20 different languages due

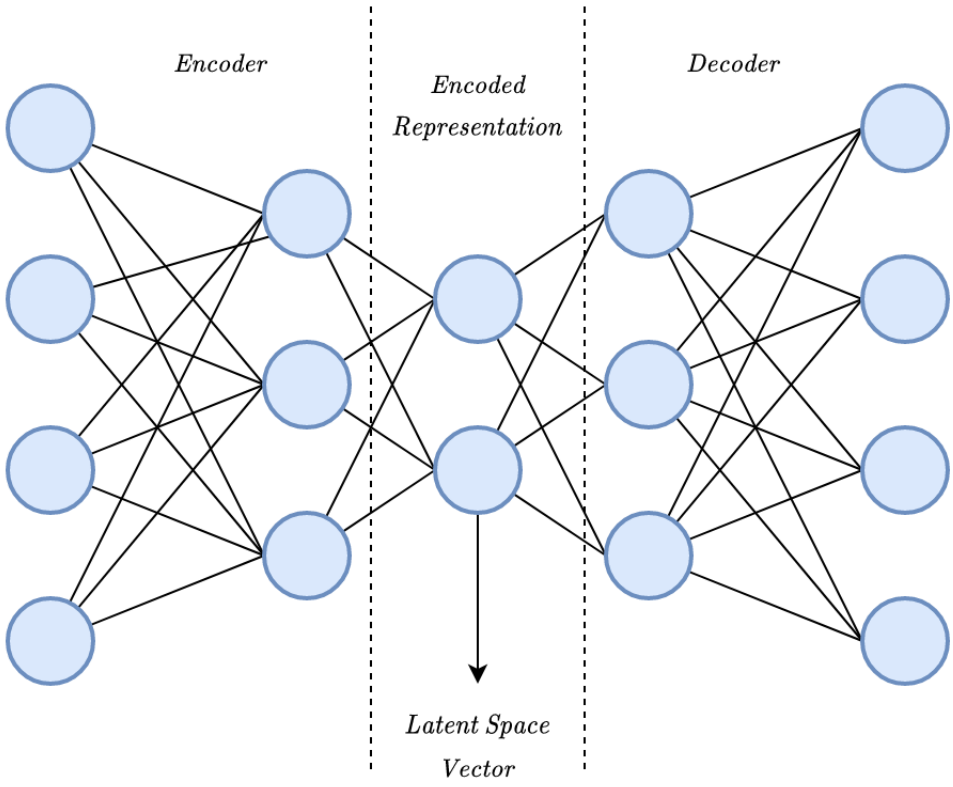

Figure 2.5: The architecture of an Autoencoder with three hidden layers.

to the popularity.

2.8.3 Sentence Similarity Based on Semantic Nets

In 2006, Li [30] observed the increasingly important role of sentence similarity in a wide variety of applications from text mining to information retrieval systems. Traditionally these systems are considered inefficient and require human input while being terrible at generalising. Thus Li proposed a model which is capable of measuring the similarity between short texts based on semantic and word order. The semantic of the text is obtained by combining lexical knowledge about each word from lexical databases. At the same time, the model proposed by Li measures the similarity in word order by the number of different words as well as the number of word pairs in different orders.

2.8.4 Meta-Embedding

Every word embedding trained with its method and resources yields a unique word representation varying in quality and characteristics. Yin and Schutze proposed to combine different pre-trained embeddings with the aim of learning meta-embeddings. The idea behind this is that by combining the embeddings, the unique Meta-embedding would make for a richer representation with the strengths of each distinct

embedding. By employing a linear neural network, which learns to project the Meta-embedding of a word into the source embeddings using their respective projection matrices [31].

An unsupervised approach is proposed and proven to be outperforming other approaches such as concatenation or single-value-decomposition based Meta-Embedding in short-text classification[32]. The proposed model consists of two parts, reconstruction and projection. First, for each word in the vocabulary, its embedded vector is reconstructed using the embedded vectors of the most similar words from every Word Embeddings. Thereafter, the Meta-Embedding is generated through projecting the reconstructed vectors to the vector space.

In 2018, Coates and Bollegala introduced a much simpler model that was comparable or better than more complex meta-embedding methods. Instead of relying on neural networks, they used simple arithmetic by averaging word embeddings together [33].

2.8.5 Dimension Reduction of Word Embeddings

Word embedding trained on large corpora of texts is widely used in many applications within natural language processing. Often as a way to format input ready to be used in downstream tasks such as text classification or semantic similarity. Due to the large dimensions of word embeddings, they are prone to memory constraints. Therefore, multiple post-processing approaches have been proposed to reduce the dimensions, effectively increasing the efficiency of the embedding.

Mu and Viswanath [34] observed that the word vectors share a common mean vector, which strongly influences the word vectors. By removing the non-zero mean vector from the word vectors, they managed to obtain word vectors, which better captured the meaning of words. This effectively reduces the dimension and results shows that they managed to increase on average 1.7% on word similarity tasks.

3
Method

This chapter details step by step how the classifier has been built. First, useful information is extracted, and decisions are made based on data analysis. After that, every step in building and evaluating the classifier is explained. Lastly, the architecture of the classifier is shown.

3.1 Data Analysis

Appva provided the data used to train the word embedding which was used for the classifier. The data set consists of clinical reports handwritten by licensed medical personnel and was collected from both private organisations and municipalities within the Swedish healthcare through Appva's medication and care support system.

Analysis of the data was performed to discover useful features while also obtaining better knowledge and understanding of the domain. This helped in the process of designing the classifier. In summary, the data set contains 620,028 entries with 108 unique KVÅ-codes. The average length of an entry is 11 words. The format of the data entry is given below.

- **List** : Category defined by an organisation (hospital or elderly home).
- **Label** : Short description of the clinical note written by personnel.
- **Description** : The clinical note.
- **KVÅ-Code** : KVÅ-code assigned by their current solution.
- **Human-assigned** : Whether the code has been manually corrected by the user.

Except for the *Human-assigned* field, all the fields are in the form of text. The *Human-assigned* field has a value of 1 if the entry has been double-checked and assigned a new KVÅ-code by licensed personnel and 0 otherwise. An example of data entry is given below, which have been translated to English.

- **List** : Narcotics.
- **Label** : Morphine patch.
- **Description** : Rotate clockwise when changing.
- **KVÅ-Code** : DT017.
- **Human-assigned** : 1.

3.1.1 The Validation Set

The validation set was used to evaluate the different parts which constitute the classifier. The results from the evaluation were used to tune parameters of the word embedding, pre-filter and enriching. The validation set contains 4872 entries and covers 29 KVÅ-codes out of 108 total. This data set has previously been manually reviewed by Appva but may still contain errors since the process has been by personnel without expert knowledge within the medical domain. However, the personnel has received assistance for classification of entries in the form of keywords to KVÅ-code suggested by licensed medical personnel. The figure 3.1 shows the distribution of KVÅ-codes in the validation set.

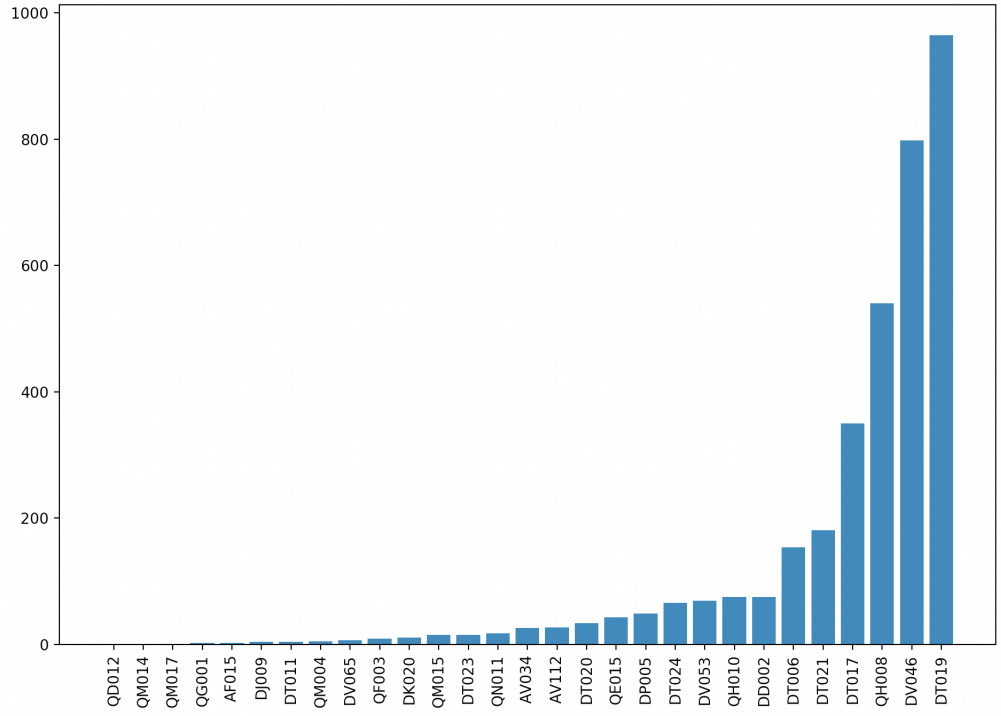

Figure 3.1: The distribution of all 29 KVÅ-codes in validation set.

3.1.2 The Test Set

The test set was used to perform the final evaluation of the classifier. It consists of 2521 labelled data entries, which cover 102 out of 108 total KVÅ-codes in use. The distribution of the codes in the test set is skewed. Many codes are occurring once while code such as DT019 is covering 25% of the entries.

Each entry in the test set has been manually classified by licensed personnel. The entries were first wrongly assigned by the current classifier at Appva and after that changed by licensed personnel. The test set may, therefore, not represent the actual distribution of codes, as there may be a bias towards misclassified KVÅ-codes. The test data shows an approximate Pareto distribution where 83.18% of the entries are assigned to 20% of the KVÅ-codes in use as can be seen in Figure 3.2.

