UPPSALA
UNIVERSITET

UPTEC STS 19034
Examensarbete 30 hp
Juni 2019

# Drug Name Recognition in Reports on Concomitant Medication

Arvid Gräns

UPPSALA
UNIVERSITET

**Teknisk- naturvetenskaplig fakultet**
**UTH-enheten**

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

# Abstract

## Drug Name Recognition in Reports on Concomitant Medication

*Arvid Gräns*

This thesis evaluates if and how drug name recognition can be used to find drug names in verbatims from reports on concomitant medication in clinical trial studies. In clinical trials, reports on concomitant medication are written if a trial participant takes other drugs than the studied drug. This information needs to be coded to a drug reference dictionary. Coded verbatims were used to create the data needed to train the drug name recognition models in this thesis. Labels for where in each verbatim the coded drugs name was, were created using a Levensthein distance. The drug name recognition models were trained and tested on verbatims with labels.

Drug name recognition was performed using a logistic regression model and a bidirectional long short-term memory model. The bidirectional long short-term memory model performed the best result with an F1 score of 82.5% on classifying which words in the verbatims that were drug names. When the results were studied from case to case, they showed that the bidirectional long short-term memory classifications sometimes outperformed labels it was trained on in single word verbatims. The model was also tested on manually labelled golden standard data where it performed an F-score of 46.4%. The results indicate that a bidirectional long short-term memory model can be implemented for drug name recognition, but that label reliability is an issue in this thesis.

Handledare: Ziring Tawfique
Ämnesgranskare: Niklas Wahlström
Examinator: Elísabet Andrésdóttir
ISSN: 1650-8319, UPTEC STS 19034

# Populärvetenskaplig sammanfattning

De senaste årens tekniska utveckling har skapat förutsättningar för att processa och lagra stora mängder data. Detta förändrar sättet vi arbetar på och möjligheterna att utvinna information. Inom kliniska prövningar då nya läkemedel testas och utvecklas samlas stora mängder data in. Bland den datan finns information ifall studiedeltagande personer tar andra läkemedel än studieläkemedlet under den kliniska prövningen. Sådan information skrivs ned och behöver sedan kodas mot en läkemedelsordbok. Kodningen av vilket läkemedel utöver studieläkemedlet en studiedeltagare har tagit, görs för att standardisera datan mot en global standard och säkerställa vilket läkemedel studiedeltagaren tagit. Arbetet med att koda den här datan mot en ordbok över alla läkemedel, görs av en medicinsk kodare. Denne läser och tolkar informationen samt kodar mot läkemedelsordlistan vilket läkemedel studiedeltagaren tagit. När all data och alla kodningar samlas, skapar detta över tid stora mängder data.

För att processa stora datamängder har maskininlärning blivit ett område inom datorvetenskapen som i allt större utbredning används inom industri och hos företag. Maskininlärning innehåller metoder för att få datorer att skapa egna regler för att exempelvis klassificera data till olika kategorier. Det finns ett antal underkategorier inom maskininlärning och klassificering varav läkemedelsigenkänning är en. Vilka ord som är läkemedel i olika typer av text har undersökt med hjälp av maskininlärning med positiva resultat i flera rapporter.

Det arbete en medicinsk kodare gör och vad läkemedelsigenkänning ämnar att göra liknar varandra. Likheten i de två sysslorna skapade intresset för att i det här arbetet se hur läkemedelsigenkänning kan användas för att hitta vilka ord i data, inhämtad inom kliniska studier, som är läkemedelsnamn. Förhoppningen har varit att arbetet i framtiden kan användas som ett steg inom den medicinska kodningen.

För att tillämpa läkemedelsigenkänning på tidigare beskrivna data har olika klassificerare tränats på stora mängder redan kodade data. Klassificerarna har under träningen skapat regler och mönster för att göra klassificeringar av tidigare osedda och ej kodade data efter träningen. De olika klassificerarnas prestanda mäts med hjälp av ett F1-mått som mäter hur väl varje ord i data klassificeras som läkemedelsnamn. Den bäst presterande klassificeraren är ett neuralt nätverk som har förmågan att ta med kontext i sin klassificering. Den presterar ett F1-mått på 82,7% vilket är i nivå med F1-mått nådda i tidigare liknande studier.

När den bäst presterande klassificeraren testas på data av högre kvalitet än den tränat på, blir det däremot tydligt att för att klassificerarna i detta arbete ska bli tillämpbara inom det dagliga arbetet för en medicinsk kodare, krävs vidare arbete med att skapa bättre data att träna klassificerarna på.

# Table of content

# 1. Introduction

As a part of World Health Organization programme for international drug monitoring, Uppsala Monitoring Centre (UMC) is leading the way on patient safety through safer medicines. The science of safer medicines is known as pharmacovigilance. As a part of the work in pharmacovigilance, certain clinical data produced as part of medical research needs to be collected. When drugs are tested on humans during medical research, data is collected if a trial participant takes other drugs than the study drug, which is the drug that is tested during medical research. The use of other drugs than the study drug is called concomitant medication. Information on concomitant medication is collected by a doctor who notes what drug alongside the study drug the trial participant has taken (verbatim), how the drug was taken (route), and why the drug was taken (indication). This information is called a report on concomitant medication. To be standardised over the world, this report needs to be coded to a global common standard on drugs. UMC has developed and maintains the most comprehensive and actively used drug reference dictionary in the world called WHODrug (UMC, 2019). The job of coding this information is done by a coding specialist, who is a specially trained pharmacist or nurse, who codes the information in the concomitant medication report to WHODrug. This process is called medical coding and ensures the reliability of reports on concomitant medication and which drug the trial participant has taken.

The process of medical coding is often costly and time consuming as it takes manual labour by trained specialists. Machine learning (ML) is an area in data science which contains methods for computers to learn from data. ML has been implemented to increase efficiency in several applications in recent years (Columbus, 2018). A subtask of ML is Drug Name Recognition (DNR) which seeks to recognize drug names in unstructured texts (Liu *et al.*, 2015). Some of the typical challenges in DNR are that drug names coincide with regular English words, they are often abbreviated, new drugs are constantly developed, drug names can contain nearly all characters, etcetera. A large amount of research has been done on DNR and great progress has been made over the last decades (Liu *et al.*, 2015). Much of the work on DNR has been performed on longer sequences of texts such as article abstracts as seen in the work done by AbdelHaby and Dzunic (2018).

Seeking to recognize which words in a text are drug names is similar to the work of a coding specialist. The similarity in the work of DNR and a coding specialist creates the idea if DNR may be applied in a medical coding context. This would mean that DNR is applied on verbatims from concomitant medication reports to find which words in the verbatims that are drug names. Verbatims are shorter texts than the ones DNR have typically been applied on in earlier work as seen in Liu *et al.* (2015).

Verbatims are supposed to include only what drug alongside the study drug the trial participant has taken. Cases when the verbatim includes other information are however common. An example of a verbatim that is coded to WHODrug is "You'll love coles paracetamol". In this example, a coding specialist is going to say that "coles paracetamol" is a drug name as "coles paracetamol" exists in WHODrug and is most likely the drug name in the verbatim. We may however see that the verbatim contains other information as well. Examples of this are "TRIATEC/00885601/(RAMIPRIL)" which is coded to "Triatec", "VEWNLAFAXINE-ER 150 MG TEVA" which contains a misspelled drug name and is coded to "Venlafaxine Teva", and "LOCABIOSAL PRO SPR?HSTO? ZUR ANWENDUNG IN DER MUNDH?HLE" which also is misspelled and coded to "Locabiosol" in WHODrug. There are also examples where the drug name in the verbatim does not resemble the coded drug name such as in the verbatim "Argentum" which is coded to the drug "silver". Verbatims may also include several drug names in the same verbatim. "ISONIAZID, PYRAZINAMIDE, RIFAMPICIN" is an example of this and is coded to "Isoniazid;pyrazinamide;rifampicin". These verbatims give examples of when the coding process is not as easy as to make a direct comparison of drug names in WHODrug and words in the verbatims. Such an approach would also have been made difficult by the similarity between different drug names in WHODrug as well as some drug names similarity to regular English words.

It is the coding specialists' job to see which words that are drug names and what drug name they match in WHODrug. WHODrug contains over 480000 drug names and is the most comprehensive drug dictionary in the world. DNR, in contrast to the work of a medical coder, aims to find which word or words in a verbatim that is/are drug names rather than which drug in WHODrug they are. Considering the inherit problem of making a classification with over 480000 drug names and the nature of DNR, this thesis aims to investigate if and how DNR may be used to find which words in verbatims that are drug names. If successful, this work could act as a step in the work of medical coding.

