KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Learning a Diseases Embedding using Generalized Word2Vec Approaches.

Milan van der Meer

Thesis submitted for the degree of
Master of Science in Engineering:
Computer Science, specialisation
Artificial Intelligence

Thesis supervisor:
Prof. dr. R. Wuyts

Assessors:
Prof. dr. ir. H. Blockeel
R. van Lon

Mentor:
Dr. E. D'Hondt

Academic year 2015 – 2016

Preface

I start with expressing my gratitude towards Roel for giving me the opportunity to work on a truly interesting subject. I also appreciate that you were not just an invisible promoter, but a promoter who showed a true interest in the subject and curiosity for the results of the research.
I also know that I'm a lucky student who fell with his butt in the butter as Ellie took the time and effort to proofread my thesis a couple of times. Together with Roel, you really provided me with a lot of support and made it possible to finish my thesis in a way I'm proud of. Thank you.
On a more personal note, I want to thank the people who stood by my side. My parents who gave me the opportunities in life and also prepared me for those opportunities. My friends, the CW kneusjes, my kotgenoten, and of course Siemen, the guy who somehow still manages to stay around. I also want to thank my future wife, just to cover all bases.

Milan out.

Milan van der Meer

Contents

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

List of Figures and Tables

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

EHR	Electronic Health Record
ICD	International Classification of Diseases
WHO	World Health Organization
MedDRA	Medical Dictionary for Regulatory Activities
THIN	The Health Improvement Network
CBOW	Continuous Bag-of-Words
knn	k-Nearest Neighbors
kd	k-dimensional
MLP	Multilayer Perceptrons
RNN	Recurrent Neural Network
LSTM	Long-Short Term Memory
DL4J	DeepLearning for Java
MSLR	Thomson Reuters MarketScan Lab Database
OMOP	Observational Medical Outcomes Partnership
OSIM2	Observational Medical Dataset Simulator Generation 2

Chapter 1

Introduction to Electronic Health Record Analytics

1.1 Introduction

In section 1.2 we explain what electronic health records are and how these are represented. In section 1.3, we explain how the electronic health records can be used to retrieve useful medical information. We also explain different approaches to retrieve this information from the health records. Those approaches range from querying databases to more advanced machine learning techniques. After stating the general context we are working in and what we want to improve, we state the outline of information in the thesis in section 1.4.

1.2 Electronic Health Records

An electronic health record (EHR) is a collection of time-stamped data about a patient over a period point of time. It is stored digitally and thus can be established for a large number of patients over a long time period.
The data stored in an EHR provides an overview of the patients' health information. Health information like demographics, medical history, diagnoses, medications, and such, are stored [2].

Recently large countries like the US and the UK, are each investing more than 20 billion dollars into EHR systems [55]. Those systems are adopted by around 70% of the physicians in the US, but of those systems a lot of different implementation are used. Which means a large number of physicians are using different methods or systems. This causes different ways of information representation and makes it harder to compare EHRs across different systems. We focus on disease codes in the next section and introduce two standards: one used by mainly insurance companies and the other used by pharmaceutical companies.

1.2.1 Disease Codes

To make EHRs practical it is important to adhere to standards for data formatting. A well-documented and consistently used standard makes it easy to store and extract information from large-scale databases of EHRs. Without the possibility of extracting information, an EHR becomes a simple digital version of medical records on paper. Part of an EHR consists of the diagnosis of the patient. It provides information about his disease trajectory and allows analysis on his health situation. With a uniform system for classifying diseases nationwide, it is possible to provide a general picture on health situations of populations. Several have been defined and their usage is scattered. We discuss the 2 most relevant to our work.

ICD-10

The International Statistical Classification of Diseases and Related Health Problems (ICD) is a medical classification list made by the World Health Organization (WHO) [7]. Several variants are available like ICD-9, ICD-10, ICD-10 CM, and others. The ICD-10 contains more than $14,400$ codes about diseases, disorders, injuries, and other related health conditions. For example, the code for a sprained ankle is $S93.4$. It also provides hierarchical categories for those codes to allow a more general overview of diseases. ICD is mainly used by insurance companies.

MedDRA

The Medical Dictionary for Regulatory Activities (MedDRA) provides medical terminology in the form of disease codes [3]. A MedDRA code is an eight digit numeric code where the code itself has no meaning. The code functions as an unique identifier so new terms are just identified with the next available code. Note that this system does not provide a clear and easy to use system to retrieve a hierarchical structure of a code. MedDRA is mainly used by pharmaceutical companies.

1.3 EHR Analytics

EHRs provide a massive amount of data which in principle can be used to create useful insights. The data contains the medical history of a patient including medical measurements, diagnoses, prescribed drugs, and demographics (see figure 1.1, where procedures (CPTs), lab results (LABs), visits to primary care physician (PCP) and visits to specialists (SPEC) are part of the EHR). Based on those values, we could obtain the following insights:

- Effects of drugs
- Medical costs for certain diseases
- Duration and recovery percentage of certain diseases
- Correlation between demographics and certain diseases

- Link between current health state and health history
- Classification and prediction of future health states based on history

Those insights can be offered on an individual level, which means a right intervention to the right patient at the right time. EHR analytics can be used to have a personalized care and benefits the healthcare system by cutting costs and improve outcomes [40].

EHR analytics is an active research field as a lot of different problems need to be solved, like different codings, privacy, interpretation, and large amounts of data. In the following sections we talk about current EHR analytical methods.

1.3.1 Querying

Analytics in epidemiology on EHRs is typically done through querying a database [25]. A specialist has a hypothesis about correlations between conditions or patients. He can support this idea by selecting cases in EHRs and analyzing the results of his query.
This method is based on the knowledge and experience of a specialist. The information has to be actively sought after and unexpected or complex correlations are not always considered. Some complex relations cannot be found because of the limitations of the querying language, ex. a first-order logic query language. Currently The Health Improvement Network (THIN) research team at UCL conducts these kinds of research [5].

1.3.2 Data Analytics

More advanced methods are applied to EHRs than querying. In general, the goal is to find patterns in the EHR data which then can be used to predict outcomes of treatments or classify a patient's disease trajectory [30].

Several predictive/classification methods from machine learning can be used and show promising results [13]. Those results are achieved by using exploratory methods which are applied to the EHR data. We also note that methods in [13] as Multi-layer Perceptron networks are not ideal for prediction of time-series like medical data, see section 5.4. We conclude that there is still a lot of room for improvement.

More specialized approaches are also applied to EHR data [59] besides the above mentioned machine learning techniques. An EHR of a patient can be transformed into a matrix structure, see figure 1.1. On these matrix structures, large-scale data mining algorithms can be applied. Those make it possible to mine patterns in EHR data. The patterns found can be used as the basis for predictive models.
It is also possible to define patient similarities [56]. So when a patient is similar to a previous known case, his treatment can be based on those previous experiences. In a way, it could be seen as a similar approach as a recommender system [**?**].