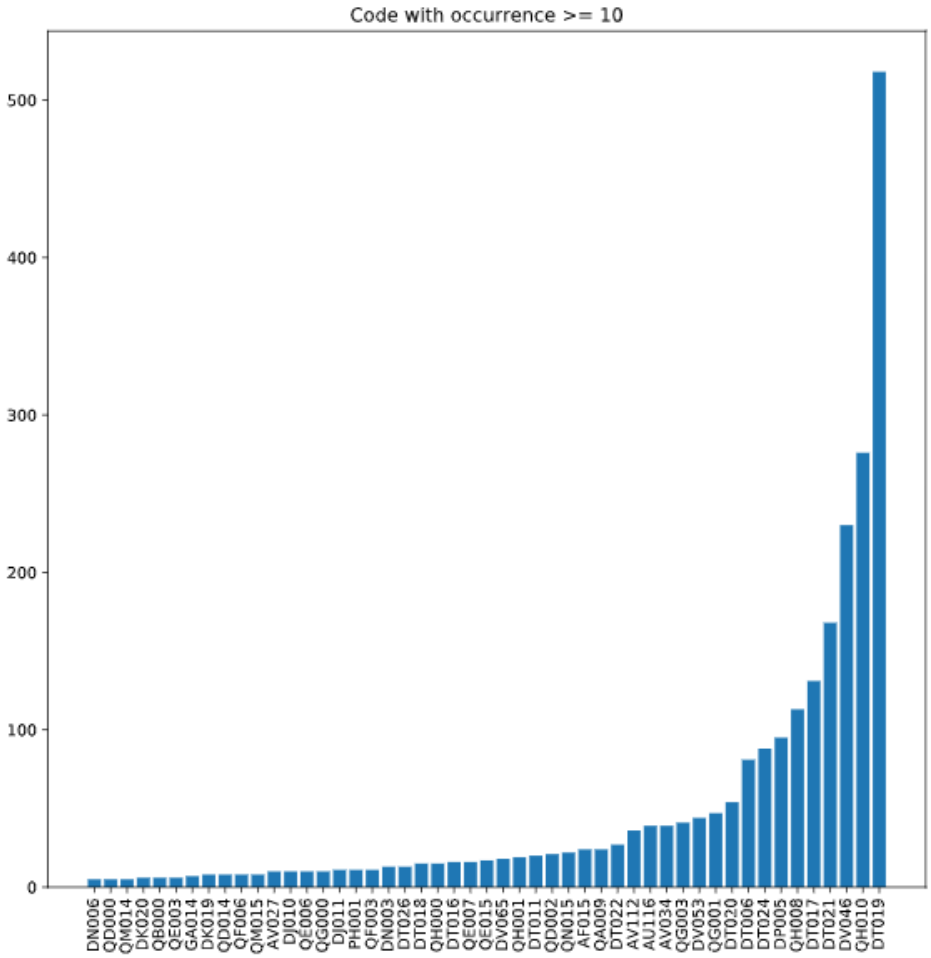

Figure 3.2: The distribution of the 49 most common codes in the test set

3.1.3 KVÅ Data Set

From Socialstyrelsen, a complete set of KVÅ-codes was obtained. In total, the data set contains 10553 unique KVÅ-codes. Each data entry consists of a KVÅ-code and a description of the code written in Swedish. However, there are only 108 KVÅ-codes in use based on the KVÅ-codes in the test and validation set. An example of a data entry translated into English is shown below:

- Code: DT019
- Description: Medicine prescription oral.
- Additional information: Medicine can be indicated by the ATC-code.

3.1.4 Self-labelling of Data

Due to the skewed distribution of labelled data where many KVÅ-codes occur rarely, self-labelling has been performed to try and alleviate the issues. The inter-annotator agreement is measured to evaluate the correctness of the newly labelled data as presented in 2.7.

First, a subset of data with known labels is manually annotated by new annotators, and Cohen's Kappa Score is calculated to measure the reliability of the manual annotation. Usually, the subset is sampled using a random sampling technique. However, due to the KVÅ-code's skewed distribution, a simple random sampling technique might result in a subset consists of only most frequent KVÅ-codes while least frequent KVÅ-codes are ignored. Thus, the stratified sampling technique is used. The stratified sampling technique is capable of providing a similar or better efficient statistical estimate [35]. The result is a subset where each KVÅ-code has at least one entry and a maximum of 5 entries. The subset is then manually annotated separately by two annotators. The score is calculated by crosschecking the labelling of each annotator against the known labels.

3.2 Text Pre-processing

Text pre-processing is an essential step for tasks within the classification of text. The purpose is to convert the data into a generalised format that is comprehensible by the machine while removing redundancies. The data was pre-processed in the following order:

1. Removal of redundant terms
2. Removal of stop words
3. Convert abbreviations to full form
4. Tokenisation of words
5. Lemmatisation of words

First, the redundant terms such as punctuation and additional white spaces were removed. Then, stop words ,which are words that do not contain meaningful information such as articles and pronouns, are removed [36]. After that, by utilising dictionaries with standard Swedish abbreviation and acronyms taken from Wikipedia, abbreviations and acronyms were transformed to their full form. The words were then tokenised and transformed to their base form through lemmatisation.

For lemmatisation, two different tools were implemented and tested separately; Sparven and Svensktext [37][38]. Sparven is an annotation tool provided by Gothenburg University's language bank and is capable of performing advanced analysis for both short text and corpus. For each word in a text, Sparven provides information such as the word's base form (lemmatisation), part-of-speech, dependency to other words and base words if the input word is a compound. Svensktext is a simple mapping tool using a static dictionary to perform lemmatisation. Svensktext is also capable of recognising some named entities that exist in the static dictionary.

An Example of pre-processing:
Input: "Two pills are given to the patient against pain"
Result: [pill, give, patient, pain"]

3.3 Training of Word Embeddings

Word embedding based on Word2vec and Fasttext were evaluated to determine the best suitable one. Different models by trying variations of vector sizes and the number of epochs were tried to train the word embedding.

Two sources of information were used to train the word embedding, the data set provided by Appva and the KVÅ codes with their corresponding description retrieved from Socialstyrelsen. Before training, the data entries corresponding to the test and validation set were removed in order to prevent the risk of bias since they are used to evaluate the classifier.

A pre-trained Fasttext model using Wikipedia data in Swedish was also evaluated and re-trained with the data set using incremental training. This model had been trained in Swedish, using CBOW with dimension size of 300, n-grams of length 5 and window size of 5 (number of surround words taken into training for a word) [39]. Incremental training is done using the same data set with the number of epochs of 5.

3.4 Pre-filtering

Some care procedures are more straightforward to classify than others. More specifically, the ones related to medicine intake (route of administration) are among the easiest since its code is directly related to its intake method.

The pre-filter was implemented using a medical list containing medicine names and other information such as dosage and form, which was retrieved from Läkemedelsverket [40]. Furthermore, 1177, a health service provided by the Swedish government, has a guide on how to determine the route of administration of medicine based on its form [41]. This guide was used to help create a relation between a medicine, its route of administration and the corresponding KVÅ-code. This relation is illustrated in Figure 3.3.

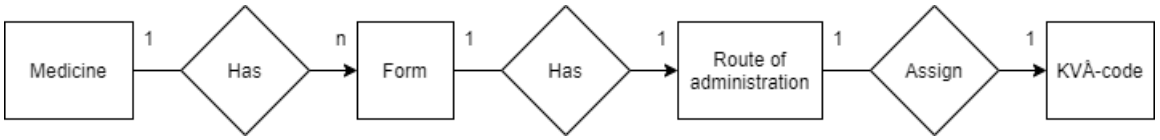

Figure 3.3: Entity relation diagram of a medicine and a KVÅ-code

A medicine might have different forms and therefore, different routes of administration. However, this can be narrowed down with additional information such as the dosage of the medicine. A medicine usually exists in several dosages and medicine in a specific dosage has a specific form. Furthermore, since a form only leads to a route of administration and there is only a corresponding KVÅ-code for each route of administration, it is significantly easier to classify these entries than other entries.

Therefore, the implementation of a pre-filter is used to reduce error and improve efficiency.

However, due to limited knowledge within the medical field, all medicine intake in the form of injection will be treated as subcutaneous (under skin) except for infusion which is intraperitoneal (injection into the peritoneum). For entries related to medical injection, the pre-filter will assign a KVÅ-code according to this limitation.

If the input contains a medicine name in the list, the pre-filter analyses the input and suggests a suitable KVÅ-code. Due to the occurrence of misspellings or abbreviations, by utilising Levenshtein distance and a spelling ratio, the pre-filter is also capable of handling misspellings of medicine names.

$$SpellingRatio = \frac{NumberOfMatchingCharacters}{NumberOfTotalCharacters} \tag{3.1}$$

3.5 Enriching the Data

The data entries contain eleven words on average (label and description), making it short compared to commonly used data sets where corpora of well-structured documents with long text are used, such as Reuters and OHSUMED. The data set was incorporated with external knowledge through enriching to overcome the shortness of the text. More specifically, the clinical notes were enriched with synonyms, definitions and medical information.

3.5.1 Enriching with BabelNet

BabelNet is a multilingual encyclopedic dictionary and a semantic-based network which covers hundreds of languages. Information such as grammatical categories, definitions and synonyms of a word can be retrieved from BabelNet. For each word, Babelnet defines its *senses* and *glosses*.

- Senses - a set of synonyms for a word
- Glosses - a set of definitions for a word

The knowledge retrieved from BabelNet were used to enrich the clinical notes and the KVÅ-descriptions in multiple ways. From enriching with the senses and glosses individually to combining them. After that, they were evaluated to see which option improved the overall text the best.

3.5.2 Enriching With Expertise

The information obtained from Läkemedelsverket was used to enrich the clinical notes [40]. Each entry that had a medicine name was identified and enriched with additional information of each medicine. Two approaches were tested. The first approach enriched the entries directly with the information obtained from Läkemedelsverket. The second approach utilised the relation between a medicine and a KVÅ-code created through a guide from 1177 [41]. Two examples are given below for each of the methods:

- Enriching with the form of the medicine obtained from the medicine list. E.g. "Alvedon" becomes "Alvedon Pill" after enriching.
- Enriching with the route of administration obtained through the guide provided by 1177. E.g. "Alvedon" becomes "Alvedon Oral."

3.6 Enhancing the Word Vectors

The word vectors obtained from a word embedding can be improved further through enhancing. In the proposed classifier, the word vectors were enhanced by meta-embedding and SIF-embedding to improve the representation further.