In this thesis, we have access to over 240000 verbatims that have been coded to each verbatims corresponding drug name or drug names in WHODrug. As the verbatims are already coded, WHODrug is not going to be used in this thesis. For each word in a verbatim, a classification if the word is part of a drug name is going to be made.

## 1.1 Problem definition

From the introduction a problem definition has been derived which is the following.

- Can drug name recognition be used to find drug names in verbatims from concomitant medication reports?
- What kind of drug name recognition may be used to find drug names in verbatims from concomitant medication reports?

## 1.2 Thesis disposition

After the introduction where background and problem definition has been presented, section 2 presents the theories and concepts used in this thesis. Section 3 presents the data, methods, and limitations in this thesis. Section 4 presents the results that have been achieved through the work presented in section 3. The results are followed by section 5 which contains a discussion of the results. The conclusion of this thesis along with a summary is presented in section 6 which is followed by section 7 on future work that may be done in order to research the subject further.

# 2. Machine Learning

Machine learning (ML) is an area in data science which contain methods for computers to learn from data. ML if often divided in two sub-categories (Goodfellow, Bengio and Courville, 2016).

- Supervised learning - Methods where the computer is handed data to learn from, as well as labels for this data. An example of this is, if the computer is passed a picture of a cat, and an answer which says that this is a picture of a cat. From this information the computer creates rules so that if it sees a similar picture in the future, it will predict it as a cat. This is called classification which is a supervised learning method.

- Unsupervised learning - Methods where the computer is handed data without labels to it. These methods learn useful properties from the structure of the data, rather than creating rules derived from data with labels. An example of this is if the computer is handed pictures of cats and dogs. It is then up to the computer to decide how to differentiate between these pictures and categorize them. This is called clustering which is a unsupervised learning method.

Many of the ML methods used today have been around since the 90's (Mitchell, 1997) but have been increasingly utilized since then (Columbus, 2018) because of a rise in computational resources and available data that can be stored. Powerful computers of today combined with ML methods, has proven to be an effective tool in many applications (Goodfellow, Bengio and Courville, 2016).

## 2.1 Classification and binary classification

Classification is a technique to decide which class an object belongs to based on one or more variables from the data. Classes are discrete and divisible, such as blood type, species, or true or false. Classification is part of pattern recognition and is an instance of supervised learning. Classification problems that are limited to two classes, true or false, are called binary classification problems.

This thesis aims to label words in verbatims as true if the word is a drug name, or false if it is not. This is a binary classification problem. There are several different models that can be used for binary classification. What they have in common is that they are trained on labelled data. After training, the models may be used to predict labels of previously unseen data. If the model performs well, it can prove to be an efficient tool to label data.

## 2.2 Logistic regression

Logistic regression is a statistical model which uses a logistic function for binary classification. The logistic function calculates the probability of data to belong to a class of 0 or 1, where the probability $p(X)$ is a value between 0 and 1. A threshold (section 2.8.6) between 0 and 1 is chosen to make the final decision of which values classify as 0 or 1. Data with calculated probability of 0.7 with a threshold of 0.5 would be classified to belong to the class 1 (Gareth James Daniela Witten, 2013).

$$p(X) = \frac{e^{\beta_0 + \sum_{i=1}^{n} \beta_i X_i}}{1 + e^{\beta_0 + \sum_{i=1}^{n} \beta_i X_i}} \qquad (1)$$

In the equation of a logistic function seen above, $p(X)$ is the probability and it is calculated as a linear function of one or more variables of the data represented by $X_i$ with corresponding coefficients $\beta_i$ which are parameters of the model (Gareth James Daniela Witten, 2013).

## 2.3 Artificial neural networks

Artificial neural networks (ANNs) is an attempt to process information like in the nervous system. The ANN systems are built on neurons, which are mini functions with their own coefficients as seen in Figure 1. The neuron may take several variables of data $x_1 \dots x_i$ as input. Each of the variables are assigned a weight $w_1 \dots w_i$ which the variables are multiplied with. In the body of the neuron, the information is integrated usually just by adding the different signals. The information is then passed to an activation function $f$ which is a primitive function, usually a ReLU function. $f$ outputs $y$ which is the result of the activation function, given the sum of the information from the weighted variables as seen in Figure 1 (Rojas, 2013).

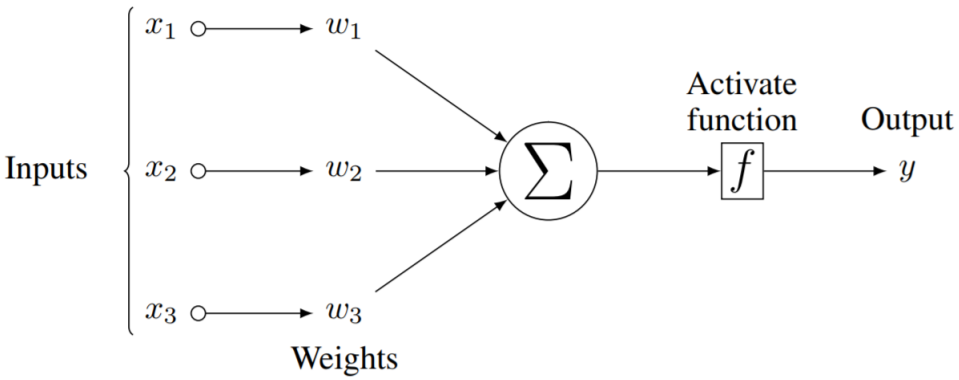


*Figure 1. A model of a single neuron.*

To form more advanced neural networks, more neurons can be used. An example of this is shown in Figure 2 where the neural network consists of several neurons. The input layer consists of 5 neurons which takes one or more variables of the data. The information from the input layer is passed to neurons in the next layer, called a hidden layer, whose output is condensed down to one single neuron in the output layer which outputs the result of the whole neural network. This architecture of neural networks is called feed-forward which means that the information is not passed in loops or backwards in the system, but is only fed forward (Rojas, 2013).

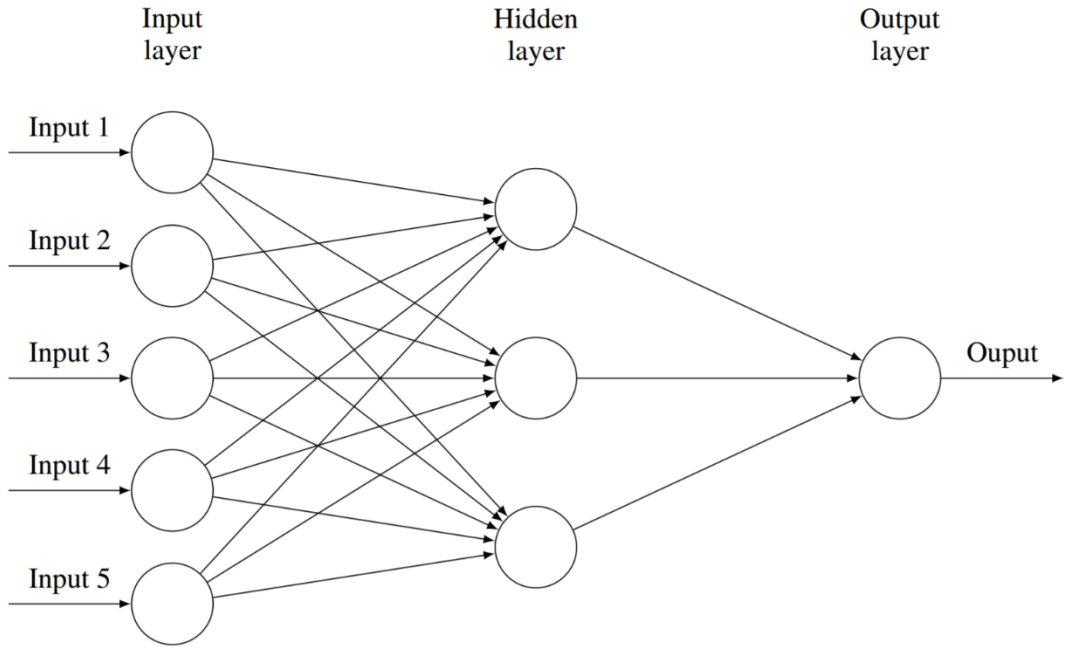


*Figure 2. A more advanced neural network consisting of several connected neurons.*

## 2.4 Recurrent neural networks

There are several variants of neural networks for different applications. Recurrent Neural Networks (RNNs) are neural networks specialized on processing sequences of data (Goodfellow, Bengio and Courville, 2016). RNNs contains cycles within the network graph which allow values in the data to be influenced by data in previous timesteps. This property allows RNNs to perform well if data values are dependent on past data. If data values should also be dependent on future data values, a bidirectional architecture can be applied to the RNN architecture (section 2.5) (Graves, 2008).