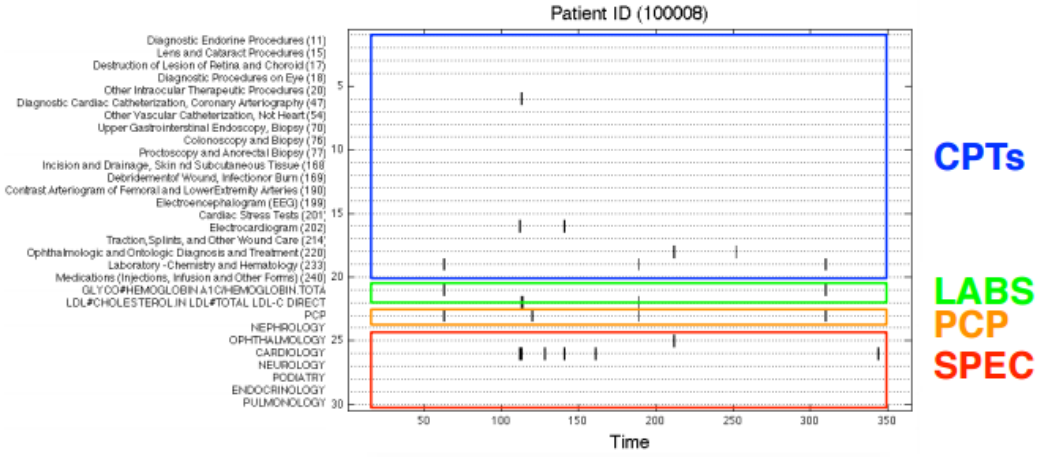


Figure 1.1: Example of an EHR transformed into a matrix structure [59]

1.3.3 Data Mining

In 2015, a more data mining approach is used to find patterns in EHR data on a dataset of the Danish population [29]. The results from [29] are clusters of disease trajectories which will be used to validate our approach described in chapter 2. Their results are used because it is currently the largest study performed on EHRs. You can find an example of a clustered trajectory in figure 1.2.

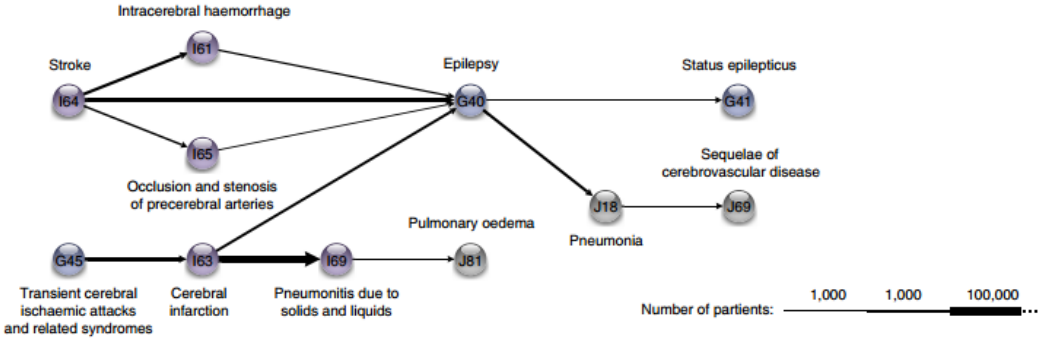


Figure 1.2: Cerebrovascular disease trajectory cluster for the Danish population [29]

The dataset which is used to apply the data analysis on, consist of EHRs collected over 15 years on over 6 million patients in Denmark. The size of this dataset makes it possible to retrieve statistically significant results.

They start with finding pairs of patients' diagnoses which have a strong temporal

correlation between them. After finding the correlated pairs, a test for directionality is applied. From this, only the pairs with a high enough indication for a direction are kept, ie which often appear one after the other within a certain time span.

The directed pairs are then connected into longer trajectories when they have overlapping diagnoses. The found trajectories are then clustered. From the clusters, diagnoses can be found which are key in the disease progression. Those key diagnoses could be used to predict future disease progression of patients.

1.4 Outline

In this chapter, we introduced you to the field of Electronic Health Record Analytics and some current research directions. We state this is an active research field with a lot of directions left to explore.
In chapter 2, we start with introducing some terminology and general background knowledge needed to introduce our approach in the field of Electronic Health Record Analytics. Then we explain our own approach, namely generalized Word2Vec approaches, and how those can be applied on medical data.
In chapter 3, we show the methods we used to implement our own approaches. This allows us to execute experiments and validate our approaches.
In chapter 4, we talk about the conclusions we can take from this thesis and sum up the results we achieved.
The last chapter, namely chapter 5, we explore some possible improvements on our approaches and some future directions left to explore.

1.5 Conclusion

EHRs contain important information of a patients medical history and current state. When a large amount of EHRs is available, relevant results can be found in the form of patterns. Those pattern can be used to predict and improve medical outcomes on a personal level.
The methods we describe vary from simple to very complex. But there is still room for improvement, especially in the field of advanced machine learning algorithms. The results of the Danish paper can be used to have a first validation of our approach, namely a generalized Word2vec approach.

In the next chapter we explain the required background knowledge to understand our approach on finding patterns in EHRs. After introducing you to those concepts, we also explain our approach, namely a generalized Word2vec approach.

Chapter 2

Generalized Word2Vec for Electronic Health Record Analytics

2.1 Introduction

In this chapter we introduce important concepts which are needed to understand our approach. These concepts can be used to solve problems described in chapter 1.

We start with explaining some background knowledge in section 2.2 such as time series and machine learning. We focus on basic concepts from machine learning and then focus more on neural networks. The main part to understand our approach is the introduction of Word2vec in section 2.3. This is then used to introduce Deepwalk as an extension on Word2Vec in section 2.4.

After introducing those concepts, we explain our generalized Word2Vec approaches in section 2.5.

2.2 Background Knowledge

2.2.1 Time Series Analysis

A time series consists of data points over a certain time period. We refer to this as a sequence of states. Where a state represents a data point and can differ from a single value to more complex representations like pictures.
The domain of time series analysis handles around extracting information or relations from a time series. It can have different goals like forecasting, classification, or exploratory.

A medical history of a patient can be seen as a time series, namely a sequence of EHRs. This means that methods which are applied on time series, are also applicable

on medical data to find patterns.

2.2.2 Machine Learning

Machine learning is a data driven approach with as goal to build a model which can be used to make predictions or decisions. Note that this model can be used to predict outcomes of time series. This task is done by algorithms which are able to learn models based on examples given by the designer. Based on the examples, machine learning aims to tackle 3 types of problems, namely supervised learning, unsupervised learning, and reinforcement learning.
Supervised learning is concerned with the learning task where there are examples given with their corresponding label. Unsupervised learning is similar to supervised learning only no labels are given. We won't go into reinforcement learning.
We can also classify the problems according to the desired output of our model. Those main tasks consist of classification, regression, and clustering.

We mention some used methods in the field of machine learning. These are used in above mentioned problems.
In the field of classification neural networks are used to achieve state of the art results. For regression, support vector machines can be used. One of the most popular methods for clustering is K-means.

2.2.3 K-nearest Neighbors

The k-Nearest Neighbors (knn) algorithm is a simple machine learning algorithm [16]. It will be used in our generalized Word2Vec approach.
When you are in a supervised context, you have several instances with labels. When you retrieve a new instance, you want to predict its label. Based on a defined similarity measure (ex. Euclidean distance), you can look for the k nearest labeled instances distanced to the new instance. From those, you pick the most common label in the pool of the k nearest instances. This label becomes your predicted label for the new instance.