Meta-Embedding was implemented to combine the strengths and reduce the weakness of both Word2vec and Fasttext. Since Fasttext is based on character-level n-grams instead of complete words, there is a potential weakness in which semantically different words with a similar structure may be closely related. For example, the 4-gram '*eage*' exists in '*lineage*' and '*eager*', this 4-gram's embedded vector will be adjusted to fit both of these words. On the opposite, Word2vec does not divide the word into n-grams and compute the word vector for a whole word. The meta-embedding combined word vectors retrieved from Word2vec and Fasttext by averaging them into one single vector.

Word Embedding does not consider word importance. While the majority of stop words were removed through pre-processing, some words still contain more information than others. SIF-embedding was implemented to take the word importance into account.

3.7 KVÅ-code Classification

The classification of the data entries was based on the similarity measure. In some sense, this is similar to how we humans would handle the issue. First, comparing the clinical note to the definitions of the KVÅ-codes and after that, choosing the KVÅ-code with the best matching definition. Each clinical note was compared against the description of the 108 KVÅ-codes in use. The KVå-code with the highest similarity score was chosen as the correct answer.

Two similarity measurements were compared as they have their own advantages and disadvantages in different domains [42]. The implemented similarity measurements consist of:

- Cosine Similarity
- Word Mover's Distance

3.8 Evaluation Metrics

The task of classifying KVÅ-codes is a multi-category problem since it consists of more than two categories. Measuring the performance of a multi-category problem is not straightforward.

The classifier was evaluated using multiple measurements to give a fair picture of how well the classifier performs. Due to the skewed distribution and the high number of categories, basing the results purely on accuracy would not give a fair picture. Instead, the classifier was measured with accuracy, precision, recall and F1-score to give a fair picture of its performance.

In order to account for the category imbalance, the weighted average of the measurements Precision, Recall and F1-score were performed. Each measure was multiplied by the number of samples in their respective category and divided by the total number of samples. A side effect of this is that recall becomes identical to accuracy.

TP = True Positives
TN = True Negatives
FP = False Positives
FN = False Negatives

3.8.1 Accuracy

Accuracy is a commonly used measurement and presents the fraction of correct answers. Accuracy is commonly used but may be misleading, especially in imbalanced data. E.g., a set with 99 negatives and one positive sample. If all the samples were classified as negative, the accuracy would be 99% accuracy when, in reality, positive samples are classified wrongly.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

3.8.2 Precision

Precision for a category is the number of data entries correctly classified to this category divided by the number of data entries classified to this category.

$$precision = \frac{TP}{TP + FP} \tag{3.3}$$

3.8.3 Recall

Recall for a category is the number of data entries correctly classified to this category divided by the number of data entries belonging to this category.

$$recall = \frac{TP}{TP + FN} \tag{3.4}$$

3.8.4 F1 Score

F1-score is the weighted harmonic mean of precision and recall [43], so it takes into consideration both false positives and false negatives. It is often more useful in imbalanced data.

$$F1score = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{3.5}$$

3.9 A Baseline Classifier

In text classification tasks, a common method is to transform the text into a numerical vector to represent various linguistic properties. One of the most common approaches for textual features extraction is the Bag-of-Words (BOW) model. Firstly, the description of each KVÅ-code will be transformed into a BOW vector representation as well as the input. Thereafter, the BOW vector of the input will be compared to the BOW vector of each KVÅ-code using cosine similarity to find the most suitable KVÅ-code.

3.10 Proposed Classifier Architecture

Two significant iterations of the classifier were performed. The chosen implementation methods in each part of the classifier are presented to provide a proper overview of the classifier.

3.10.1 First Iteration

The first version of the classifier consists of five parts; *pre-processing*, *pre-filtering*, *word embedding*, *sentence embedding* and *comparison*. First, an input is pre-processed to a general form. Thereafter, the input is sent into the pre-filter, which classifies the simpler entries. In case the entry cannot be pre-filtered as it does not contain a medicine name, each word in the input is transformed into an embedded vector through the Word Embedding. Then a sentence vector is created by averaging the word vectors. In the final step, the sentence embedding is compared against all sentence embeddings of KVÅ-codes using cosine similarity. The most similar one, which has the highest cosine score, is chosen. An illustration of the first iteration is shown in Figure 3.4.

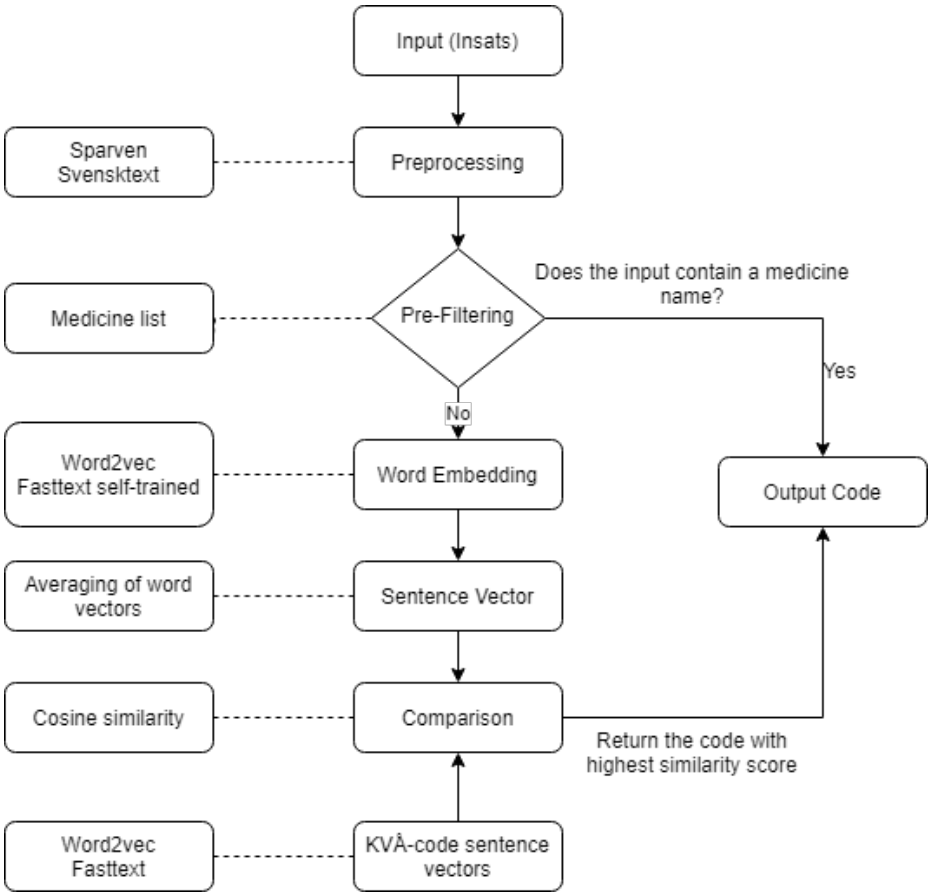

Figure 3.4: The first iteration of the classifier architecture including implemented methods on each part

3.10.2 Second iteration

Understanding the weakness of the pre-filter and having only a word embedding, the second version has been built to tackle this issue and explore other possible methods for each part.

In the second version, the pre-filter has been removed, and medical expert knowledge from the medicine list used by the pre-filter is transferred to the enricher. Each word in the enriched input text is then transformed into an embedded vector as in the first version. Here, a new part has been added to strengthen the embedded word vectors by utilising Meta-Embedding. Finally, the comparison is made as in the first version to find the most suitable KVÅ-code. The classifier's second iteration has an architecture as in Figure 3.5.

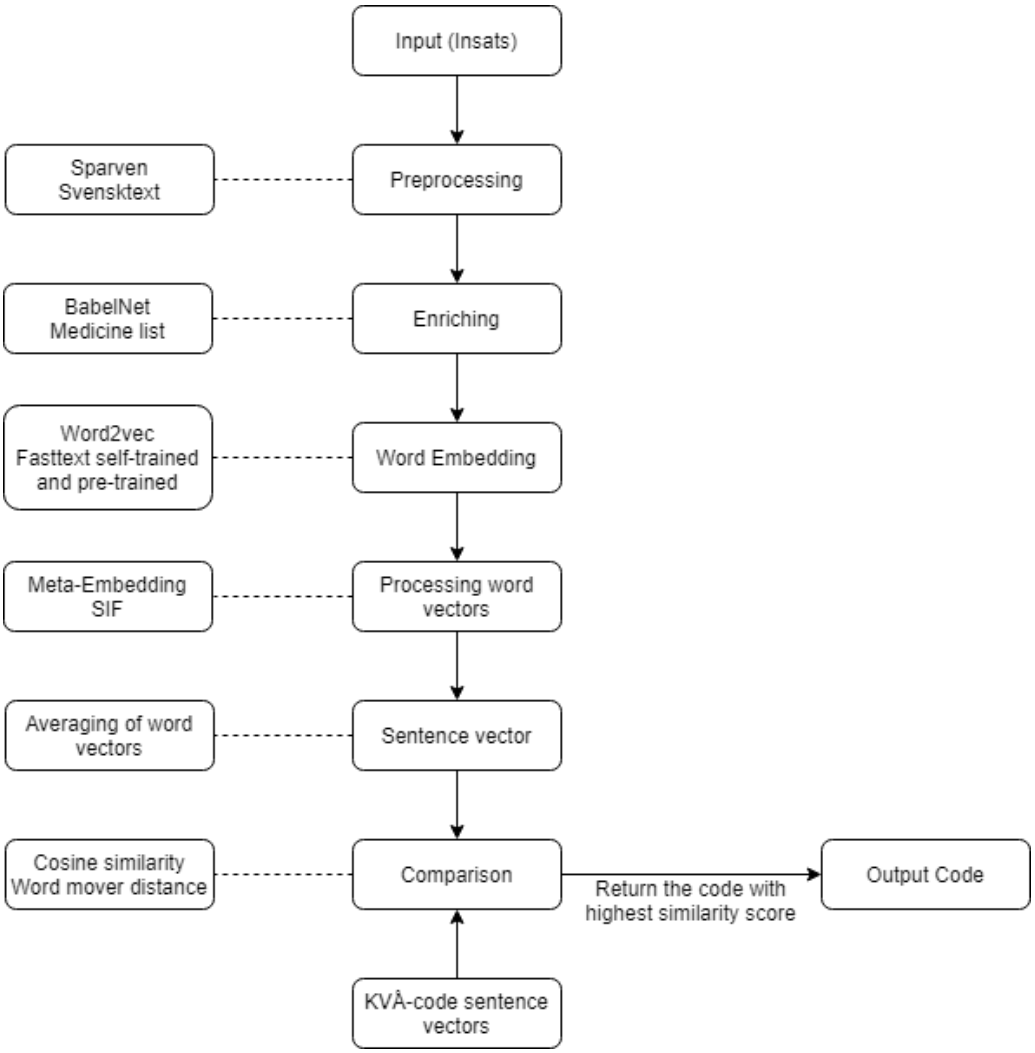

Figure 3.5: The second iteration of the classifier architecture including implemented methods on each part

3.11 Classification of a Clinical Note

A clinical note is classified by first preprocessing the note which is then enriched with external knowledge and then turned into a word embedding. The word embedding is processed, averaged and then compared against the KVÅ-code descriptions. The example in Figure 3.6 show how a clinical note gets processed before finally being classified to a corresponding KVÅ-code.