The structure of a simple RNN with no outputs can be described by the following equation:

$$h(t) = f(h(t), x(t)) \qquad (2)$$

The variable $h(t)$ represents the state at time $t$, $f$ is the function of this RNN, $x(t)$ is data at time $t$, and is what parameters the function $f$. The equation can be represented as seen in Figure 3 (left). The RNN processes information from the data variable $x$ which is incorporated into $h$ which is passed forward in time to itself. In the unfolded version (right), each neuron represents a time step.

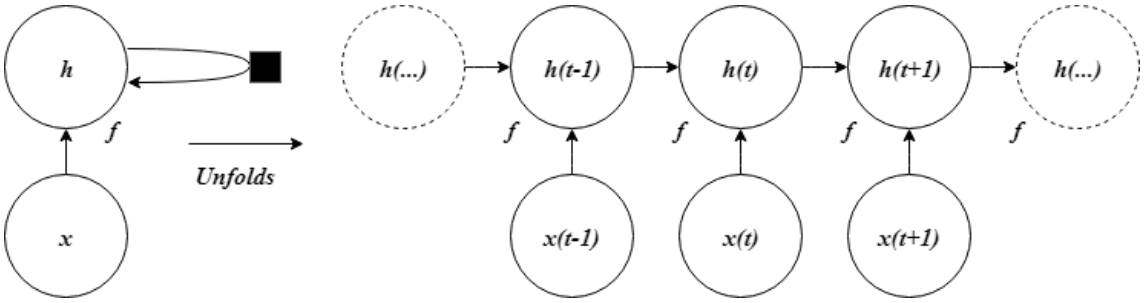


*Figure 3. Circuit and unfolded simple RNN.*

There are several variants of RNNs and three of the most common according to Goodfellow, Bengio and Courville (2016) are:

- RNNs which have an output at each time-step and recurrent connections between the hidden units.
- RNNs which have an output at each time-step and recurrent connections between the output and the hidden layer in the next time step.
- RNNs which have recurrent connections between hidden units and create only one output after reading the whole sequence.

An example of an RNN which has an output at each time-step and recurrent connections between the hidden units can be seen in Figure 4. This RNN has output $o$ and a loss function $L$ which calculates how far each $o$ is from the corresponding label $y$. This is a representation of a supervised machine learning model as discussed in section 2.1. $U$, $W$, and $V$ are weight matrixes.

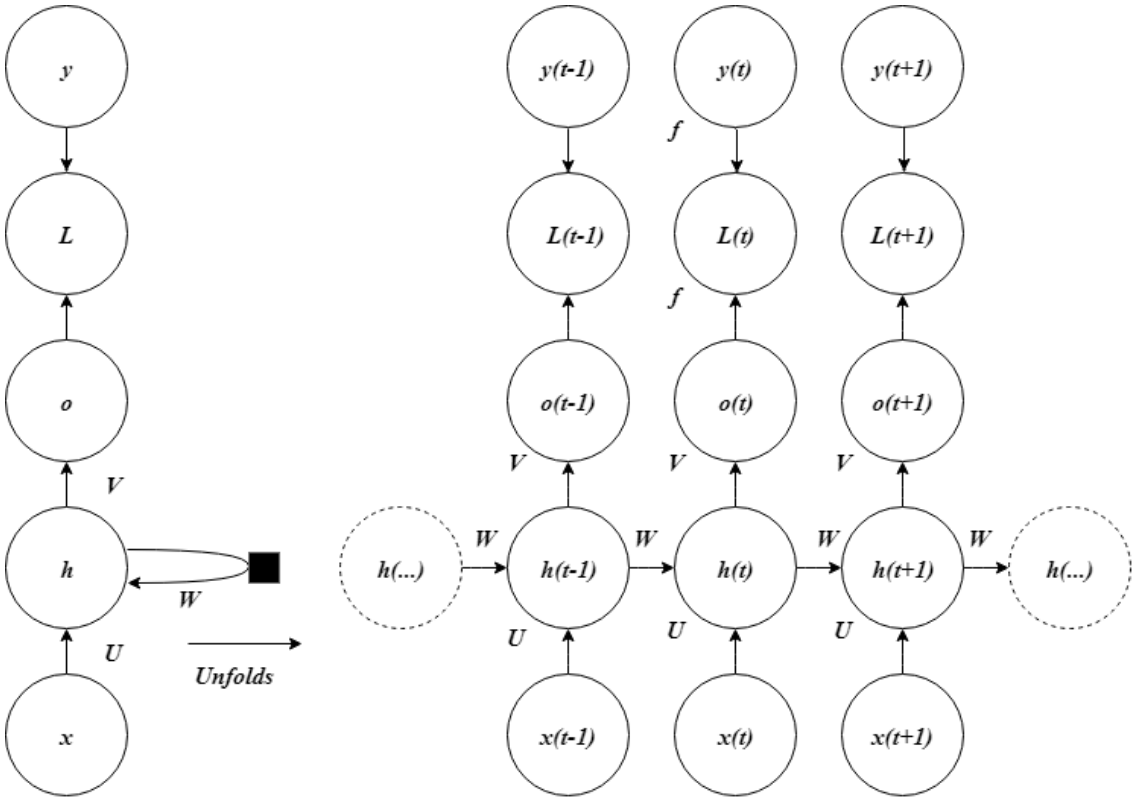


*Figure 4. An RNN with a recurrent connection between output and the hidden layer at the next timestep. x is a data variable, h the hidden layer, o the output, L the loss, and y the label. U, V, and W are weights.*

## 2.5 Bidirectional recurrent neural networks

When the whole sequence of data is known and the prediction would benefit from being influenced from future values, a bidirectional version of an RNN can be used. The bidirectional architecture combines two RNNs where one RNN is inverted to present data variables from future data. The combined standard RNN and the inverted RNN creates the bidirectional RNN whose output is predicted both from past and future data (Graves, 2008).

## 2.6 Long short-term memory

RNNs allow data variable values to be influenced by past data. Despite this, RNNs typically have problems learning long-term dependencies. Gradients propagated over many time steps tend to either vanish or explode. Even if the RNN succeeds in storing memories without the gradients exploding, the problem of the RNN assigning exponentially smaller weights to long-term interactions compared to short term ones remain. One efficient way to deal with this problem has been to introduce gates in the network instead of recurrently connected hidden layers. The gates are paths through

time that have derivatives that neither vanish or explode. The idea is that the accumulated information is used by the neurons and that the RNN learns when to clear the state of accumulated information in the gates, when not needed anymore. One of these gated RNNs is called long short-term memory (LSTM) (Goodfellow, Bengio and Courville, 2016).

The cells in the LSTM are recurrently connected to each other which can be seen in Figure 5. This replaces the recurrently connected hidden units of regular RNNs.

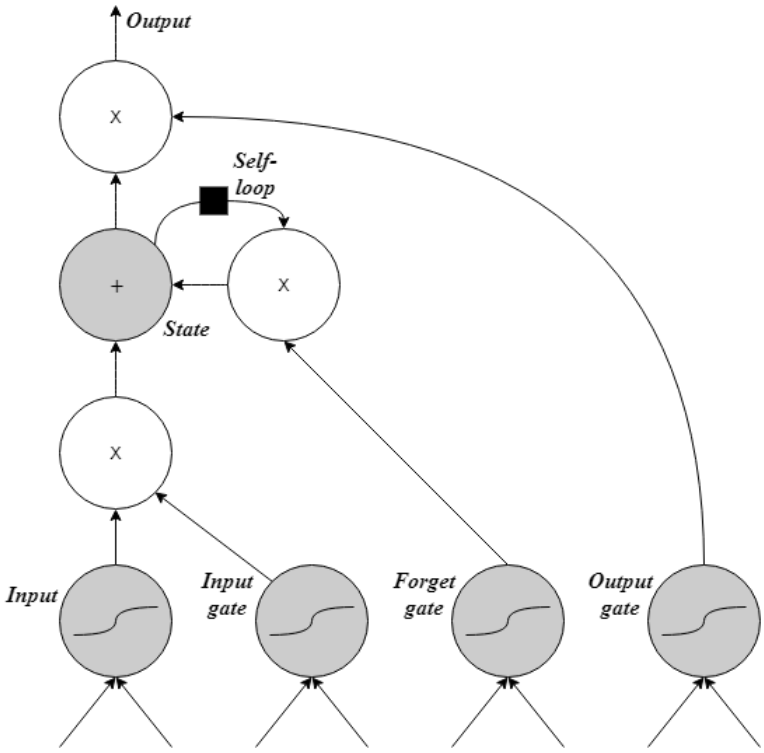


*Figure 5. The LSTM network cell*

A shallow LSTM recurrent network architecture using forward propagation according to Goodfellow, Bengio and Courville (2016) is given below.