2.2.4 K-dimensional Tree

A naive way to calculate the nearest neighbors for a new element in a vector space, is by comparing all members with the new element and keep track of those distances. A more efficient way is to use a k-dimensional tree (kd tree) [54]. In this section we explain the workings of this approach in more detail [42].

A kd tree is a way of storing k-dimensional points. It is a binary tree where each node represents an element in the vector space. The node also contains information on how the tree is split up. It keeps track of the plane it is split on and the left and right sub tree. In figure 2.1 you can see an example on how a simple kd tree is made and how it splits up the x,y plane. In the upper figure, the splitting plane is not

mentioned, it is the $y = 5$ plane for the $[2, 5]$ and $x = 3$ for $[3, 8]$.

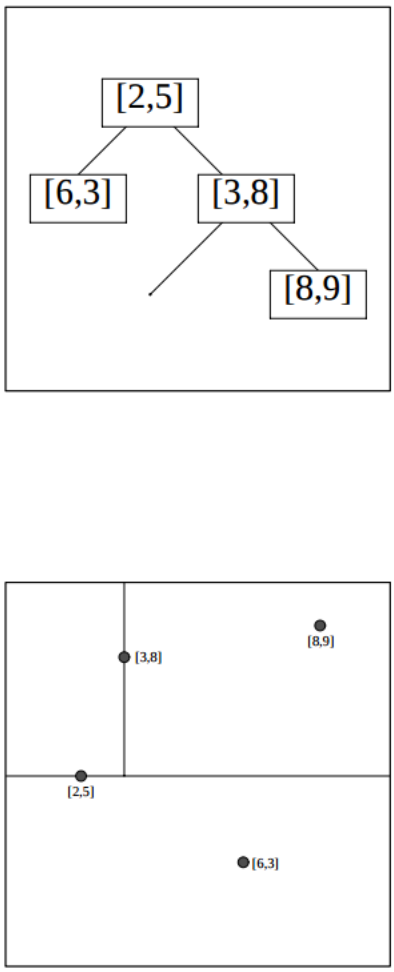


FIGURE 2.1: Representation on how a binary kd tree splits up the plane [42]

Now that we can construct the kd tree, we can use it to find the nearest neighbor for a certain input point.
We start with root node and recursively go down the kd tree. At each node it goes to the left or right depending on if the input is lesser or greater than the current nodes value on the splitting plane. Once it reaches a leaf node, it marks this node as the current nearest neighbor.
The algorithm unwinds the recursion of the tree, performing the following steps at each node:

- If the current node is closer than the current best, then it becomes the current best.

- It checks if it is possible whether there is the possibility of closer points on the other side of the splitting plane. It makes a hypersphere around the current node with a radius equal to the current nearest distance.
  - If the hypersphere crosses the splitting plane, there could be a closer point on the other side. This means the algorithm will move down the other branch of the current node.
  - If the hypersphere does not cross the splitting plane, the whole other branch can be skipped.

This algorithm is easily extended to find the k-nearest neighbors by keeping track of the k current bests.

2.2.5 Neural Networks

A neural network is a machine learning approach based on biological neural networks. It can be used to find patterns and do predictions on time series. Those time series can be medical data.

Perceptron

The basic component of a neural network is a perceptron [50]. A perceptron takes multiple binary inputs and has a single binary output (see figure 2.2). Each input has a corresponding real numbered weight w_j. The output is decided on the following equation:

$$output = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if if } \sum_j w_j x_j > \text{ threshold} \end{cases} \tag{2.1}$$

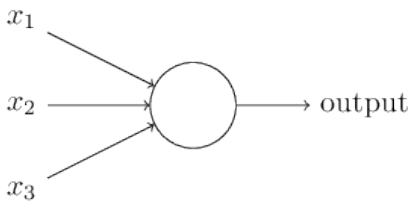


Figure 2.2: Simple presentation of a perceptron [43]

We can build a network by connecting multiple perceptrons (see figure 2.3). By building these networks, more complex decisions can be made. The reason for this, is that once there are atleast 3 layers of perceptrons (and non-linear activation functions), the network can find non-linear relations between the input and output [27].

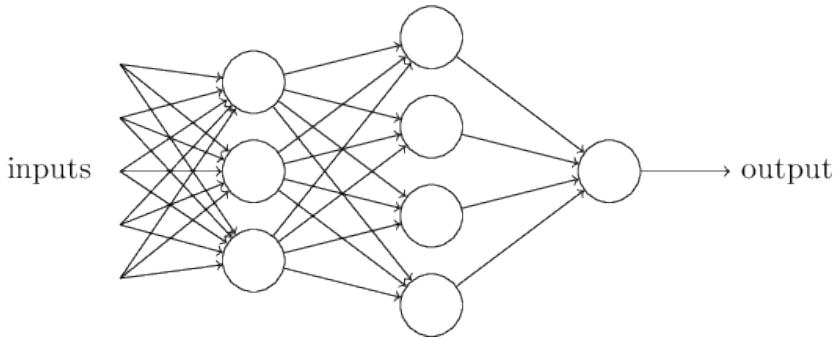


FIGURE 2.3: More complex network made by connecting multiple perceptrons [43]

Now we have seen how a general network is constructed, we look at some vocabulary.
In figure 2.4, we see a four-layer network. As mentioned on the figure, we call the first layer the input layer, the last layer the output layer, and the layers in between are called hidden layers. Sometimes a multiple layer network is referred to as multilayer perceptrons or MLP.

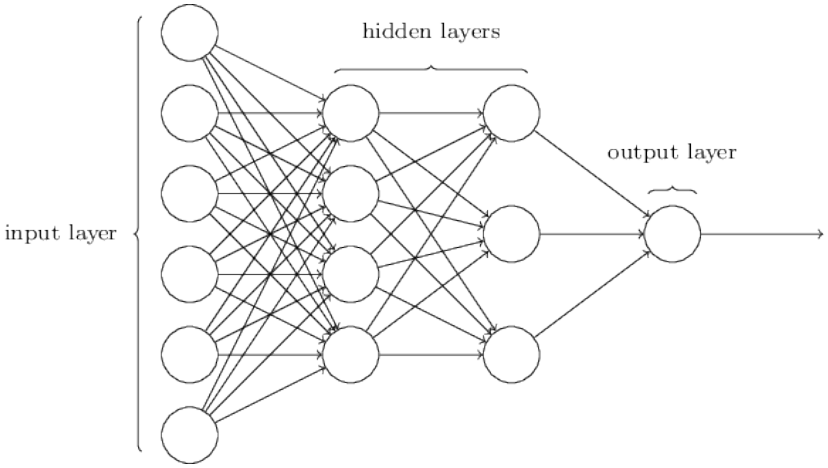


FIGURE 2.4: General vocabulary of a multilayer network [43]

Training a network

To train a neural network, we input an example with a known label. The network will calculate a certain output based on the current weights (forward pass). When this output is incorrect, it should be possible to adjust the weights with as effect that the network now has as output the correct label (backward pass). Note that the change in weights, should only effect the output by a small bit (see figure 2.5). The reason for this is that otherwise all the previous inputs could now be labeled incorrectly. So, the concept of training a neural network means, adjusting the weights

in a way that the behavior of the network doesn't change completely on the previous seen examples but that the current examples is labeled correctly.