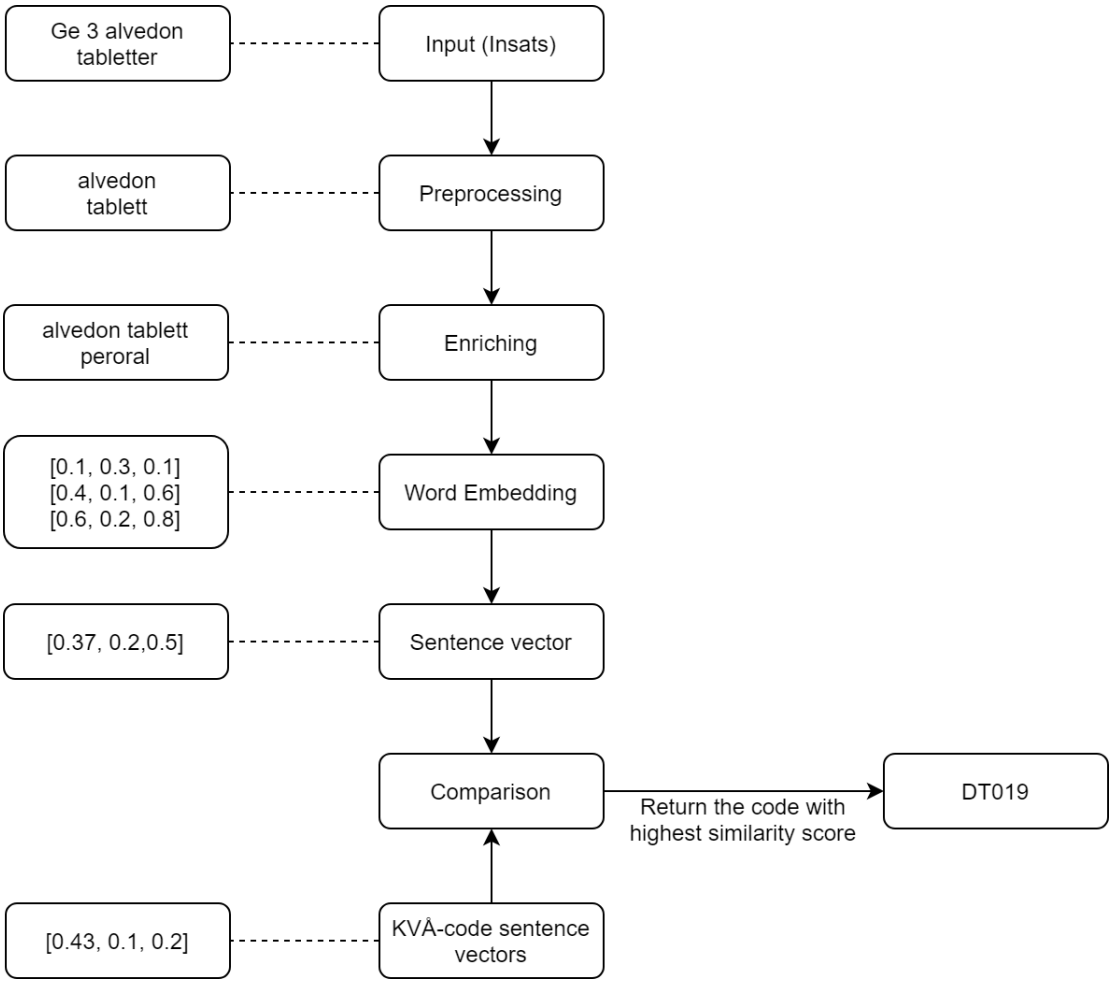

Figure 3.6: An example of how a clinical note gets classified.

3.12 Ethical considerations

The emergence of Artificial Intelligence and Machine Learning has led to numerous debates regarding its effect on our society. Due to the nature of this work, it is evident that it may affect the users and patients the proposed classifier were built for. Thus, the process of designing and implementing the solution had to be carefully thought out.

In this thesis, the data provided by Appva had been made anonymous by removing all personal information from the patients. Although the data has been rendered anonymous and ruled to not be in breach of privacy within the GDPR laws, it is still vital to ensure it is not used carelessly. There have been cases where cross-checking data has been successfully used to identify specific individuals.

The reason for building this classifier was to help licensed personnel correctly identify care procedure codes. The prediction made by the classifier would have a significant impact on what is reported. Ensuring that the wrong code is not reported is of

a serious matter. To ensure that the user does not entirely rely on the classifier, the user must know its error rate. Otherwise, identifying wrong codes could have negative consequences since the KVÅ-codes can and is used for statistics which assists the government in getting a better understanding of care procedures. This, in turn, dictates where and how research and funds are invested.

4
Evaluation and Results

This section provides the result of the implemented methods evaluated individually and collectively. Section 4.3, 4.4 and 4.5 presents the performance of different word embedding models, different types of pre-filter and enricher evaluated on the *validation set*. The testing on the validation set provided result to determine which method should be included and tested on the *test set* for final evaluation.

4.1 Self-labelling of Data

Based on the consensus on how to interpret the Cohen's Kappa score, the reliability of the manual annotation is considered moderate and therefore not good enough to be trustworthy [24]. Thus, no self-labelling was included in the dataset. Instead, new characteristics of the data were found which were later confirmed through a questionnaire sent to medical experts.

Table 4.1: Result of evaluating pre-filtering with and without spellcheck

Annotators	**Cohen's Kappa Score**
Annotator 1 vs licensed personnel	0.4
Annotator 2 vs licensed personnel	0.4
Annotator 1 vs annotator 2	0.6

4.2 Evaluation of Base-line Classifier

As mentioned, a heuristic base-line classifier was created based on knowledge within NLP and intuitive thought. This classifier generates a vocabulary of size 57,434 from the data set. Thereafter, an entry is transformed into a vector representation of size 57,434. Classification of an entry is done by finding the KVÅ-code with the highest cosine similarity. The result of the test set is shown in table 4.2

Table 4.2: Performance of the base-line classifier.

Base-line classifier	**Accuracy**	**Precision**	**Recall**	**F1-score**
BoW + cosine similarity	02.10	58.86	02.10	02.38

4.3 Evaluation of Word Embedding

This section presents the different methods used to implement and evaluate the word vectors. First, the two architectures, Word2vec and Fasttext, were implemented and compared. After that, the training utilising different combinations of data and hyperparameters were evaluated. Lastly, the evaluation of enhancing of word vectors through SIF and Meta-embedding.

4.3.1 Comparison of Measurements

The evaluation of measurements was on the comparison between the data entries and KVÅ-code from the sentence embedding obtained by averaging the word vectors from the Fasttext model. The results in 4.3 indicates that cosine similarity performs better than word mover's distance. Cosine similarity was chosen as the measurement to use in the rest of this thesis.

Table 4.3: Comparison of cosine similarity and word mover's distance.

Model	Measurement	Accuracy	Precision	Recall	F1-Score
Fasttext	Cosine Similarity	**43.220**	77.042	**43.220**	**43.346**
Fasttext	Word Mover's Distance	31.097	88.419	31.097	34.177
Word2Vec	Cosine Similarity	28.108	76.817	28.108	30.875
Word2Vec	Word Mover's Distance	28.559	**90.744**	28.559	27.918

4.3.2 Comparison of Word2Vec and Fasttext

Word2vec and FastText were trained from scratch. The word embedding models were trained on their standard parameters without any modification to ensure a fair comparison. The evaluation was performed on the validation set using cosine similarity as measurement. The results in 4.4 show that Fasttext performed better than word2vec. Furthermore, Fasttext is capable of handling out-of-vocabulary and is better at representing less frequent words. Fasttext was therefore chosen as the Word Embedding of choice and used in further evaluations.

Table 4.4: Comparison of Word2vec and fasttext.

Word Embedding	Accuracy	Precision	Recall	F1-score
Word2Vec	28.108	76.817	28.108	30.875
Fasttext	**43.220**	**77.042**	**43.220**	**43.346**

4.3.3 Comparison of Input Data

Every clinical note contains a mandatory label and an optional description. Therefore, the input data can be the label itself or a combination of both the label and the description. The results show that the addition of description drastically worsen the results as seen in Table 4.5.

Table 4.5: Comparison of using label and description as input data

Input Data	Accuracy	Precision	Recall	F1-score
Label	**43.220**	77.042	**43.220**	**43.346**
Label + Description	31.632	**77.225**	31.632	29.065

4.3.4 Pre-trained Word Embedding

Pre-trained Fasttext models are provided by the Fasttext authors which were trained on data crawled from Wikipedia. The pre-trained model has been evaluated as well as the incremental trained version of this model using the data set. Compared to the result from different self-trained models, the performance of the pre-trained as well as incremental trained models is significantly worse and therefore, will be excluded.

Table 4.6: Pre-trained Fasttext model.

Fasttext model	Accuracy	Precision	Recall	F1-Score
Pre-trained FastText	18.80	36.46	18.80	14.31
Incremental trained	12.69	24.76	12.69	11.12

4.3.5 Comparison of training data

Various combinations of the data were trained on Fasttext to make the best word embedding possible for the domain. As mentioned, the clinical note contains a mandatory label and an optional description. The training always used the KVÅ-descriptions and either the label data or combining it with the description data. The evaluation with cosine similarity shown in 4.7 indicates that the performance increased when utilising all data available to train the embedding.