The internal state unit is the most important component of the LSTM which has a linear self-loop as represented in Figure 5. The weight of the self-loop is controlled by the forget gate unit $f_i^{(t)}$ where $t$ representes the time step and $i$ the cell. It calculates a weight between 0 and 1 through the sigmoid unit as seen in the equation below.

$$f_i^{(t)} = \sigma(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}) \qquad (3)$$

The input vector is $x$ and $h$ is the hidden layer which contains the output from all the LSTM cells. The equation also contains the biases $b^f$, the input weights $U^f$, and the recurrent weights $W^f$ for the forget gates. Using the forget gate, the internal state is updated as seen in the equation below.

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}) \quad (4)$$

In this equation it can be seen how the biases $b$, input weights $U$, and recurrent cells $W$ are passed into the LSTM cell.
The external input gate $g_i^{(t)}$ is calculated using a sigmoid unit to calculate a weight between 0 and 1 much like the forget gate as seen in the equation below.

$$g_i^{(t)} = \sigma(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}) \quad (5)$$

The output $h_i^{(t)}$ is calculated according to the following equation:

$$h_i^{(t)} = \tanh(s_i^{(t)}) q_i^{(t)} \quad (6)$$

$$q_i^{(t)} = \sigma(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}) \quad (7)$$

Here the output gate $q_i^{(t)}$ can shut off the output $h_i^{(t)}$ as the output gate $q_i^{(t)}$ uses a sigmoid to calculate a weight between 0 and 1 much just like the external gate, but using its own parameters. If the output gate returns 0, $h_i^{(t)}$ will therefore also be 0.

## 2.7 Word embeddings and word comparison

Regular text written in text strings is a way for humans to communicate with each other. Computers are however a lot better at processing numbers than text. Much more so than humans. This means that an effective way of presenting words for a computer to handle them, is to convert them to numbers. The representation of words in numbers is called word embeddings. Word embedding is an efficient way of representing words for neural networks.

### 2.7.1 Word2Vec

Word2Vec is a unsupervised model that is used to create word embeddings and consists of a two-layer neural network. This is a model proposed by Mikolov et al (2013). It takes text as input and outputs vectors of a chosen dimension that represents these words in a shared vector space. Words with similar context are located closer to each other in vector space than words with unsimilar context. As the words are given a

relation to each other depending on their context, the following relations should be true: vec("King") + vec("Woman") - vec("Man") = vec("Queen") (Mikolov *et al.*, 2013).

The Word2vec model can use two different methods of finding word relations. One is the Continuous Bag Of Words (CBOW) which uses a window of surrounding words to predict the current word. The other is Skip-Gram, which uses the current word to predict the surrounding window of context words. CBOW is the faster of the two but Skip-Gram is better for predicting words that do not appear frequently (Mikolov *et al.*, 2013).

Several pre-trained Word2Vec word and phrase vectors exists. One of them being trained on Google News which is a comprehensive news database that contains news articles from all over the world. (Google, 2019) These pre-trained vectors contains 300-dimensional vectors for 3 million words and phrases and performs well on datasets containing English words. This gives the opportunity to use Word2vec both as pre-trained word vectors, or to train domain-specific word vectors.

### 2.7.2 Levensthein distance

When comparing word context similarity, Word2vec is an efficient way of doing this. If we are more interested in the similarity of two words from the perspective of how much they resemble each other, not in context, but in actual letters, a distance called the Levensthein distance can be a more efficient method. The Levensthein distance is a metric for comparing the difference between two sequences of text. It calculates the minimum number of how many deletions, insertions, and substitutions that is needed to change one of the words into the other. This distance, or a ratio computed from the distance, can be used to compare text similarity character-wise (Yujian and Bo, 2007).

## 2.8 Metrics for evaluation

To evaluate a model's performance there are different metrics to be used which will give different results. Metrics are the basis at which a model is evaluated to be good or bad compared to other models and is dependent on what the output is going to be used for. Metrics is therefore one of the backbones when creating and comparing models.

### 2.8.1 Confusion matrix

A confusion matrix displays how a model performs compared to test data in actual numbers. The classification is compared to the label and the result of the comparison is counted in accordance to the following list:

- True Negatives (TN) - True Negatives are items of class 0, classified as class 0
- False Negatives (FN) - False Negatives are items of class 1, classified as class 0
- True Positives (TP) - True Positives are items of class 1, classified as class 1
- False Positives (FP) - False Positives are items of class 0, classified as class 1

The confusion matrix lets us explicitly see the results of how the classifications compare to the labels as seen in Table 1.

*Table 1. Confusion matrix*

| | Classified 0 | Classified 1 |
|---|---|---|
| Label 0 | TN | FP |
| Label 1 | FN | TP |

### 2.8.2 Accuracy

Accuracy is the overall performance on true positives and true negatives, divided by the total number of classifications. If the classes are balanced and true positives and true negatives is of equal interest, accuracy is a good measure.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (8)$$

### 2.8.3 Recall

Recall is a metric on the models ability to correctly classify true positives. It does not consider false positives and a model with high recall may include a lot of false positives. This is a good measure if we want to make sure to predict all the positives and that negatives are neglectable for the model performance, as well as if false positives does not bother our evaluation.

$$Recall = \frac{TP}{FN + TP} \quad (9)$$

### 2.8.4 Precision

Precision measures the models ability to correctly classify positives. If the model classifies a lot of false positives the precision will be low. It does not take predicted negatives into account. Precision may be used when false negatives is of no interest but it is important that the classifications made as positives, are correctly classified.

$$Precision = \frac{TP}{FP + TP} \quad (10)$$

### 2.8.5 F1-score

F1-score uses both recall and precision to compute a score that is considered to be a harmonic average of the two, as it takes every category of positive classifications into its calculation. A high F1-score means that the model has few false positives and few false negatives. F1-score is a good score for data with uneven class distribution and when we are not interested in true negatives.

$$F - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \qquad (11)$$

### 2.8.6 Threshold

In binary classification, a value of 1 is classified as true and a value of 0 is classified as false. Binary classifiers predict a probability between 0 and 1 which then needs to be converted to 0 or 1 to be compared to labels. To decide if a value should be rounded to 0 or 1, a threshold needs to be introduced. A threshold of 0.5 will classify values of 0.5 or above to 1 and values under, to 0. Thresholds are problem dependent and is a parameter to be tuned. Depending on the data and model, a different threshold may be applied. One way to do this is to optimize the threshold depending on the F1-score. This is done by testing different thresholds and choosing the one which gives the highest F1-score as proposed by Lipton, Elkan and Naryanaswamy (2014).

## 2.9 Related work

Domain specific word embeddings using Word2vec together with LSTM has successfully been used in earlier applications of extracting biomedical entities. Dzunic and AbdelHaby (2018) trained a domain specific word embedding model using Word2vec on 27 million abstracts from medical publications PubMed. The word embeddings were used in combination with a labelled training data set to train a LSTM model for entity extraction. Using this approach, they reached an F1-score of 76% on finding biomedical entities in the abstracts, which was considered a success. Word embeddings using unsupervised learning has shown to improve performance for entity extraction compared to only using an RNN model as well. In a study done by Song et al. (2018) the results of an RNN algorithm gave a F1-score of 60% while applying word embeddings, a F1-score of 72% was reached. Dzunic and AbdelHaby (2018) concludes from their work, a recipe for building a custom entity extraction pipeline with the following points:

1) Get a large amount of in-domain unlabelled data.
2) Train a Word2vec model on unlabelled data.
3) Get as much labelled data as possible
4) Train an LSTM-based Neural Network

Dzunic and AbdelHaby (2018) also conclude their findings on word embeddings that they are powerful to use in the context of entity extraction as they convey word semantics, perform better than traditional features, and that no feature engineering is needed. This means that entities can be extracted with no or little domain knowledge.