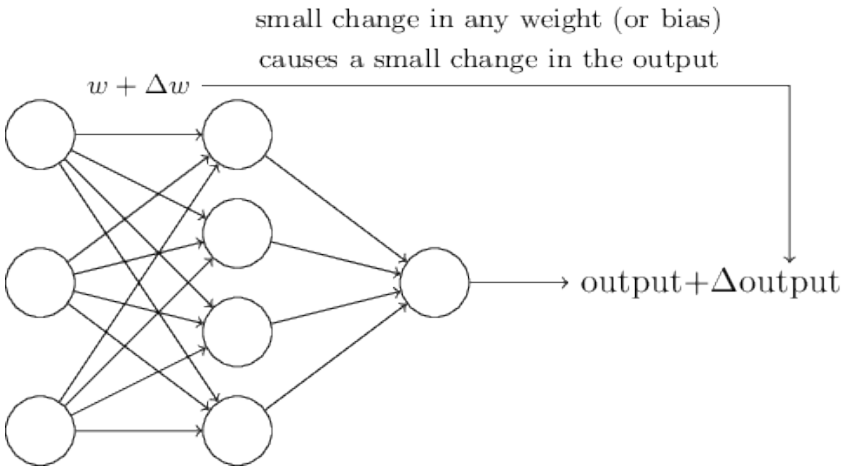


Figure 2.5: Small change on the weights, only has a small impact on the output [43]

To achieve this effect, we change our above explained perceptrons to sigmoid neurons. A sigmoid neuron has the same basics as a perceptron. It still has inputs but now it also has a bias b. The inputs still have weights but the weights can now range between 0 and 1. The output is now calculate with $\sigma(w * x + b)$ where σ is the sigmoid function. This results in the following formula:

$$\frac{1}{1 + exp(-\sum_j w_j x_j - b)} \tag{2.2}$$

The sigmoid function makes it possible to calculate the gradient and makes the output a linear combination of Δw_j and Δb as $\Delta output$ is approximated by

$$\Delta output \approx \sum_j \frac{\partial output}{\partial w_j} \Delta w_j + \frac{\partial output}{\partial b} \Delta b \tag{2.3}$$

Because of the linearity, it is now easier to choose changes for the weights and biases to achieve a correct output. By adjusting the weights, we will train our network to achieve a higher accuracy on the seen examples.

2.2.6 Backpropagation

Backpropagation is an algorithm which is used to train neural networks [52]. It calculates the gradient of a chosen cost function with respect to the individual weights. Based on the gradient, the weights are updated and the cost function is minimized.

Terminology

We use w^l_{jk} to denote the weight corresponding to the connection between the k^{th} node in the $(l-1)^{th}$ layer and the j^{th} node in the l^{th} layer. We use b^l_j for the bias of the j^{th} node in the l^{th} layer and a^l_j for the activation of the j^{th} node in the l^{th} layer. See figure 2.6.

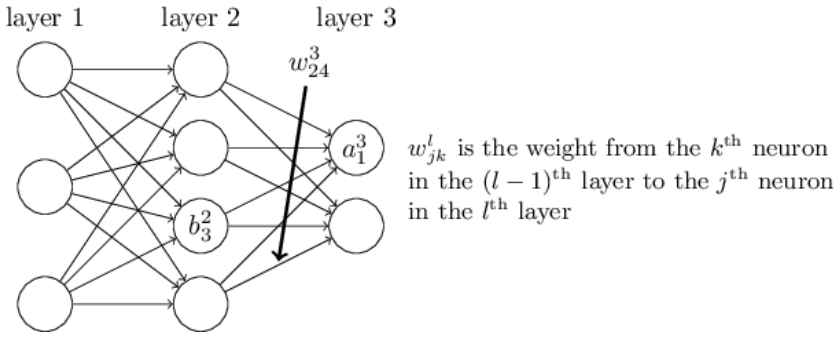


FIGURE 2.6: Visual representation of the terminology for a neural network [43]

We can now convert this notation to a vector representation. We remove the indexes for the node numbers which results in the following:

$$a^l = \sigma(w^l a^{l-1} + b^l) \tag{2.4}$$

Cost function

As mentioned before, backpropagation has as goal to calculate the partial derivatives of the cost function C with respect to each weight and bias.
The cost function has to fulfill certain criteria. The first one is that it needs to be possible to write it as a summation over cost functions for individual training examples. Secondly, it needs to be derivable. And lastly, the cost function is a function of the activations of the last layer.

Fundamental equations

Backpropagation has 4 equations. They allow us to calculate the error for each node and adjust the weights based on the gradient descent.

First we calculate the error of each node which is based on how much the cost function is influenced by each of its activation and on how much the activation function is influenced by z^L_j:

$$\delta^L_j = \frac{\partial C}{\partial a^L_j}\sigma'(z^L_j) \text{ with } z^L_j = \sum_k w^L_{jk} a^{L-1}_k + b^L_j \tag{2.5}$$

This can be written as a neat vector equation:

$$\delta^L = (a^L - y) \circ \sigma'(z_j^L) \tag{2.6}$$

The next equation explains why the algorithm is called backpropagation. The equation calculates each layers error vector based on the layer after it, it propagates the error back over the layers:

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \circ \sigma'(z_j^L) \tag{2.7}$$

With those 2 equations we calculate the error in each layer of the neural network. Those errors can be used to calculate the derivatives of the cost function with respect to the weights and the biases:

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l. \tag{2.8}$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l. \tag{2.9}$$

When the derivatives are calculated, we can apply the gradient descent and update the weights and biases accordingly. This process represents the learning of a neural network.

2.3 Word2Vec

2.3.1 Motivation

We will explain Word2Vec in this section. It is explained from a linguistic point of view. This explanation is needed to introduce our generalized Word2Vec approach which can be applied to medical data.

In natural language processing tasks, a good representation of words helps learning algorithms perform better. A representation is learned which maps words to vectors in a low-dimensional space compared to the vocabulary size. In this representation, we try to map context-similar words close to each other in the new vector space. The new representation is sometimes also called a 'word embedding'.
We could say in an informal way: a linguistic background is made which the learning algorithm can use.

2.3.2 Skip-gram

There are two main models used for Word2Vec [41], namely Continuous Bag-of-Words (CBOW) and the Skip-Gram model.
The first one tries to predict a word based on a given context (ex. predict Paris when capital France is given). And the second one does the inverse of this approach [49]. Empirical results have shown that the Skip-Gram model tends to do better on larger

datasets [58] and gives a better representation for infrequent words [1]. In medical data there are often infrequent cases which are important. For those reasons, we choose to go further with the Skip-Gram model.

So one way of learning a Word2Vec representation of a corpus $Text$, is by using the Skip-Gram model.
Based on given words w and their contexts c, we set the parameter θ of $p(c|w;\theta)$ to maximize:

$$\arg\max_{\theta} \prod_{(w,c\in D)} p(c|w;\theta) \tag{2.10}$$

with D the set of all word and context pairs we extract from the corpus.
Here we also note that $p(c|w)$ is indeed the chance of a context appearing after seeing a specific word as mentioned before.

Finding word-context pairs

Given a sequence of words, we define their context based on n-grams [24]. In figure 2.7, n-grams are explained on the sentence "This is a sentence".