Table 4.7: Comparison of the data used to train Fasttext.

Dataset	Accuracy	Precision	Recall	F1-score
Label + KVÅ	41.331	76.898	41.331	39.489
Label + Description + KVÅ	**43.220**	**77.042**	**43.220**	**43.346**

4.3.6 Comparison of Dimension Size

Different dimensionalities for the space of word vectors were compared. Four different configurations of Fasttext with different dimensionality were implemented within the range recommended by the Fasttext authors. The results indicate that the dimension changes did not affect the results significantly but increased the memory required by the embeddings. Therefore a dimensionality of 100 was chosen to reduce memory requirements. See Section 4.8.

Table 4.8: Comparison of word embedding of various dimensions.

Dimension	Accuracy	Precision	Recall	F1-Score	Disk Size (GB)
FastText 100	42.036	77.499	42.036	44.166	0.794
FastText 200	41.725	76.183	41.725	42.948	1.59
FastText 300	42.120	76.415	42.120	43.186	2.38
FastText 400	41.782	76.127	41.782	42.863	3.18
FastText 500	42.036	76.152	42.036	43.110	3.97

4.3.7 Evaluation of training epochs

The epoch parameter decides how many times the network is optimised on the given data during training. Large epoch value results in a better representation of the training data. The risk is that it may become overfitted on the provided data. As seen in 4.9, training on one epoch leads to underfitting of the data while training it too much instead overfits, leading to a worse result. Therefore something in between should be used, and epoch two was chosen as the optimal choice.

Table 4.9: Word embedding trained with different epochs.

Epoch	Accuracy	Precision	Recall	F1-Score
1	40.006	71.905	40.006	39.353
2	45.954	71.763	45.954	44.517
3	44.263	75.107	44.263	44.148
4	43.755	75.398	43.755	44.360
5	41.162	75.335	41.162	43.326

4.3.8 Results From Enhancing of Word Embedding

The enhancing of word embedding were made with SIF and Meta-embedding, see 4.10 for the results. SIF-embedding resulted in worse performance than expected since it has in previous research been proven to be quite successful. Meta-embedding also decreased the performance but not drastically in comparison to the SIF-embedding.

Table 4.10: Results of post processing the embedding with SIF and Meta

Embedding Type	Accuracy	Precision	Recall	F1-Score
Standard Embedding	**43.220**	77.042	**43.220**	**43.346**
SIF Embedding	32.685	52.810	32.685	31.775
Meta Embedding	42.825	**77.421**	42.825	41.865

4.4 Evaluation of Pre-filtering

As expected, many of the entries that contain a medicine can be directly assigned a KVÅ-code. The percentage of data entries containing a medicine name and the

accuracy indicates the benefit of implementing a pre-filter.

Table 4.11: Result of evaluating pre-filtering with and without spellcheck

Dataset	**N.o input**	**Accuracy**
Matching + Spelling ratio = 0.8	1650	**65.15**
Matching + Spelling ratio = 0.7	1720	63.13
Matching + Spelling ratio = 0.6	**2454**	49.84

The pre-filter was evaluated on the Danderyd data set. In table 4.11, *N.o input* represents the number of entries found with a medicine name. *Matching* represents the pre-filter model where the only word fully matched with a medicine name is accepted and *Misspelling ratio* represents a model that accepts misspelt medicine name within the limit.

In case the pre-filter suggests a KVÅ-code, the classifier will skip over the other parts. Therefore, in case the pre-filter causes an error, the other parts will not have the opportunity to correct it. Therefore, in the second iteration, the information in use of pre-filter was instead applied to the enricher. The pre-filter was also removed to eliminate the potential flaw of pre-filter. The trade-off can be seen in the section 4.6 where both pre-filer and enriching are evaluated and compared.

4.5 Evaluation of Enriching

The knowledge used to enrich the data entries came from two sources. Läkemedelsverket and 1177 where a medicine list with additional information about medicine was obtained. BabelNet, where synonyms and definition of words were retrieved. The results enriched using the medicine list proved to be useful and increased the results significantly, as seen in 4.12.

Table 4.12: Enriching using Medicine List

Enriching Source	**Accuracy**	**Precision**	**Recall**	**F1-Score**
No enriching	43.220	77.042	43.220	43.346
$Medicine_1$ (Route of Administration)	**58.726**	**80.567**	**58.726**	**56.752**
$Medicine_2$ (Original Form)	46.687	78.087	46.687	45.709

Enriching with BabelNet, on the other hand, proved to be detrimental to the classification as seen in 4.13.

Table 4.13: Enriching using BabelNet

Enriching Source	Accuracy	Precision	Recall	F1-Score
No enriching	**43.220**	**77.042**	**43.220**	**43.346**
BabelNet Senses	28.644	65.423	28.644	22.463
BabelNet Glosses	30.364	72.352	30.364	25.741
BabelNet Senses + Glosses	26.840	63.419	26.840	19.366

4.6 Final evaluation

The final classifier was evaluated on the test set consisting of 102 unique kvå-codes and 2521 entries. Since the clsasifier is built from self-contained parts, the classifier was evaluated from the parts themselves to the complete classifier. Table 4.14 shows that the best performing classifier was the combination of the embedding with pre-filter and the list which obtained the highest score overall in every measurement.

4.6.1 Evaluation of Common and Infrequent KVÅ-codes

Since the frequency of the KVÅ-codes in the test set is imbalanced, an overall measurement by itself, cannot provide a proper indication of the performance. The classifier was evaluated using the entries belonging to the 20 most frequent and 20 least frequent KVÅ-codes. This provides insights into how well the classifier generalise. See 4.15 for the results.

Table 4.15: The results of the 20 most common and uncommon KVÅ-codes

Dataset	Accuracy	Precision	Recall	F1-Score
KVÅ-code (20 most common)	61.993	73.442	61.993	57.137
KVÅ-code (20 least common)	30	95	30	30

The Figure 4.1 shows the confusion matrix for the 20 most common KVÅ-codes. Every square with coordinate X and Y and a number P represents the proportion P of entries belong to KVÅ-code Y but assigned to KVÅ-code X. The diagonal line represents the entries that are correctly assigned by the proposed classifier. The squares that are not on the diagonal line are assigned wrong KVÅ-code. From this matrix, it shows that the classifier performs quite well on average. It is also possible to identify a couple of frequent codes which seem always to be classified wrongly, *AF015, AU116 and QA009*.

The Figure 4.2 plots the results of the 20 least common KVÅ-codes. It seems that the classifier has a problem with infrequent codes, which was expected. However, it still manages to classify some of them correctly.

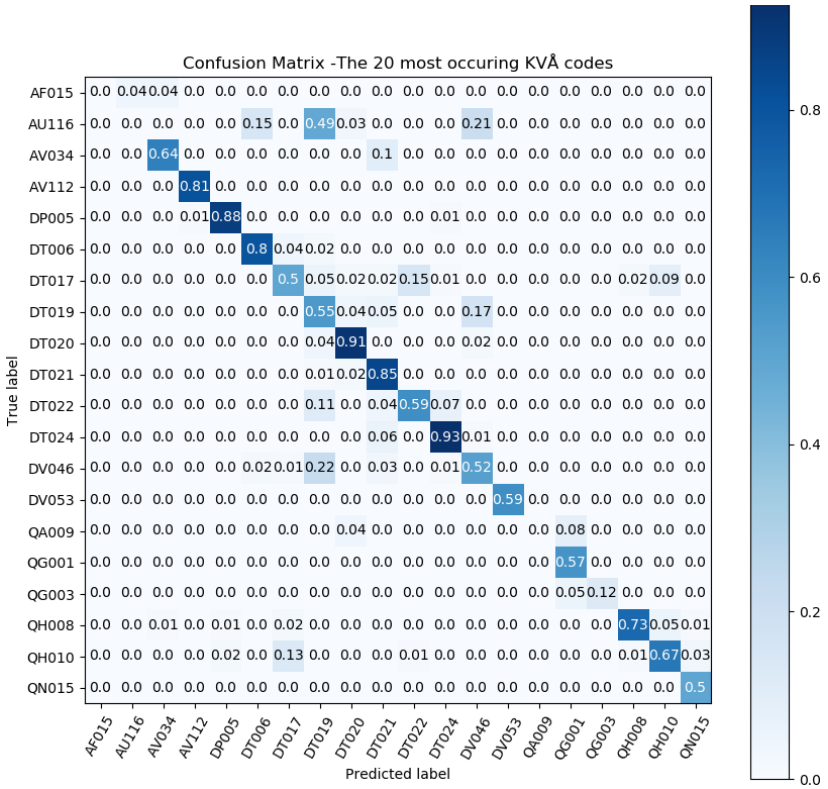

Figure 4.1: Confusion matrix on top 20 most common codes

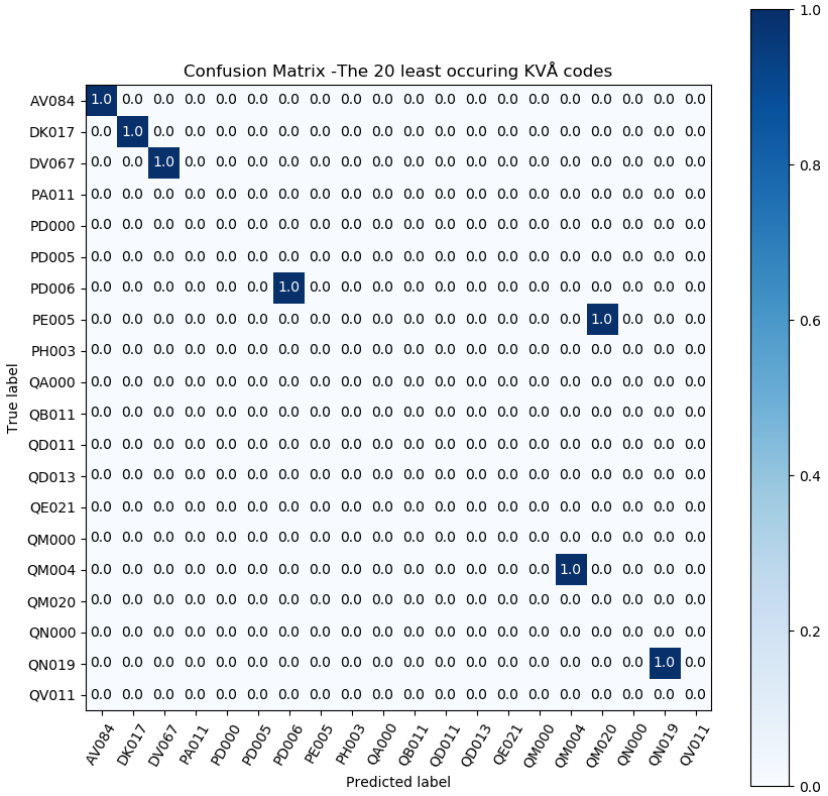

Figure 4.2: Confusion matrix on the 20 least common codes.