More researchers have found that domain knowledge is not a necessity when dealing with biomedical entity extraction. Korkontzelos et al. (2015) studied how a model based on a manually annotated corpus of training data compared to a model trained on data that had not been manually annotated. The results showed to be similar and that drug named entity extraction can be done competitively without manually annotated data. This was done using heterogenous models built on dictionary knowledge and comparing the results to the best models trained on gold-standard data.

Related work that has been presented is similar to the work done in this thesis. They are similar in what kind of methods have been used and in what kind of entities are searched for. The entity extraction in the work of AbdelHaby and Dzunic (2018), Korkontzelos *et al.* (2015), and Song *et al.* (2018) are from longer texts giving a lot of context to the data. In this thesis a similar approach as in the work of AbdelHaby and Dzunic (2018) will be taken, but with a different data set which contains shorter strings of text rather than longer texts.

# 3. Method and data

The aim of this thesis is to investigate if and how drug name recognition can be used to find drug names in concomitant medication reports. This is going to be done by classifying which words in verbatims that are drug names. This section will explain what kind of data and methods that has been used to perform this.

A schematic figure of the method in this thesis is presented in Figure 6.

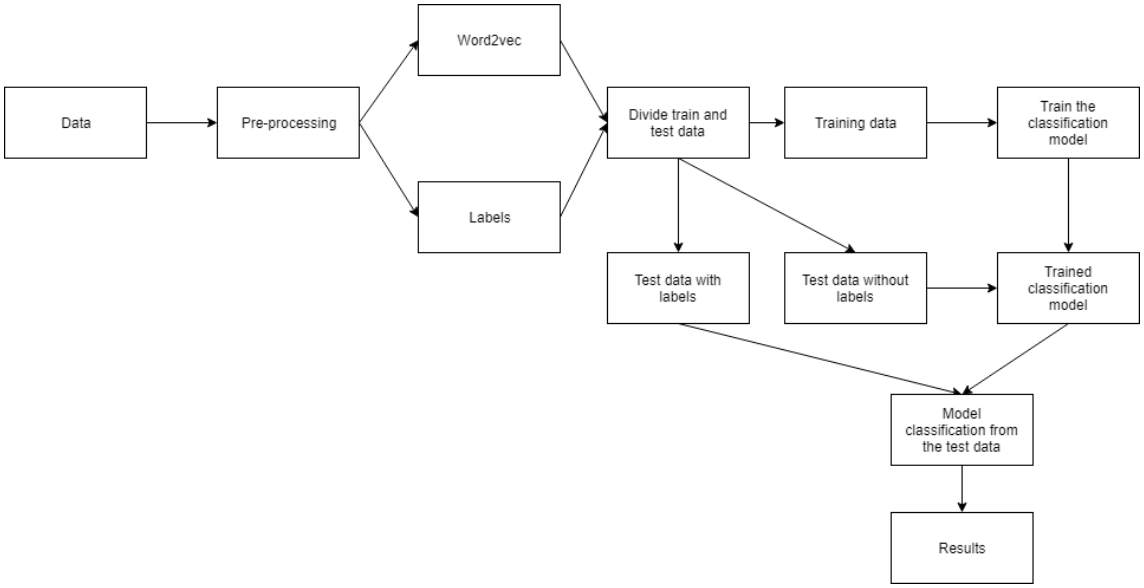


*Figure 6. Schema for the work process in this thesis.*

## 3.1 Data set

To train and test a drug name recognition model, data is needed. In this thesis, data from a UMC database containing verbatims manually coded by coding specialists at the UMC has been used. The manually coded verbatims are verbatims where a coding specialist has coded which drug name or drug names each verbatim contain. This information is found in the column coded drug name in Table 2. The data in the database are from cases when verbatims were not managed to be automatically coded by an in-house automatic drug coder. The data has been collected over several years and contains over 246000 verbatims which has successfully been coded to WHODrug by the coding specialists. Each row contains the verbatim, the coded drug name for each verbatim, and a unique identification number (ID) for each row. See Table 2 for an example of 5 different rows.

*Table 2. Raw data.*

| ID | Verbatim | Coded drug name |
|---|---|---|
| 90192 | Chemotherapy with Hycamptin1 | hycamtin |
| 168655 | (VFend) ????? | vfend |
| 211710 | Chemo iberica Omeprazole | omeprazole |
| 151132 | Metilbromuro Homatropine | homatropine methylbromide |
| 145277 | Excedrin menstrual complete express | excedrin menstrual complete |

## 3.2 Data tokenization

The data consists of complete strings of text as seen in Table 2. It contained non alphanumeric characters (characters other than A-Z, 0-9), lower- and upper-case letters, etcetera. To make the text more uniform and easier to handle in later stages, without removing context in the text, the text was processed in several stages. To shorten the length of long verbatims containing for example 15 question marks in a row, sequences of more than three of the same character in a row was shortened to three characters. To handle commas, parenthesis, and other non-alphanumeric characters in words, whitespaces were placed in front of and after all non-alphanumeric characters as well as numbers. Line breaks and tabs were converted to single whitespaces.

After pre-processing the texts, they were split on white spaces and the result was appended to a list for each verbatim and coded drug name as seen in Table 3.

*Table 3. Tokenized data*

| ID | Verbatim | Coded drug name |
|---|---|---|
| 90192 | [chemotherapy, with, hycamptin, 1] | [hycamtin] |
| 168655 | [(, vfend, ), ?, ?, ?] | [vfend] |
| 211710 | [chemo, iberica, omeprazole] | [omeprazole] |
| 151132 | [metilbromuro, homatropine] | [homatropine, methylbromide] |
| 145277 | [excedrin, menstrual, complete, express] | [Excedrin, menstrual, complete] |

## 3.3 Creation of labels

The verbatims are coded by hand to which drug the patient has taken from the WHODrug dictionary. In our case we are interested in which word in the verbatim that is the drug name. Because of this, we need to create a new column with a field connected to each row that contain a labelling system.

To create the labels, an array was created with the same length as the list of tokens in verbatim for each row. Labels of ones were placed at the index where the drug name were placed in the verbatim and zeroes in the other indices. To decide on which index to put the ones, fuzzy matching (Fuzzywuzzy, 2019), which uses the Levensthein distance between words as seen in section 2.8.2, was used to compare the similarity of words in coded drug name and verbatim. If the words were similar enough, a one was placed at the index of the drug name in the corresponding label array. An example how this would look like is seen in Table 3.

*Table 3. Labeled data*

| ID | Verbatim | Coded drug name | Label |
|---|---|---|---|
| 90192 | [chemotherapy, with, hycamptin, 1] | [hycamtin] | [0, 0, 1, 0] |
| 168655 | [(, vfend, ), ?, ?, ?] | [vfend] | [0, 1, 0, 0, 0, 0] |
| 211710 | [chemo, iberica, omeprazole] | [omeprazole] | [0, 0, 1] |
| 151132 | [metilbromuro, homatropine] | [homatropine, methylbromide] | [1, 1] |
| 145277 | [excedrin, menstrual, complete, express] | [Excedrin, menstrual, complete] | [1, 1, 1, 0] |

Different thresholds for Levenstein distance ratio were tested before choosing a ratio of 80%. The words in coded drug name and verbatim had to be at least a 80% match for a word in Verbatim to be labelled as one in the label. The evaluation of the Levensthein distance was done by looking at the examples and deciding on where the matching was good enough without being too generous and mislabelling words. The Fuzzywuzzy (Fuzzywuzzy, 2019) package was used to calculate the Levensthein distance ratio.

As used in Ribeiro *et al.* (2018), as a complement to the main data set, a small set of golden standard data was created to be used in model evaluation. To create this golden standard data, 320 random rows were extracted from the main data set. The 320 rows were then manually labelled by three coding specialists at UMC instead of being labelled by the fuzzy matching.

## 3.4 Feature extraction from text

To extract textual features of the verbatims and to make it easier for the models later implemented to process them, Word2vec (section 2.7.1) was used to create vector representations of all tokens. Each tokenized verbatim was fed into the Word2vec algorithm using Skip-Gram to be vectorised into a vector of dimension 100, thus creating a domain specific Word2vec model from this data. Using the domain specific Word2vec model, a representation of each verbatim was created using each tokens vector representation instead of the token. This information was stored in a new column to be passed into the models at a later stage. This method has proven to be effective in previous work such as AbdelHaby and Dzunic (2018) who claim that part of the recipe of successful entity extraction is to use a domain specific Word2vec (section 2.9).