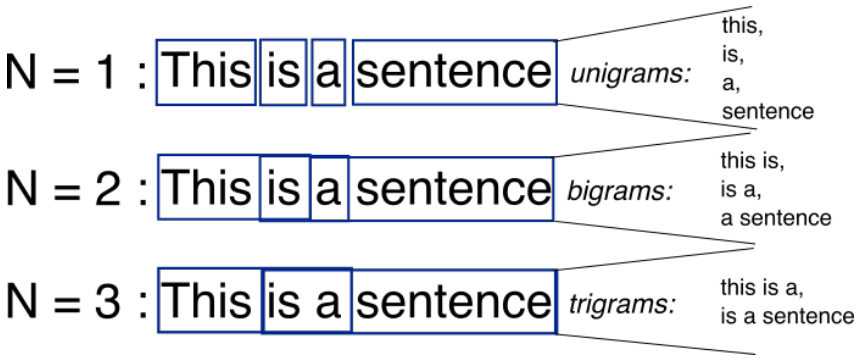


FIGURE 2.7: Explanation of n-gram [4]

For the Skip-gram model, we define the context of a word w_t as w_{t+j} with j between $-c$ and c. A larger c typically results in more specific contexts for training examples and thus can lead to a higher accuracy if there is enough data to back this up. Because if there are not enough cases handling those specific contexts, the correlation between the word and his context is less statistically significant.

Parameterization

We start with rewriting the conditional probability using soft-max:

$$p(c|w;\theta) = \frac{e^{v_x * v_w}}{\sum_{c'\in C} e^{v_{c'} * v_w}} \tag{2.11}$$

where v_c and v_w are vector representations for c and w, and C is the set of all available contexts. This means that the parameters θ are v_{c_i} and v_{w_i}. Computing the optimal parameters is very expensive because you need to calculate this over all contexts c'. We also switch from product to sum by taking the logs:

$$\arg\max_{\theta} \sum_{(w,c)\in D} \log p(c|w;\theta) = \sum_{(w,c)\in D} (\log e^{v_c * v_w} - \log \sum_{c'} e^{v_{c'} * v_w}) \tag{2.12}$$

2.3.3 Negative Sampling

To compute the vectors using the Skip-gram model more efficiently, we introduce negative sampling [21].
Instead of calculating $\sum_{c'\in C} e^{v_{c'} * v_w}$ over all contexts, we make a set D' which consists of randomly sampled word-context pairs. With this new set, we remove the costly term $\sum_{c'\in C} e^{v_{c'} * v_w}$ and replace it with $\sum_{(w,c)\in D'} e^{v_{c'} * v_w}$.

In a less formal way: we are not making sure that if words appear in the same context, their vectors are more similar than all the other word vectors, but only more similar than several vectors chosen randomly. This makes the Skip-gram model usable in terms of computation.

2.3.4 Neural Networks

When the Word2Vec algorithm is trained using the Skip-gram model, one will have a lookup table. This table contains the mapping of words to their vector representation. This lookup table can be found by training a 2-layer neural network with as cost function the function described in the previous section. The training can be done with backpropagation for example.
The trained 2-layer neural network can be placed in front of another neural network [44]. It will convert the words to their vector representation and feed it into the next neural network. It is empirically shown that this can improve the results of the neural network by putting the lookup table in front of it. As mentioned before, in a way, you offer background knowledge to the neural network.

2.4 DeepWalk

DeepWalk is an approach where graph structured data is transformed into sequences of vertices [48]. Word2Vec is then applied on those sequences to learn a good vector representation for the vertices. We can say that it is an extension on the Word2Vec approach.

In figure 2.8, we see an overview of the DeepWalk algorithm. It exist of two parts. First a random walk generator. For each vertex v_i of the graph G, it will generate a random walk of length t. It will do this γ times but the order to which the vertices are

traversed, is randomly ordered each pass. With those walks, a sequence of vertices is generated.
Secondly, those vertices are used for Word2Vec. This process is explained in section 2.3.

Algorithm 1 DeepWalk(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$
Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree T from V
for $i = 0$ to γ **do**
  $\mathcal{O} = \text{Shuffle}(V)$
  for each $v_i \in \mathcal{O}$ **do**
    $\mathcal{W}_{v_i} = RandomWalk(G, v_i, t)$
    SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
  end for
end for

FIGURE 2.8: Overview of the DeepWalk algorithm [48]

2.5 Generalized Word2Vec Approaches

In this section we explain our approach on how to find patterns in EHRs. For this, we use generalized Word2Vec approaches to learn disease embeddings.

2.5.1 Data representation

Before we can start with explaining how Word2Vec can be applied on EHR data, we start with introducing our data representation.
The medical history of a patient is a time series of EHRs. On certain timestamps, an EHR is available containing the medical state of that patient. We call this EHR the vector m_t^p, with p a patient number and t a timestamp. This means that each patient has a sequence of vectors $s^p = m_t^p, m_{t+1}^p, m_{t+3}^p, \ldots$ Each vector m_t^p contains certain values depending on the EHR. It can contain values of the timestamp, demographics, blood pressure, diagnoses, and others.

2.5.2 Generalized Word2Vec

As explained in section 2.3, Word2Vec is applied on a large text corpus containing a large amount of sentences. After applying this method, an embedding is found that represents words in a new vector space. In this vector space the relationship between

words is shown by the distance of the words from each other.

A large medical dataset containing EHRs from different patients, can be seen as a large text corpus. It contains patients, each having a sequence s^p, which is equivalent to one sentence. All the different patients sequences, make the whole dataset similar to how sentences make a text corpus. Each vector m_t^p is removed from its link to the patient and his timestamp to make it not unique anymore, we call this vector m. This is to make the link to words. Different sentences can contain the same words, similar to how the vectors m are in the different sequences.

With those links, it is possible to apply Word2Vec not only to simple objects like words, but also on more abstract objects like vectors. We call this a generalized Word2Vec approach. From this, we can learn an embedding for the vectors m in the new vector space. In this new vector space, the relationships between the vectors m are found based on their occurrences with others in the sequences.

We can now apply Word2Vec to medical data. The vector m is an EHR and by applying the generalized Word2Vec method, we find an embedding for the vectors m. The result is a disease embedding which projects the EHR data to a new vector space and projects EHRs which have a close relationship to each other, close to each other in the new vector space.

2.5.3 K-Nearest Neighbors Word2Vec

A problem with using an embedding which is found on a certain dataset, is that it is possible that certain instances are not present in the dataset. In the case of a complete new instance, a Word2Vec model cannot find a representation for this instance. As we are working in the context of a generalized Word2Vec approach, where instances are represented by complex objects like vectors, the chances of this happening increases. It increases because there are a lot more possible combination of vectors possible than there are words in a dictionary for example.

Therefore we introduce a K-Nearest Word2Vec approach. As we are working with more abstract objects, namely vectors, we can extend our generalized Word2Vec approach with a knn feature (see section 2.2.3). After the training of a Word2Vec model, we have learned a lookup table which maps a known vector v_{old} to his new vector representation v_{new}. When a not-yet-seen vector $v_{unknown}$ needs to be mapped to his v_{new}, a normal Word2Vec is not capable of this.
In our approach, we find the k-nearest neighbors of the $v_{unknown}$ from the old representations v_{old} in the lookup table. We then take the new representations of the knn, and combine them using a weighted average to estimate the new representation v_{new} for $v_{unknown}$. This way our approach is capable of handling complete new instances.