4.6.2 Evaluation of the Highest Scored KVÅ-codes

Instead of only allowing the classifier to suggest one KVÅ-code, an experiment was done where the classifier is allowed to suggest 3 and 5 KVÅ-codes for each clinical note. The result of this experiment can be seen in Table 4.16. As expected, the accuracy improves when the number of compared KVÅ-codes increases. However, comparing the top 3 KVÅ-codes increased the accuracy by 13.804% while top 5 has further increased it by 5.514%. This indicates that often in cases where the classifier has assigned a wrong KVÅ-code, the correct KVÅ-code is within the highest scored KVÅ-codes and most likely within the top 3.

Table 4.16: Results from evaluating the highest scored KVÅ-codes.

Recommendations	**Accuracy**
Top 5 KVÅ-codes	78.937
Top 3 KVÅ-codes	73.423

4.7 Experimenting With the Field *List*

KVÅ-code is not the only category that an entry can be assigned to. Each entry also has a *list* assigned to it by the organization where the licensed personnel work. By analysing the codes in each *list*, it was possible to determine which code were frequent in the different *lists*. By limiting the KVÅ-codes a data entry is compared with by its assigned *list*, the accuracy increased as seen in 4.17.

To ensure fairness and prevent bias, the test set was divided into two sets with a ratio of 80/20. 80% were used to identify the frequency of the KVÅ-codes while the 20% were used to evaluate the performance of using list. Each version has been tested 5 times to reduce the effect of randomness in dividing the test set.

Table 4.17: Result from using different components

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Standard Embedding + Prefilter	61.069	70.809	61.069	60.123
SIF-embedding + Prefilter	58.812	73.748	58.812	57.683
Meta-embedding + Prefilter	61.426	70.865	61.426	60.084
Standard embedding + Enriching ($Medicine_1$)	60.911	70.507	60.911	59.633
SIF-embedding + Enriching ($Medicine_1$)	50.337	66.664	50.337	48.021
Meta-embedding + Enriching ($Medicine_1$)	60.832	70.883	60.832	59.547

4.8 Proposed Classifier Architecture

The proposed classifier with the chosen method based on the overall performance can be seen in Figure 4.3.

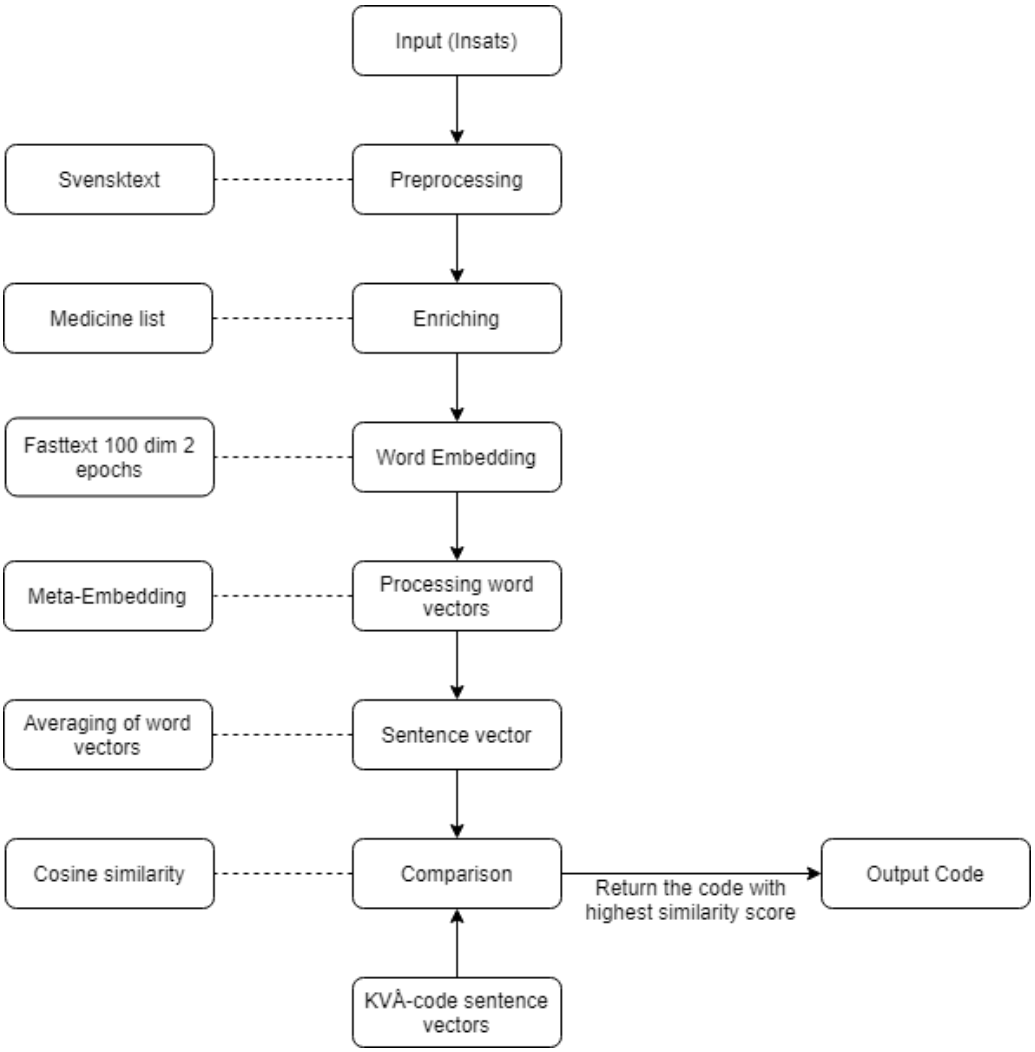

Figure 4.3: The proposed classifier architecture with chosen method in each part

Table 4.14: Result from using different components

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Base-line classifier	02.10	58.86	02.10	02.38
Standard embedding	44.506	57.179	44.506	41.106
SIF-embedding	41.333	54.695	41.333	37.841
Meta-embedding	43.911	56.207	43.911	40.457
Standard Embedding + Prefilter	59.342	**66.775**	59.342	58.878
SIF-embedding + Prefilter	57.993	66.328	57.993	58.222
Meta-embedding + Prefilter	**59.619**	66.553	**59.619**	**59.043**
Standard embedding + Enriching ($Medicine_1$)	58.231	65.403	58.231	57.780
SIF-embedding + Enriching ($Medicine_1$)	51.051	61.108	51.051	50.312
Meta-embedding + Enriching ($Medicine_1$)	58.390	65.235	58.390	57.863
Standard embedding + Enriching (BabelNet glosses)	37.604	48.529	37.604	33.863
SIF-embedding + Enriching (BabelNet glosses)	40.500	48.153	40.500	35.444
Meta-embedding + Enriching (BabelNet glosses)	35.462	49.563	35.462	32.243
Standard Embedding + Prefilter + Enriching (BabelNet glosses)	56.090	61.410	56.089	54.603
SIF-embedding + Prefilter + Enriching (BabelNet glosses)	58.033	64.811	58.033	57.826
Meta-embedding + Prefilter + Enriching (BabelNet glosses)	56.208	62.013	56.208	54.977
Standard Embedding + Enriching ($Medicine_1$ + BabelNet glosses)	43.832	44.207	43.832	38.177
SIF-embedding + Enriching ($Medicine_1$ + BabelNet glosses)	49.186	54.604	49.186	46.323
Meta-embedding + Enriching ($Medicine_1$ + BabelNet glosses)	44.427	45.911	44.427	39.342

5

Discussion

This chapter reflects and discuss the purpose of this paper, the methods used and the results obtained.

5.1 The Base-line Classifier

Choosing a base-line classifier to compare the classifier against was not straightforward. Supervised-methods were not feasible as there is a lack of labelled data. Instead, the choice fell upon a classifier which utilises bag-of-words and cosine similarity since they are the building blocks of the purposed classifier and have shown to work well in downstream supervised tasks.

By analysing the results of the base-line classifier, a discovery was made. The main reason for the poor result of the base-line classifier is that it is trying to find matching words between the clinical note and the KVÅ-descriptions. As mentioned, the size of the resulting BOW vectors is 57,434. Since the text is short, with an average of 11 words, the resulting vector representation is sparse as it is filled with zeros and only 11 ones. After that, the similarity is measured using cosine similarity, as shown in equation 2.4.

The numerator in the cosine formula is simply the number of matching words between the input text and a KVÅ-code description. Since the dimension of a vector representation is enormous compared to the average number of words per text, the probability that a pair of input text and KVÅ-code description has matching words is low. Thus the majority of comparisons will result in a cosine score of zero, making the comparison useless. Due to this, 1247 out of 2521 entries were not assigned a KVÅ-code. Thus, rendering the base-line useless on around 50% of the clinical notes.

Furthermore, the denominator is the square root of the number of unique words in the vectors. A KVÅ-code with a shorter description will, therefore, have a higher similarity even if multiple KVÅ-codes have the same number of matching words. Finally, the matching words are not always the keywords. An example is the word "sond" which is a tube used to deliver food and medicine into the body. Since this word only exists in the KVÅ-code DJ010, the entries containing "sond" were assigned to DJ010 when only two of them should have been assigned to it.

5.2 Self-labelling and the Resulting Questionnaire

The self-labelling did not work as we expected because it is hard to label correctly without the medical expertise required. However, the characteristics of the data that were found during the self-labelling process were confirmed by medical personnel in the questionnaire that was sent. Based on their answers, the assumptions were confirmed, which made it possible to draw the following conclusions:

- An entry may have more than one correct KVÅ-code.
- There are clinical notes with wrongly assigned KVÅ-code.
- The assigned KVÅ-code on some entries are unusual and not correct based on the clinical note.