## 3.5 Classification models

This thesis handles the binary classification problem of classifying words in a sequence of either being a drug name or not (section 2.1). In binary terms, this may be translated to words being classified as true (label 1) if it is a drug name, false (label 0) if it is not. To solve this classification problem, two different classification models are going to be implemented.

### 3.5.1 Logistic regression

A simple and popular classifier is the logistic regression model as seen in section 2.2. It is easy to implement for binary classification and this made it an interesting model to test in this thesis. It is used to give a baseline.

A logistic regression model takes one input at a time and does not have a function for remembering past or future words. To make the classifications from the logistic regression model more context dependent, windows of two different sizes were created to be passed to the model. Windows of size one was passed to the model to test it in its basic form. Windows of size three were also tested to give the model more context. An example would be the verbatim [metilbromuro, homatropine] that would be translated into the following windows, [<B> , metilbromuro, homatropine], [metilbromuro, homatropine, <E>], where the <B> and <E> tags were added to mark beginning and end of verbatim. This also meant that the label lists were padded with a 0 at the beginning and the end for windows of size three and that corresponding windows were created from the labels as well. The verbatim windows were translated into Word2vec vector representations (section 3.4) of the windows. The Word2vec vectors, each of dimension 100, were merged into one vector with dimension 300 to represent the three words in each window of size three. In our implementation this gave over 1160000 windows of size three.

The windows were fed into the logistic regression model together with corresponding labels that labelled the window as a drug name if the middle word in the window was a

drug name. The model gave an output of a probability between 0 and 1 and a threshold of 0.65 for window size one and 0.68 for window size three for predicting true. These thresholds were chosen through F1-score maximizing of the thresholds as seen in section 2.8.5 and 2.8.6.

As there were class imbalances where the label 0 occurred four more times than 1, class weights were fed into the model together with the data where weights for 0 was one and weights for 1 was set to four.

The activation function for the model in Keras (section 3.6) was sigmoid as this is the standard S-curve for logistic regression. Adam was used as optimizer with a learning rate of 0.001. Binary cross entropy was used as loss function

### 3.5.2 Bi-LSTM

Earlier work such as AbdelHaby and Dzunic (2018) and Korkontzelos *et al.* (2015) has implemented neural networks for entity extraction (section 2.9). To try if neural networks (section 2.3) could be useful in the application of this thesis, a neural network was implemented. As our input was sequences of data, an RNN (section 2.4) was chosen as it takes sequences as input and is able to transfer information from earlier timesteps when predicting a word. For the RNN to better catch the context of the whole sequence and to avoid the inherit problem of exploding or vanishing gradients in RNNs, a LSTM type of RNN is chosen as it is better at passing information through the sequence (section 2.6).

In our data we had access to the whole sequence of data, both past and future words, when classifying words. This led us into applying a bidirectional function to our LSTM model. This lets the LSTM predict each word using data from the past and future in the sequence (section 2.5). This gave us a bidirectional LSTM model.

In order for us to use batch-learning, where a batch is a number of samples and batch-learning in when the Bi-LSTM models variables gets updated in training, in our implementation, the sequences fed to the Bi-LSTM needs to be of equal length. To make the sequences of equal length, the length of the longest verbatim was measured and all the vector representations of verbatims and labels were padded with zeroes in the beginning of the sequences to match this length. This created a class imbalance in the labels which was compensated for when the model was fitted by adding class weights according to the class imbalances.

The output of the model was probabilities between 0 and 1 for each word and a threshold for predicting 1 was chosen by which threshold gave the maximum F1-score on the training data (section 2.8.5, 2.8.6). This maximum F1-score was reached with a threshold of 0.17.

The Bi-LSTM model implemented in Keras (section 3.6) used an embedding layer which contains numbers linked to Word2vec vector representations of the verbatims.

The Word2vec vectors are together with corresponding labels the input for the model. The implementation also used a batch-size of 128, a dropout of 0.5, a binary cross entropy loss function and the adam optimizer with a learning rate of 0.001.

## 3.6 Tools

To read and process the large amount of data, an efficient tool has been needed. Pandas (Pandas, 2019) and Numpy (NumPy, 2019) are popular libraries in Python (Python, 2019) used for data processing. These tools were used to fulfil the requirement of efficient and easy data processing.

To be able to implement the models used, Keras (Keras, 2019) running on top of Tensorflow (Tensorflow, 2019) has been used. Keras is a high-level neural networks API and Tensorflow is an open source machine learning library. SciKit-learn (SciKit-learn, 2019) has been used to evaluate the models.

To implement the word embeddings, a Word2vec implementation through gensim (Rehurek, 2019) was used. The training and testing of the models was run in GPU mode for faster processing.

## 3.7 Evaluating classifiers

The logistic regression classifier was implemented to give a baseline on performance with a simpler classifier. The baseline of the logistic regression model was used to compare it to a more advanced bidirectional LSTM model. As the data showed large imbalances before but even more so after padding, F1-score was chosen as the measurement to compare models. As the target of this thesis has been to label drug names rather than to label non-drug names, the F1-score is considered a well performing measurement for such a task as it gives a measure on how well the models predict ones rather than zeroes and deals with class imbalances (section 2.8.5). In addition to F1-score, precision has been measured.

To test the models, a test dataset was created by randomly taking 10% of the rows from the processed data set (section 3.3) before training the models. The models were then evaluated against each other using this test data. To evaluate the best performing model, a gold standard data, manually annotated by specialists was used (section 3.3). This gave a measurement on how well the model performed on perfect data. The labels created using the Levensthein distance (section 3.3) was also tested on the golden standard data to give a measurement how well the Levensthein distance labels were created.

## 3.8 Limitations

The work has been limited in time and on available data. The data that was found for the application in this thesis, was coded with coded drug name. It was not labelled with

indexes of where in the verbatims the drug names were. Having to label the data automatically has limited the quality of the labels and therefore also the results of the models. A limitation was also set that the data is classified in true or false rather than classifying which drug in WHODrug each word is. This was done because a classification problem of over 480000 classes, which represents how many drugs are represented in WHODrug, would be a hard to solve classification problem which would not fit in this thesis.

The work has also been limited to testing two classifiers that was thought to give the best results. If more time was given, more classifiers could have been tested.

## 3.9 Place for work

The work was done at Uppsala Monitoring Centre (UMC) in Uppsala, Sweden. UMC has provided data, office space, guidance and a computer.

# 4. Results

The results in this thesis are classifications produced from a logistic regression model as seen in section 3.5.1, and a bidirectional long short-term memory model (Bi-LSTM) as seen in section 3.5.2. The models have been trained on Word2vec vector representations of verbatims with corresponding labels of where in the verbatim a drug name exists (section 3.6). Verbatims are short texts on which concomitant drug a trial participant has taken and comes from concomitant medication reports (section 1). The labels have been created using the Levensthein distance ratio between words in verbatims and words in coded drug name (section 3.3). The coded drug names are manually coded by coding specialists of which drug names are present in each verbatim.

The logistic regression model (section 3.5.1) and the Bi-LSTM model (section 3.5.2) that are implemented are evaluated using an F1-score. F1-score is a metric that performs well on unbalanced classes and gives a number on how the model performs on classifying true labels (section 3.8).

The models have been trained on 90% of the data and tested on 10% of the data (section 3.3). The best performing model is then tested on a small gold standard data set (section 3.1). The thresholds for predicting true or false are chosen through which threshold optimizes F1-score for the different models (section 3.8).

The logistic regression model was trained on 1000 epochs and used a sigmoid activation function, an adam optimizer with a learning rate of 0.001, and a binary cross entropy loss function (section 3.5.1). The Bi-LSTM model was trained on 100 epochs and used a Word2vec embedding layer (section 3.5.2), a bidirectional LSTM layer, a dropout of 0.5, a time distributed dense layer with outputs for each sequence with a sigmoid activation function. The model was compiled with a binary cross entropy loss function and an adam optimizer with a learning rate of 0.001.

Results from the logistic regression and Bi-LSTM model performance on test data is seen in Table 4.

*Table 4. Performance on test data.*

| Model | Threshold | F1-score | Accuracy |
|---|---|---|---|
| Bi-LSTM | > 0.18 | 82.7% | 99.4% |
| Logistic Regression window size three | > 0.68 | 70.7% | 76.0% |
| Logistic Regression windows size one | > 0.65 | 66.2% | 80.5% |

The Bi-LSTM model gives an F1-score of 82.7% on test data with a threshold of 0.18 whereas the logistic regression model with window size three produces an F1-score of 70.7% with a threshold of 0.68. The logistic regression model fed with windows of size one produced the lowest F1-score of 66.2% and a threshold of 0.65. The Bi-LSTM model is the model that performs best with the highest F1-score and this model is therefore evaluated further.