In short: we look for the knn of the $v_{unknown}$ from all the known vectors in the lookup table in their original representation v_{old}. Based on the found knn, we take

an weighted average of the old representations v_{old}. This weighted average is the v_{new} for the $v_{unknown}$.

2.5.4 Generalized Deepwalk

Deepwalk itself is hard to apply on EHR data as it starts from a graph structure. It however provides an extra Word2Vec approach which can be generalized to more abstract objects like vectors.

One application of Deepwalk is the need for less data to achieve a good Word2Vec model. When the EHR dataset becomes very large, it would take a considerable amount of time to train a Word2Vec model. Therefore it would be interesting to transform the EHR data into a weighted graph structure. The weights are based on the frequency of diagnoses following each other in all the sequences.
After the graph transformation, the amount of weighted random walks can be limited to create a smaller set of sequences based on the original dataset. Those sequences can be used to train a Word2Vec model faster as there is fewer data. The same logic from the previous two sections is applicable to generalize DeepWalk from words to more abstract objects.

2.6 Conclusion

In this chapter we talked about general machine learning concepts and focused on Word2Vec. We conclude that Word2Vec is used to find good representation of words based on their context. It also causes that similar words will be close to each other in this new representation.

With the concepts explained, we introduced our approaches. We extended Word2Vec so that it can handle more abstract objects than words, namely vectors representing an EHR. We also extended Word2Vec to find a representation for a new instance based on knn. Also DeepWalk is generalized to more abstract objects.

Chapter 3

Validation

3.1 Introduction

In this chapter

DiseaseMapping (generalization) OSIM Clusters TensorFlow DL4J

3.2 Dataset

To validate the approaches mentioned in chapter 2, we used a dataset generated by Observational Medical Dataset Simulator Generation 2 (OSIM2) [46]. This dataset is used by Observational Medical Outcomes Partnership (OMOP) to validate their methods to predict the effects of drug treatments. It contains around 10 million of hypothetical patients based on Thomson Reuters MarketScan Lab Database (MSLR). MSLR contains administrative claims between 2003 ad 2009 from a privately-insured population.

The OSIM2 dataset is contains multiple database tables which are dumped as comma-separated values (csv) files. To make it easier to work with this dataset, we joined the multiple files into one file with on each row an event of a patient containing all relevant information. The relevant information which is kept is: birth year, gender, condition type, condition, time difference since previous diagnosis, and season (summer, fall, winter, spring). Thus, one EHR is 7 dimensional.

Using our approaches on this dataset, we can compare our results to the found clusters in Anders Boeck Jensen et. [29]. Although these found clusters are not a golden standard, it is a first validation point for our approaches.

3.3 Software

3.3.1 TensorFlow

TensorFlow is an open-source machine learning software library released at the end of 2015 [8]. It is developed by the Google Brain Team.
It provides a Python interface for efficient C++ code. After some time, we found that Tensorflow is not well documented at the moment and does not gave the needed freedom to easily rewrite some core features of their Word2Vec implementation, for example manipulating the internal trained lookup table.

3.3.2 DeepLearning4Java

DeepLearning4Java (DL4J) is an open-source machine learning software library released by Skymind [57].
It runs on their scientific computing engine ND4J which provides fast matrix operations. DL4J is completely written in Java and provides a lot of freedom to manipulate lookup tables and extend their Word2Vec methods to work on abstract object like vectors. The developers are active on Gitter and offer a lot of information on how to use certain parts of DL4J.

3.4 Experiment Setup

3.4.1 Generalization

As described in chapter 2, we use generalized Word2Vec approaches to find patterns in EHR data. We use the OSIM dataset and represent each EHR as a 7 dimensional vector. This vector is comparable to a word in a normal Word2Vec approach and functions as the abstract object in our generalized Word2Vec approaches.

Because we are working with high dimensional data, most instances of OMOP are quite unique. This is mainly due to a combination of specific disease codes and time intervals. To find patterns which are more general applicable, we start with generalizing our data. With generalizing we mean that values for some attributes are projected into categories. For example, the time intervals are projected into 4 categories.

It is easy to generalize concepts as time intervals and demographics, for example we can say that people between the age 0 and 10 belong to category A. This becomes complex for disease diagnoses as it is domain specific and requires a lot of knowledge to generalize those. We talk about our solution to this in section 3.4.2.

To see the effect of our generalization, see table 3.4.1. You can see the 3 most common vectors from our dataset before and after the generalization.

Before Generalization	
Vector	*Occurences*
[1956.0, 8532.0, 65.0, 5.00000701E8, 0.0, 3.0]	17574
[1954.0, 8532.0, 65.0, 5.00000701E8, 0.0, 3.0]	17536
[1955.0, 8532.0, 65.0, 5.00000701E8, 0.0, 3.0]	17476

After Generalization	
Vector	*Occurences*
[6.0, 8532.0, 65.0, 784955.0, 1.0, 3.0]	282086
[7.0, 8532.0, 65.0, 784955.0, 1.0, 3.0]	235459
[5.0, 8532.0, 65.0, 784955.0, 1.0, 3.0]	230216

TABLE 3.1: Three most common vectors in our dataset before and after generalization

3.4.2 Disease Code Mapping

In the section we discuss the method to generalize the disease codes used to label the diagnoses in the OMOP dataset. The basic idea is that we want to generalize the diagnoses. For example, a bruise on your right leg is the same diagnoses as a bruise on your left leg. We would have liked to use the method described in [18] to map the codes but therefore the mapping made by UMLS between ICD 10 and MedDRA is needed. This is however not publicly available.

The OMOP dataset uses MedDRA disease codes for the diagnoses. We described the MedDRA disease codes in chapter 1. We mentioned that MedDRA does not have a clear hierarchy. With a clear hierarchy, it would be trivial to generalize a disease code to its highest hierarchy.
To solve this, we map the MedDRA disease codes to the ICD 10 disease codes. The reason behind this is that ICD 10 provides a clear and easy to use hierarchy. Besides the easy to use hierarchy, we also need the ICD 10 disease codes to make it possible to compare our results with Anders Boeck Jensen et. [29] as their results also use ICD 10 disease codes.

The mapping is based on the description of each disease code. Both MedDRA and ICD 10 have a short medical description of each code. Each description is first filtered from stop words. Afterwards, the MedDRA code is matched to the ICD 10 code based on the matching of both description. The matching is done on the highest percentage of words matching.

In figure 3.1, we show the statistics of this mapping process. Note that we only take disease codes into account if they are also part of the OMOP dataset. In the figure, each bar represents the percentage of the words matching between descriptions. The height of the bars represent the percentage of all disease codes in the OMOP dataset who have this amount of matching percentage.