The new conclusions led to consequences which could not have been foreseen. Because an entry may have more than one correct KVÅ-code, the problem is not only a multi-category problem but also a multi-label problem. Since the test and validation set only contain one assigned KVÅ-code per entry it did no longer give a fair representation since an assigned KVÅ-code by the classifier could now be correct but wrong when compared to the assigned KVÅ-code. The wrongly assigned entries would also lead to false negatives, and the unusual assigned KVÅ-codes would be impossible to classify correctly. Thus, the results obtained may be better than presented in the results.

5.3 Pre-Filter

The observation that some entries are more straightforward to classify than others led to the addition of a pre-filter module. Utilising this knowledge made it possible to improve the classifier through the extension of a pre-filter that covered medicine intake. While it proved to help in the overall accuracy, there were cases where it introduced errors in otherwise correct predictions. As mentioned, in cases where the medical prescription is not the central part of the described medical procedure, the pre-filter will assign an inaccurate KVÅ-code. This issue leads to a restriction in the pre-filter due to the loss of generalisation.

5.4 Enriching

The medicine list obtained from Läkemedelsverket for enriching proved to be effective in improving the classification. Instead of assigning a KVÅ-code based on keywords as in the pre-filter, the enriching assists the classification by adding useful information while letting the classifier determine the suitable KVÅ-code. Since it performed almost on par with the pre-filter while being less restricted compared to the pre-filter, it may be the better alternative.

Surprisingly, enriching with BabelNet made the proposed classifier worse with a significant loss of accuracy. The addition of synonyms and descriptors seemed to introduce more noise than it helped. In some ways, this makes sense since synonyms are relatively close in the vector space making the addition of them redundant. After

analysing, it shows that some words were enriched with out-of-domain knowledge which introduced undesired noise. Furthermore, BabelNet is quite limited when it comes to Swedish since it was not able to provide synonyms and definitions for the majority of the vocabulary. This leads to a problem of disproportionate enriching where not all words become enriched. The obtained sentence vector may therefore have had its weight shifted towards words that had synonyms and descriptions.

5.5 Word Embedding

The choice of using word embedding was due to how well it is able to capture semantics between words. Since the classification is based on comparing the KVÅ-descriptions with clinical notes, the semantics between words were necessary. After comparing, Fasttext proved to perform better than Word2Vec. This may be due to how Fasttext considers words in the form of n-grams, making it better on morphologically abundant languages. While Swedish may not be considered a morphologically rich language, the data set contains clinical notes which are rarely well structured. The notes often have misspellings which can be seen as a morphology of the corrected spelt word. Thus, Fasttext can find the similarity between the correct word and misspellings while Word2Vec would consider them to be unique.

5.5.1 Pre-trained Fasttext

The Word Embedding was trained from scratch to represent the medical domain of the data in its best form. This proved to be correct since pre-trained Fasttext performed significantly worse. Furthermore, incremental learning did not improve the performance of the pre-trained model. The reason seems to be the difference in features between the data used to train the pre-trained Fasttext and the data in use:

- Structured and unstructured text. The corpus from Wikipedia is considered structured text as it follows syntactic rules of a language. Oppositely, the text in the data set consist mainly of keywords and does not follow the rules as the individual writing them are trying to make them short.
- Difference in the domain. Since the corpus from Wikipedia is from all possible domains, words that have several meanings will be located differently in the vector space to justify the contexts.

5.6 Enhancing of Word Embedding

Both methods used for enhancing of the vectors, SIF- and meta-embedding showed worse performance on the validation set. However, evaluation of the test set shows a difference in the performance of SIF- and meta-embedding on different combinations of implemented parts. Compared to standard Fasttext embedding, SIF improves the performance of the classifier significantly when enriching with BabelNet's glosses is in use. However, when enriching with BabelNet's glosses is not part of the classifier, the performance of SIF-embedding decreases significantly. As mentioned, enriching

using BabelNet is not substantial as there are words without glosses or senses. Furthermore, since many of them are out-of-domain, thus rare words, SIF-embedding gives the enriched words a lower weight. Therefore, SIF-embedding is actually not improving the performance of a classifier with BabelNet based enricher but reduce the error introduced which the standard embedding cannot handle.

Meta-embedding on the other hand, performs better when the pre-filter or enriching with medicine list is used. The question is if the performance gain is due to randomness and the error of margin or if it helped the classifier. However, it is an exciting result and indicates that meta-embedding should not be ignored.

5.7 Building Sentence Embeddings

The decision to average word vectors proved to be useful and shows that it is an effective way of concentrating the sentence information while keeping the dimension size the same as a single word vector. The side effect of this is the inherent simplification of the representation. For example, word order was no longer considered, meaning that two sentences with identical words in a different order would have the same score. Thus, the structure of the sentence is lost which is not ideal.

5.8 Label or Description

Since the data entries were short, it seemed that utilising all possible information would be beneficial. Instead, the results became worse when combining the label and description into one input data. The label itself may, therefore, contain enough information to be classified correctly. This indicates that including more information may not always be the better option and that trying to concentrate on the keywords may work.

Something to note is that some entries contained labels which did not indicate anything. For example, there were notes where the label was "other", and the description contained the majority of information. Thus, by using the label as input, these entries would never be classified correctly with the proposed classifier. A similar outcome is found in section 5.4 where enriching with glosses and senses from BabelNet has worsen the performance. Thus the quality of the text is more important than the quantity even in cases where the input text is concise.

5.9 Experimenting on the Field *List*

The usage of the *List* field improved the performance of the classifier. One problem with this approach is that each respective organisation customises their list categories. Thus there is no logical coherence, where many of the lists overlap but were considered different. Since this is just an experiment to determine the possibility of

improvement using the *list* field, no efforts were put on trying to figure out which list were distinct.

5.10 The Final Evaluation

The final evaluation showed that the classifier managed to achieve an accuracy of 59.619%. The result may look below average for a classifier in general, but when compared to other solutions to classify KVÅ-codes, the achieved results are considered good. The question remains if relying on this classifier is acceptable, which only the healthcare industry can answer. From evaluating the highest scored KVÅ-codes, a considerable improvement in accuracy was achieved. This indicates that the classifier can be utilised as a recommendation system which would be able to guide and simplify the classification of KVÅ-codes.

5.11 Evaluation Metrics

In addition to accuracy, other metrics are used to ensure the quality of the evaluation. One thing noticed is that when measuring the performance of different methods is that accuracy and recall is always the same. Since the weighted average is used to generate an overall evaluation score from the score of each category, the overall recall score is computed as bellow:

$$weighted\_average\_recall = \sum_{n=1}^{k} recall_n * W_n \tag{5.1}$$

$$= \sum_{n=1}^{k} recall_n * \frac{TP_n + FN_n}{N} \tag{5.2}$$

$$= \frac{\sum_{n=1}^{k} recall_n * (TP_n + FN_n)}{N} \tag{5.3}$$

$$= \frac{\sum_{n=1}^{k} \frac{TP_n}{TP_n + FN_n} * (TP_n + FN_n)}{N} \tag{5.4}$$

$$= \frac{\sum_{n=1}^{k} TP_n}{N} \tag{5.5}$$

Here k is the number of categories,N is the number of data entries and W_n is the weight of n-th category. Since the overall accuracy is simple the number of correct classified entries divided by the number of data entries, overall recall and overall accuracy is the same.
Another detail is that F1 score is sometimes lower than both precision and recall, despite it is the harmonic mean of them. The reason is that the weighted average is used to compute the overall score, thus categorises that covering many entries with lower scores will make the final overall F1-score lower. Lastly, precision is always higher than other metrics. While accuracy and other metrics are considerable lower, the high precision score indicates a stable classification. Among all entries classified to a KVÅ-code, precision is the percentage of entries that are correctly classified.

5.12 Future Work

The proposed classifier shows promising results, but some things can still be improved. Below we expose some ideas that could be beneficial from further research.

- A wide variety of word representation methods exist, and the ones employed here were strong candidates. However, there exist other options which have shown excellent results within the English domain and in downstream supervised tasks. Utilising those advanced architectures such as BERT and Universal Sentence Encoder would be of interest and could be beneficial to the classifier. They also allow for a more advanced sentence representation which can take word order into account.

- Post-processing of embedding has shown success in removing noise. Since enriching seemed to introduce noise, investigating post-processing could be beneficial.

- Investigate ways to solve the skewed data with many infrequent categories. While this paper focused on making a general classifier which can classify all categories, focusing on the rare cases and finding out what could improve these are of great importance and need to be researched further.

- Look for more relevant data to train the embeddings. As shown in the results, utilising more in-domain data proved to be efficient while the pre-trained out of domain models were too general to generate good results. Thus, finding domain-specific data in Swedish would benefit the classifier.

- Investigate alternative Meta-Embeddings such has Locally Linear Meta-Embedding since the results show that it may be beneficial.

6
Conclusion

This thesis aimed to research and identify how an unsupervised method could be used for the classification of very short clinical notes in Swedish. An unsupervised classifier for the classification of short clinical notes has been implemented based on previous research and related work. The model was evaluated on the test data provided by Appva on 108 different KVÅ-codes.

Two major iterations were made where one classifier utilised a pre-filter which managed to classify simpler entries directly. The second classifier utilised an enricher by incorporating additional knowledge from other sources. Although the pre-filter based classifier performs better than the one using enriching, it is much more restricted due to how the pre-filter works. Thus the final proposed classifier uses enriching since the small improvement of pre-filter does not outweigh the loss of generalisation.

The proposed classifier manages to find relations between a clinical note and a KVÅ-description in a similar way a human would approach the problem. This is an interesting distinction from supervised-methods where mapping would be learned directly from a clinical note rather than looking for semantic similarity.

From the results obtained, it can also be concluded that the quality of the text to be classified is significantly more important than the quantity of data. This applies to both texts used in training and evaluating the proposed classifier and enriched text.

Bibliography

[1] Socialstyrelsen. "Klassificering och koder". In: *Social Styrelsen* (2019-08).

[2] Chefsjurist Pär Ödman. "Gemensamma författningssamlingen avseende hälso- och sjukvård, socialtjänst, läkemedel, folkhälsa m.m." In: *Social Styrelsen* (2017-12).