Eight representative examples as seen in a larger portion of the data of when the classification from the Bi-LSTM model matches the labels is shown in Table 5. The table contains the verbatims and the labels that the model is tested on. For evaluation purposes, the table also contains the model output classification, and the coded drug name for each corresponding row. Note that the model is not trained using the coded drug name. The label and model classification are compared in the column match, which contains information if label and classification matches 100% or not. Note that the models are trained on Word2vec representations of the verbatims and not the verbatims themselves.

*Table 5. Examples on cases when the Bi-LSTM classification matches the label. The verbatim ['bascupan'] is classified by the Bi-LSTM model as [1] wich is a correct classification as it matches with the label [1].*

| Coded drug name | Verbatim | Label | Classification | Match |
|---|---|---|---|---|
| ['buscopan'] | ['bascupan'] | [1] | [1] | Yes |
| ['dextseran', 'r'] | ['331955403'] | [0] | [0] | Yes |
| ['ropivacaine'] | ['ropivacaine', 'strength', 'do', 'not', 'know'] | [1 0 0 0 0] | [1 0 0 0 0] | Yes |
| ['cyanocobalamin', '; ', 'pyridoxine', ';', 'thiamine'] | ['cyanocobalamin', '/', 'pyridoxine', '/', 'thiamine'] | [1 0 1 0 1] | [1 0 1 0 1] | Yes |
| ['ometasone', 'furoate'] | ['mometasone', 'furoate', '(', 'mometasone', 'furoate', ')', '(', 'nasal'] | [1 1 0 1 1 0 0 0] | [1 1 0 1 1 0 0 0] | Yes |
| ['casanthranol', ';', 'docusate', 'sodium'] | ['casanthranol', 'and', 'docusate', 'sodium'] | [1 0 1 1] | [1 0 1 1] | Yes |
| ['fentanyl'] | ['sandoz', 'fentanyl', 'transdremal', 'system'] | [0 1 0 0] | [0 1 0 0] | Yes |

| | | | | |
|---|---|---|---|---|
| ['pyralvex'] | ['pyralvex', '(', 'salicylzuur', '/', 'rheum', ')', 'aanstipvloeistof', ',', '(', 'millig'] | [1 0 0 0 0 0 0 0 0 0] | [1 0 0 0 0 0 0 0 0 0] | Yes |

Table 6 contains typical examples as seen in a larger portion of the data of when the Bi-LSTM classification and the labels do not match. The coded drug name ['good', 'sense', 'dual', 'action', 'complete'] shows how regular English words can be used in a drug name.

*Table 6. Examples on cases when the Bi-LSTM classification did not match the label.*

| Coded drug name | Verbatim | Label | Classification | Match |
|---|---|---|---|---|
| ['good', 'sense', 'dual', 'action', 'complete'] | ['good', 'sense', 'complete'] | [1 1 1] | [0 0 0] | No |
| ['pemetrexed'] | ['培美曲塞'] | [0] | [1] | No |
| ['humulus', 'lupulus'] | ['houblon'] | [0] | [1] | No |
| ['hmg', 'coa', 'reductase', 'inhibitors'] | ['stating', '(', 'hmg', 'coa', 'reductase', 'inhibitors', ')'] | [0 0 1 1 1 1 0] | [0 0 1 1 1 0 0] | No |
| ['cough', 'and', 'cold', 'preparations'] | ['dayquil', 'fox', '(', 'dextromethorphan', 'hydrobromied', ',', 'pseudoephedrine', 'hydrochlorid'] | [0 0 0 0 0 0 0 0] | [0 0 0 1 0 0 1 1] | No |

The Bi-LSTM model was tested on the golden standard data (section 3.3) and its performance is shown in Table 7.

*Table 7. Performance on golden standard data*

| Model | Threshold | F1-score | Accuracy |
|---|---|---|---|
| Bi-LSTM | > 0.18 | 46.4% | 97.9% |

To test how the labels created using the Levensthein distance compared to the manually labeled golden standard data (section 3.7), an F1-score of how the Levensthein labels performed on the golden standard data was calculated as seen in Table 8.

*Table 8. Levensthein labels performance on golden standard labels*

| Label | F1-score |
|---|---|
| Levensthein | 86.5% |

# 5. Discussion

The F1-scores of the logistic regression model and the Bi-LSTM model are shown in Table 4. We can see that the Bi-LSTM model performs better as there is a difference of over 10% between the logistic regression model with a window size of three and the Bi-LSTM model. The difference is not immense which shows that the logistic regression model can perform well with its relatively simple structure, compared to Bi-LSTM. The logistic regression model fed with window size three performs better than the logistic regression model fed with window size one. This shows that a model with more context performs better than a model with less context. This may also be why the Bi-LSTM performs the best as this model is built to take context into consideration. Bi-LSTM is not limited by a window size in what context it considers for its classifications.

The logistic regression is a linear model. Its relatively good performance shows that our data in many cases are linearly classifiable. However, the F1-score shows that some cases our data is better classified using a non-linear and more flexible model such as Bi-LSTM, which leads to better results. The thresholds between the models shows a big difference in which threshold maximizes the F1-score for logistic regression and Bi-LSTM. This may be connected to that the Bi-LSTM model is padded which creates a larger class imbalance. This may create lower classification values for words that are drug names which may have resulted in that a lower threshold is favoured.

In Table 5 and 6, the four verbatim examples ['bascupan'], ['houblon'], ['培美曲塞'], and ['331955403'] are shown. They are chosen from a larger portion of tested data as typical examples of how the Bi-LSTM model classifies when the verbatim contains one word. When studying the tested data, we can see that for verbatims containing one word, it will be classified as true unless the verbatim only consists of a number, which will be classified as false. This is seen throughout the data and shows that the model sometimes makes a better classification which words are drug names than the Levensthein distance, used for labelling, does. When studying the verbatims and comparing them to the coded drug name in the three true-classified verbatims, the prediction should indeed be true, which the Bi-LSTM model classifies, rather than false, which it has been labelled as. This gives an indication that the model may handle non-English text better than the labelling process. For further discussion on label reliability, see section 5.3. When the model is confronted by numbers, the numbers are classified as false. This is the same behaviour as when the Levensthein distance labels, as numbers in verbatims do not resemble drug names in coded drug names.

In ['good', 'sense', 'complete'] all words are classified as false, in contrast to the label where all words are labelled as true. It is seen throughout the test data that the model performs poorly on labelling drug names that contains words that are also regular English words. This may be because regular English words are not common as drug names in the data. The model therefore has learnt that these words are usually not drug names. This behaviour makes the model ill-suited for some applications of drug name

recognition. However, it shows better performance on drug names which look like typical drug names to a human. Considering that a human may also predict regular English words as false, the model could be performing as good as a human without any access to the WHODrug dictionary would.

Longer verbatims when the model performs well are for example ['azijnzuur', '/', 'triamcinolonacetonide', '(', 'mg', 'per'], ['ropivacaine', 'strength', 'do', 'not', 'know'], and ['cyanocobalamin', '/', 'pyridoxine', '/', 'thiamine'] which gives examples of when the model classifies 100% correctly on verbatims containing drug names and regular English words. These are also examples of when the verbatims are longer and therefore gives context the words. As Bi-LSTM can handle context, it could be that Bi-LSTM therefore performs well on longer verbatims as well. There are however also examples as seen in Table 6 where it performs less well on longer verbatims.

## 5.1 Model performance on golden standard data

The F1-score of the Bi-LSTM model in Table 7 was 46.4% with an accuracy of 97.8% on the golden standard test data. The high accuracy is, as in the other test data, not an appropriate measure as it does not deal with the imbalanced classes that are seen in the padded sequences of the data, that is fed to the Bi-LSTM model. The F1-score of 46.4% is almost half of that of the F1-score on the test data. This result shows that the Bi-LSTM performs poorly on the golden standard data.

In the process of manually labelling the golden standard data, the terminology specialists who labelled the data, reported that they were adding a substantial number of true labels. This was measured as an F1-score of how well the Levensthein labels compared to the golden standard ones. The F1-score was 86.5% as seen in Table 8. The difference between the Levensthein labels and the golden standard labels may be because words in verbatims sometimes are abbreviations or otherwise hard to map using a Levensthein distance as seen in section 3.

The difference in the two labels may be why Bi-LSTM performs poorly on the golden standard data. It is therefore not robust in the sense that it does not perform well on other kind of data than it was trained on. As the golden standard resembles the training data in other ways, it is possible that if the model was to be trained exclusively on golden standard data, it would perform well on this kind of data as well.