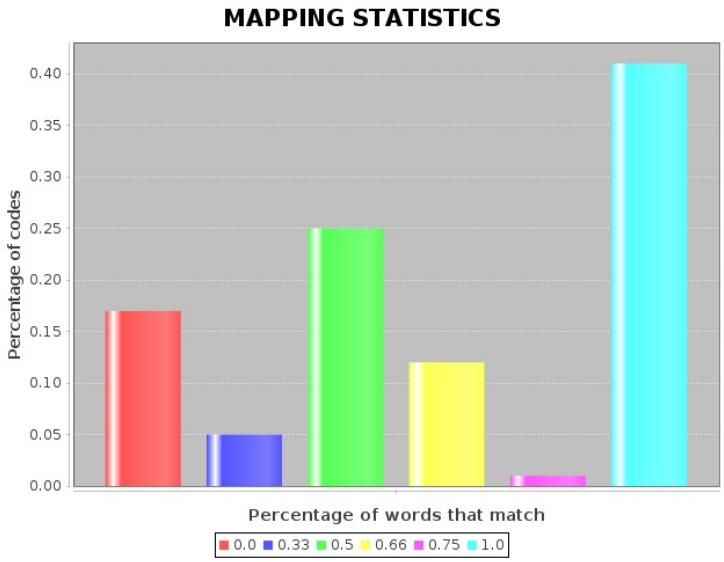


Figure 3.1: Statistics of the mapping approach

We have an average of 63% of the words from the description that matches between MedDRA and ICD 10 descriptions. We also see that around 85% of all the disease code mappings have a match of atleast 33%. We assume this is already a good mapping as medical terms are quite specific and if 33% matches, the upper hierarchy will be a good enough match. For the codes which have a zero percent match, the Damerau-Levenshtein algorithm [9] is applied. The algorithm calculates the edit distance between two strings using character insertion, character deletion, character replacement, and adjacent character swaps. The description with the lowest edit distance, is then chosen as best match.
Another option would have been to remove those codes from our dataset. As an EHR contains more information than only the disease code, a lot of other information would have been lost. This however is not tested and is a possibility for future work.

3.5 Results

3.5.1 Experiments

To test our approaches we will compare our results with the clusters found from Anders Boeck Jensen et. [29]. For this, we also need to generate our own found clusters. This method is described in this section. Note that the comparison with the Danish paper is not ideal but it is currently the largest study performed on EHRs. We refer to chapter 5 for more information about another possible experiment in the future.

First we apply our approaches on the OSIM dataset. For a description of the used parameters and their functions, see section 3.5.2. As end result, we get a lookup table containing the mapping between the old vector space and the new one. From this lookup table, we take EHRs and find their k-nearest neighbors depending on their mapping in the new vector space (see section 3.5.2 for more information about clusterK). Depending on the approach, we take certain EHRs. For the generalized Word2Vec, we check all EHRs in the lookup table. For the knn Word2Vec, we only check the complete new instances added to the lookup table using the knn method. For DeepWalk, we again take all EHRs in the lookup table.

Based on the formed clusters from our lookup table (Word2Vec clusters), we can now compare it to the Danish clusters. We test the matching between the two clusters based on two experiments.
The first one is as follows (we later call this experiment 1):

- We start with an element e_{wv} from the lookup table and form a Word2Vec cluster C_{wv} around it.
- We then check to which Danish cluster C_d element e_{wv} belongs.
- For each element in C_{wv}, we check if it belongs to C_d.
  - If it does, we add 1 to the number $t_{matches}$.
  - If it does not, we add 1 to the number $t_{non\_matches}$.
- After doing this for all elements in C_{wv}, we calculate the match percentage as $p_{match} = \frac{t\_matches}{t_{matches}+t_{non\_matches}}$.

The second one is as follows (we later call this experiment 2):

- We start with an element e_{wv} from the lookup table and form a Word2Vec cluster C_{wv} around it.
- We then check to which Danish cluster C_d element e_{wv} belongs.
- We set t_d equals to the amount of elements in C_d.
- For each element in C_{wv}, we check if it belongs to C_d.
  - If it does, we add 1 to the number $t_{matches}$.
- After doing this for all elements in C_{wv}, we calculate the match percentage as $p_{match} = \frac{t\_matches}{t_d}$.

For each method, we calculate the average p_{match} over all elements of the lookup table and keep track of the maximum and minimum value for p_{match}.

3.5.2 Parameters

The effectiveness of a neural network like the one used to learn the disease embeddings, depends on parameter tuning [10]. Therefore we explored some different settings. Note that this is not an extensive parameter tuning setup, but an overview on the effect and logic behind some parameters. An extensive parameter tuning setup is possible as future work.

In table 3.5.2, you can see an overview of the different parameters for each approach. For each approach, all possible combinations of the mentioned parameters have been tested.

Parameter	Generalized Word2Vec	Knn Word2Vec	DeepWalk
Vectorlength	[50, 100]	[50, 100]	[50, 100]
Batch Size	500	500	500
Epoch	1	1	1
Window Size	[5, 10, 15]	[5, 10, 15]	[5, 10, 15]
Learning Rate	[0.025, 0.1]	[0.025, 0.1]	[0.025, 0.1]
Minimum Word Frequency	[5, 10]	[5, 10]	[5, 10]
ClusterK	[100, 1000, 5000]	[100, 1000, 5000]	[100, 1000, 5000]
K	/	[10, 50, 100]	/
Walklength	/	/	[5, 10, 15]

Table 3.2: Parameters which are tested for the different approaches

Vectorlength decides the dimensions of the new vector space to which the diseases are projected to. A larger size is more expressive but a smaller size decreases the complexity of the vocabulary after the training. Each dimension can be seen as a representation of an unknown linear relation to some context-disease pairs [36].
Batch size is the number of training examples used in one gradient descent step (both forward and backward pass).
Epoch is the amount of times you go over all the training examples and use them to train your neural network.
Window size is the size of the n-gram we use to create contexts. A larger window tends to make more specific contexts and therefore get a better model [41] [35].
Learning rate decides how the weights are adjusted during backpropagation. A large value can make the neural network learn faster but may also cause the network to not learn at all. A small value on the other hand can cause a slow convergence or overfitting [34]. Note that in our implementation there is a decay of the learning rate as the number of examples left decreases.
Minimum Word Frequency removes instances from the training set below the minimum. A low frequency causes some instances to only have a few contexts where relations can be learned from. So a higher frequency could improve the accuracy for a Word2Vec model but also throw away information. Note that the influence of this parameter is not well researched.

ClusterK is the parameter which creates the Word2Vecclusters as described in the previous section. A higher clusterK causes the Word2Vec cluster to be larger and a lower value the other way around.
K is a specific parameter for the knn Word2Vec approach. It decides on how many nearest neighbors the new vector representation is based on. For more explanation see chapter 2.
Walklength is a specific parameter for DeepWalk. It decides how long the generated sequence will be. For more explanation see chapter 2.

3.5.3 Generalized Word2Vec

Here we discuss the results from our two experiments on the generalized Word2Vec approach. We trained a Word2Vec model on the complete OSIM2 dataset. From this model, we compare the Word2Vec clusters with the Danish cluster as described in section 3.5.1. For each parameter, we fix the other parameters and check their influence on the matching percentage. After checking the individual parameters, we take the parameters setting with the best matching percentage and interpret those results.

Vectorlength

Both in experiment 1 and experiment 2, we found that the vectorlength had no substantial influence on the matching percentage. See figure 3.11 for the matching percentage of experiment 1 and figure 3.9 for experiment 2. The two bars in the figures each show a parameter value and the black stripes both represent the minimum and the maximum value of the matching percentage. Similar results are achieved with all the other parameter settings.