[3] Socialstyrelsen. "Klassifikation av medicinska åtgärder 2020". In: *Social Styrelsen* (2020-01).

[4] Fabrizio Sebastiani. "Machine Learning in Automated Text Categorization". In: *ACM Comput. Surv.* 34.1 (2002-03), pp. 1–47. ISSN: 0360-0300. DOI: 10.1145/505282.505283. URL: https://doi.org/10.1145/505282.505283.

[5] B. J. Jansen and S. Rieh. "The Seventeen Theoretical Constructs of Information Searching and Information Retrieval". In: *the American Society for Information Sciences and Technology* (2010), pp. 1517–1534.

[6] Mita K Dalal and Mukesh A Zaveri. "Automatic text classification: a technical review". In: *International Journal of Computer Applications* 28.2 (2011), pp. 37–40.

[7] W. Weng et al. "Medical subdomain classification of clinicalnotes using a machine learning-basednatural language processing approach". In: (2017). URL: https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-017-0556-8.

[8] Weidi Xu and Haoze Sun. "Semi-supervised Variational Autoencoders for Sequence Classification". In: *ArXiv* abs/1603.02514 (2016).

[9] Zied Haj-Yahia, Adrien Sieg, and Léa A. Deleris. "Towards Unsupervised Text Classification Leveraging Experts and Word Embeddings". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019-07, pp. 371–379. DOI: 10.18653/v1/P19-1036. URL: https://www.aclweb.org/anthology/P19-1036.

[10] Ponti M.A Fernandes R. "Machine Learning - A Practical Approach on the Statistical Learning Theory". In: (2018).

[11] Xin Yao. "Evolving artificial neural networks". In: *Proceedings of the IEEE* 87.9 (1999), pp. 1423–1447.

[12] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[13] Eric Brill. "A simple rule-based part of speech tagger". In: *Proceedings of the third conference on Applied natural language processing*. Association for Computational Linguistics. 1992, pp. 152–155.

[14] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[15] Min Chen, Shiwen Mao, and Yunhao Liu. "Big data: A survey". In: *Mobile networks and applications* 19.2 (2014), pp. 171–209.

[16] Tomas; et al. Mikolov. "Distributed representations of words and phrases and their compositionality". In: (2013).

[17] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[18] Armand Joulin et al. "Bag of tricks for efficient text classification". In: *arXiv preprint arXiv:1607.01759* (2016).

[19] Quoc Le and Tomas Mikolov. "Distributed representations of sentences and documents". In: *International conference on machine learning*. 2014, pp. 1188–1196.

[20] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. "A simple but tough-to-beat baseline for sentence embeddings". In: (2016).

[21] Wael Gomaa and Aly Fahmy. "A Survey of Text Similarity Approaches". In: *international journal of Computer Applications* 68 (2013-04). DOI: 10.5120/11638-7118.

[22] Matt Kusner et al. "From word embeddings to document distances". In: *International conference on machine learning*. 2015, pp. 957–966.

[23] McHugh ML. "Interrater reliability: the kappa statistic". In: *Biochemia medica* (2012), pp. 276–282.

[24] J. Richard Landis and Gary G. Koch. "The Measurement of Observer Agreement for Categorical Data". In: *Biometrics* 33 (1977). ISSN: 0006341X, 15410420. URL: http://www.jstor.org/stable/2529310.

[25] J. Richard Landis and Gary G. Koch. "The Measurement of Observer Agreement for Categorical Data". In: *Biometrics* 33.1 (1977), pp. 159–174. ISSN: 0006341X, 15410420. URL: http://www.jstor.org/stable/2529310.

[26] K. Krippendorff. "Validity in content analysis". In: *Computer Strategien für die kommunikationsanalyse* 33.1 (1980), pp. 69–112. URL: http://repository.upenn.edu/asc_papers/291.

[27] Mark Hughes et al. "Medical text classification using convolutional neural networks". In: *Stud Health Technol Inform* 235 (2017), pp. 246–50.

[28] Yu Chen and Mohammed J. Zaki. *KATE: K-Competitive Autoencoder for Text*. 2017. arXiv: 1705.02033 [stat.ML].

[29] Daniel Cer et al. "Universal sentence encoder". In: *arXiv preprint arXiv:1803.11175* (2018).

[30] Yuhua Li et al. "Sentence Similarity Based on Semantic Nets and Corpus Statistics". In: *IEEE Transactions on Knowledge and Data Engineering* 18 (2006-09), pp. 1138–1150. DOI: 10.1109/TKDE.2006.130.

[31] Wenpeng Yin and Hinrich Schütze. "Learning Word Meta-Embeddings". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, 2016-08, pp. 1351–1360. DOI: 10.18653/v1/P16-1128. URL: https://www.aclweb.org/anthology/P16-1128.

[32] Danushka Bollegala, Kohei Hayashi, and Ken-ichi Kawarabayashi. "Think Globally, Embed Locally - Locally Linear Meta-embedding of Words". In:

CoRR abs/1709.06671 (2017). arXiv: 1709.06671. URL: http://arxiv.org/abs/1709.06671.

[33] Joshua Coates and Danushka Bollegala. "Frustratingly Easy Meta-Embedding – Computing Meta-Embeddings by Averaging Source Word Embeddings". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, 2018-06, pp. 194–198. DOI: 10.18653/v1/N18-2031. URL: https://www.aclweb.org/anthology/N18-2031.

[34] Jiaqi Mu, Suma Bhat, and Pramod Viswanath. "All-but-the-Top: Simple and Effective Postprocessing for Word Representations". In: *CoRR* abs/1702.01417 (2017). arXiv: 1702.01417. URL: http://arxiv.org/abs/1702.01417.

[35] Shahrokh et la. "Effect of separate sampling on classification accuracy". In: *Bioinformatics* (1977), pp. 242–250. DOI: doi:10.1093/bioinformatics/btt662.

[36] Jeffrey David Ullman Anand Rajaraman. "Data Mining". In: (2011), pp. 1–17. DOI: 10.1017/CBO9781139058452.002.

[37] Lars Borin et al. "Sparv: Språkbanken's corpus annotation pipeline infrastructure". In: *SLTC 2016. The Sixth Swedish Language Technology Conference, Umeå University, 17-18 November, 2016*. 2016.

[38] Peter M. Dahlgren. *Svensk text*. Svenska. https://github.com/peterdalle/svensktext. 2018-12. URL: https://snd.gu.se/sv/catalogue/study/ext0278 (visited on 2018-12-20).

[39] Mikolov et la. "Advances in Pre-Training Distributed Word Representations". In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*. 2018.

[40] Läkemedelsverket. "Om Läkemedelsfakta". In: (2020). URL: https://www.lakemedelsverket.se/sv/om-lakemedelsfakta#hmainbody1.

[41] 1177. "Olika sätt att ta läkemedel". In: (). URL: https://www.1177.se/behandling--hjalpmedel/behandling-med-lakemedel/olika-satt-att-ta-lakemedel.

[42] Kavitha A, Mintu Philip, and K Lubna. "Comparative analysis of similarity measures in document clustering". In: (2013-12), pp. 857–860. DOI: 10.1109/ICGCE.2013.6823554.

[43] Yutaka Sasaki et la. "The truth of the F-measure". In: (2007).

A

Complete Heat Map

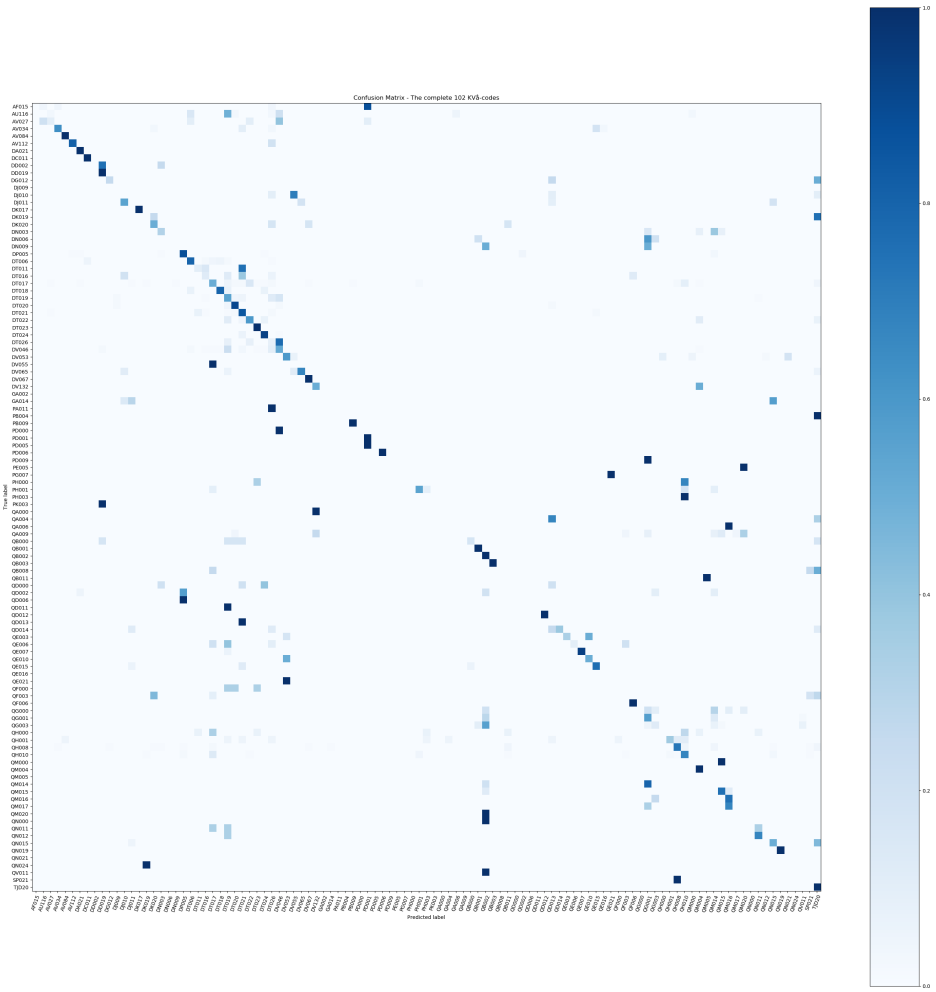

Figure A.1: The complete heatmap