## 5.2 Data set and pre-processing

As drugs in WHODrug can contain almost all characters, it is not thought to be a good idea to remove characters from verbatims as this potentially could remove important information for certain drugs. To be able to split data efficiently for cases when special characters were present without removing them, a compromise was to add whitespaces to the beginning and the end of the special characters. This retained context that could be perceived in the models as well as helping to split cases where two drugs were

connected, such as ”paracetamol;orifarm”. Characters that occurred more than three times in a row were however shortened to length of three. This retains the information of many characters, without weighing down the model with extra-long verbatims without rational information. In addition to this, the verbatims were made to lower case letters and split on whitespaces to create single words.

A more aggressive split on the data could have been implemented which would have removed all characters other than a-z. This would have drastically reduced the amount of characters in the verbatim. A test was made on this and the loss of information did affect he quality of the data to a substandard level which is why it was not used.

## 5.3 Label reliability

The labels in the training data were created using a fuzzy matching with a ratio of the Levensthein distance. As seen in Table 5, ['houblon'] is a verbatim that is coded to ['humulus', 'lupulus'] in WHODrug. The verbatim is labelled as 0 while the classification is 1. In this case, the classification could be argued to be correct even though the label is not 1. Typically seen as a trend in the classified data, in each verbatim containing one word, this word is classified as 1. This has in many cases, as in the case of ['houblon'], shown to be a correct classification as ['houblon'] represents the drug ['humulus', 'lupulus'] in WHODrug. This, together with the statements made from the terminology specialists, highlights that fuzzy matching with Levensthein distance in many cases gives the wrong label to words in verbatim.

The labelling method did not handle cases where the language in verbatim was not English and therefore differed from the language in coded drug name. One such verbatim is ['培美曲塞'] which is Chinese for the drug ['pemetrexed'] which it was coded as. Such cases are impossible for the Levenshtein distance to predict as it takes a complete deletion and writeback of the verbatim. This would take a Levensthein matching ratio of 0%, which would label all words in verbatim as 1. This is also true for cases such as ['331955403'] which has a coded drug name which it shares no similarity within terms of characters.

## 5.4 Comparison to related work

The closest comparison in related work that has been done may be seen in AbdelHaby and Dzunic (2018) who used domain specific word embeddings with Word2vec in combination with a labelled training data set to train a LSTM model for entity extraction (section 2.9). They reached an F1-score of 76% on finding biomedical entities. In comparison with the binary classifier in this thesis, their classifier handled several classes. Their work included a data set of labelled data which they did not create automatically in the scope of their work. The reliability of their labels is therefore thought to be higher than in this thesis. The F1-score reached with the Bi-LSTM model in this thesis is however higher which may have to do with the bidirectional architecture

applied. The work of AbdelHaby and Dzunic (2018) could potentially benefit from implementing a bidirectional architecture to their work.

# 6. Conclusion and summary

This thesis aimed to investigate if, and in that case, what kind of drug name recognition could be used to find drug names in verbatims from concomitant medication reports. It has been proven that it is possible to implement drug name recognition in the form of a machine learning model. The model produces positive results on classifying which words in the short text of a verbatim that are drug names. This has been done by implementing a bidirectional long short-term memory recurrent neural network model, which has been trained on Word2vec representations of verbatims with corresponding labels of wherever in the verbatim drug names exists.

The model that has been implemented shows good performance with an F1-score of 82.7% on the kind of data it has been trained on. The model shows poorer results on manually annotated data which it has not been trained on. The main improvement point of this thesis is to create more reliable labels for the data the model trains on.

# 7. Future work

In this thesis, the main improving point is the quality of the labels. In future work, efforts should be made on making all the labels of gold standard quality. A better label quality in combination with the Bi-LSTM model implemented in this thesis could have the potential of improving tools of automatic drug coding in the future, as well as the daily work of medical coders.

In the future, the word embeddings may benefit from being trained on a larger data set, such as the WHODrug dictionary and potentially concomitant drug case reports which are longer reports on concomitant medication. This approach may increase the reliability of relations between drug names and other regular English words. Such an approach might however decrease the domain specific aspect sought after as discussed in section 2.7 when the word embedding is trained on only verbatims.

Fasttext (Fasttext, 2019) was initially the word embedding method and implemented for this thesis. Due to technical problems which were not solved before the end of the thesis work, Word2vec was implemented instead as a substitute. Future work should include implementing and testing the Bi-LSTM model using Fasttext or similar word embeddings, which handle previously unseen words better than Word2vec. This could make the model more robust.

In this thesis some optimization of hyperparameters has been done. To make the Bi-LSTM model better in the future, a more extensive testing may be done on hyperparameter optimization for the Bi-LSTM model.

# Bibliography

AbdelHaby, M. and Dzunic, Z. (2018) Deep Learning for Domain-Specific Entity Extraction from Unstructured Text. Available at: https://databricks.com/session/deep-learning-for-domain-specific-entity-extraction-from-unstructured-text (Accessed: 3 June 2019)

Columbus, L. (2018) Roundup Of Machine Learning Forecasts And Market Estimates, 2018. Available at: https://www.forbes.com/sites/louiscolumbus/2018/02/18/roundup-of-machine-learning-forecasts-and-market-estimates-2018/#7e5fae5f2225 (Accessed: 19 May 2019).

Fasttext (2019). Available at: https://fasttext.cc/ (Accessed: 26 June 2019).

Fuzzywuzzy (2019) Available at: https://github.com/seatgeek/fuzzywuzzy (Accessed: 3 June 2019).

Gareth James Daniela Witten, T. H. R. T. (2013) An Introduction to Statistical Learning. Springer-Verlag New York.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) Deep Learning. MIT Press.

Google (2019) Google Code Archive - Long-term storage for Google Code Project Hosting. Available at: https://code.google.com/archive/p/word2vec/ (Accessed: 19 May 2019).

Graves, A. (2008) Supervised sequence labelling with recurrent neural networks. Technical University of Munich. Available at: https://www.cs.toronto.edu/~graves/phd.pdf (Accessed: 3 June 2019)

Keras (2019). Available at: https://keras.io/ (Accessed: 3 June 2019).

Korkontzelos, I. et al. (2015) Boosting drug named entity recognition using an aggregate classifier, Artificial Intelligence in Medicine, 65(2), pp. 145–153.

Lipton, Z. C., Elkan, C. and Naryanaswamy, B. (2014) Thresholding Classifiers to Maximize F1 Score. Available at: https://arxiv.org/pdf/1402.1892.pdf (Accessed: 19 May 2019).

Liu, S. et al. (2015) Drug Name Recognition: Approaches and Resources, Information, 6, pp. 790–810.

Mikolov, T. et al. (2013) Efficient Estimation of Word Representations in Vector Space. Available at: http://ronan.collobert.com/senna/ (Accessed: 19 May 2019).

Mitchell, T. M. (1997) Machine Learning. McGraw-Hill. Available at: http://profsite.um.ac.ir/~monsefi/machine-learning/pdf/Machine-Learning-Tom-Mitchell.pdf (Accessed: 19 May 2019).

NumPy (2019). Available at: https://www.numpy.org/ (Accessed: 3 June 2019).

Pandas (2019). Available at: https://pandas.pydata.org/ (Accessed: 3 June 2019).

Python (2019). Available at: https://www.python.org/ (Accessed: 3 June 2019).

Rehurek, R. (2019). Available at: https://radimrehurek.com/gensim/ (Accessed: 3 June 2019).

Ribeiro, A. H. et al. (2018) Automatic Diagnosis of Short-Duration 12-Lead ECG using a Deep Convolutional Network, arXiv e-prints, p. arXiv:1811.12194.

Rojas, R. (2013) Neural networks: a systematic introduction. Springer International Publishing AG.

SciKit-learn (2019). Available at: https://scikit-learn.org/stable/ (Accessed: 3 June 2019).

Song, H.-J. et al. (2018) Comparison of named entity recognition methodologies in biomedical documents, BioMedical Engineering OnLine, 17(Suppl 2), p. 158.

Tensorflow (2019). Available at: https://www.tensorflow.org/ (Accessed: 3 June 2019).

UMC (2019) UMC | WHODrug Global. Available at: https://www.who-umc.org/whodrug/whodrug-portfolio/whodrug-global/ (Accessed: 31 May 2019).

Yujian, L. and Bo, L. (2007) A normalized Levenshtein distance metric, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(6), pp. 1091–1095.