We conclude that a vectorlength of 50 is already expressive enough to differentiate between different diseases. Only sometimes the vectorlength of 100 gives a small gain in matching percentage which can be explained by the increase of expressiveness of the new vector space.

Window Size

Both in experiment 1 and experiment 2, we found that the window size had no substantial influence on the matching percentage. See figure 3.11 for the matching percentage of experiment 1 and figure 3.9 for experiment 2. Similar results are achieved with all the other parameter settings.

We expected the window size to have more impact as a higher size makes more specific contexts. And more specific contexts, tend to make better contexts. A reason for this could be that we don't have enough data to justify more specific contexts. Also, because our sequence lengths have an average length of around 19, a window size of 5 is not that much different than a size of 10 as in many cases it will cut of

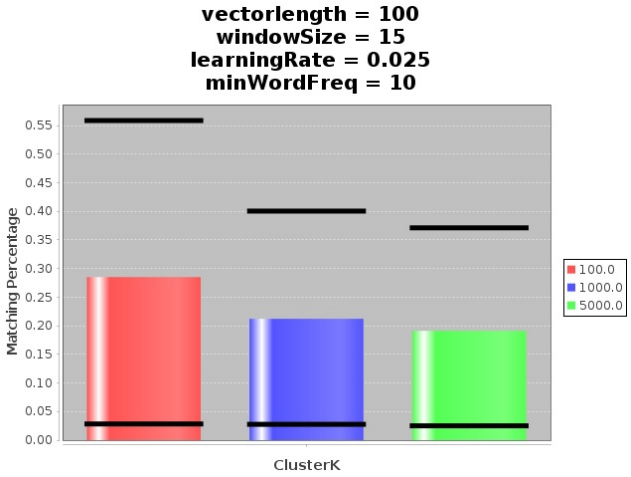


Figure 3.2: Mapping percentage with the vectorlength parameter of generalized Word2Vec approach in experiment 1.

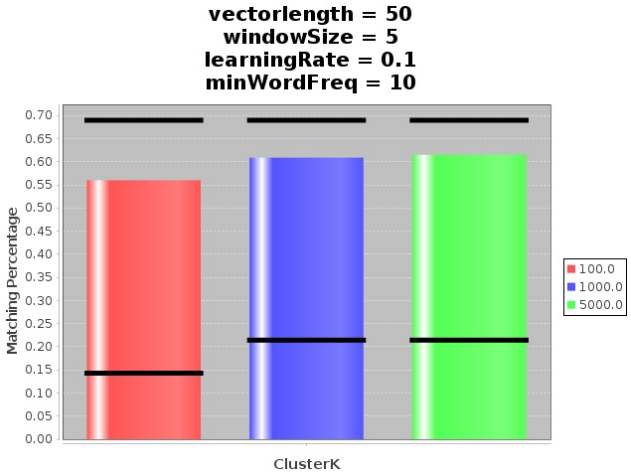


Figure 3.3: Mapping percentage with the vectorlength parameter of generalized Word2Vec approach in experiment 2.

the size at the start or end of the sequence. The Danish paper uses sequences of length 4, so this could also imply that a window size of 5 is already enough to cover the relationships the Danish paper found and therefore a higher size will not have an impact on the matching percentage.

Learning Rate

Both in experiment 1 and experiment 2, we found that the learning rate had no substantial influence on the matching percentage. See figure 3.11 for the matching percentage of experiment 1 and figure 3.9 for experiment 2. Similar results are

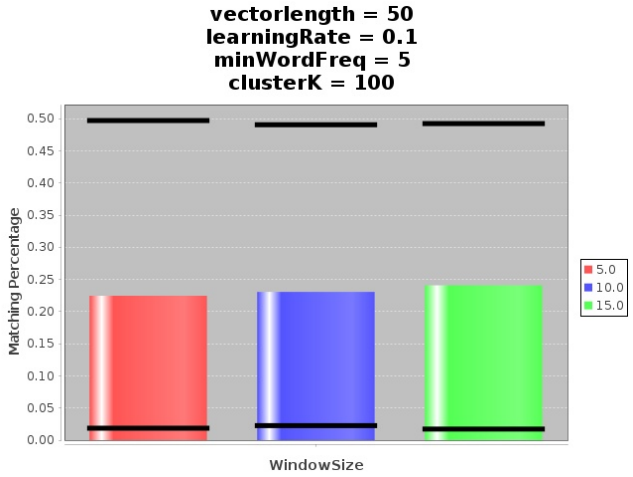


FIGURE 3.4: Mapping percentage with the window size parameter of generalized Word2Vec approach in experiment 1.

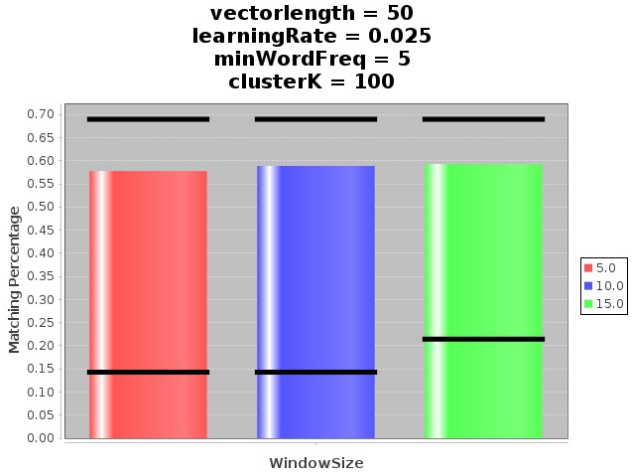


FIGURE 3.5: Mapping percentage with the window size parameter of generalized Word2Vec approach in experiment 2.

achieved with all the other parameter settings.

Learning rate is a parameter which is hard to tune. This explains the amount of literature around learning rate and the implementation of automatic tuning algorithms for learning rate. Therefore our results do not provide any possible conclusion as we only tested 2 different learning rates.
Although it is also possible that we chose our learning rate high enough in both cases and because of the learning decay, they both achieved the same results.

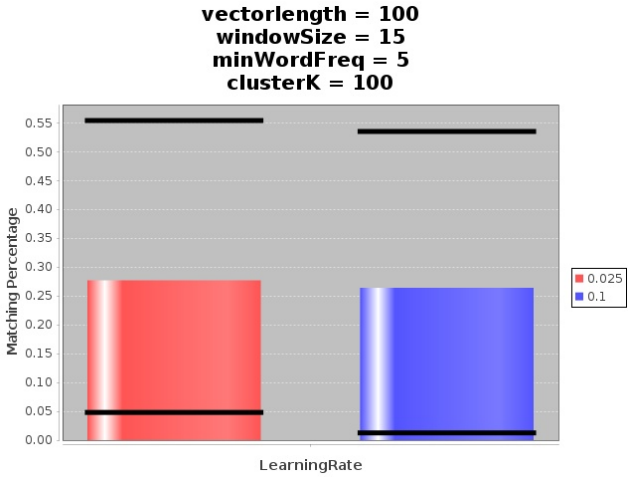


Figure 3.6: Mapping percentage with the learning rate parameter of generalized Word2Vec approach in experiment 1.

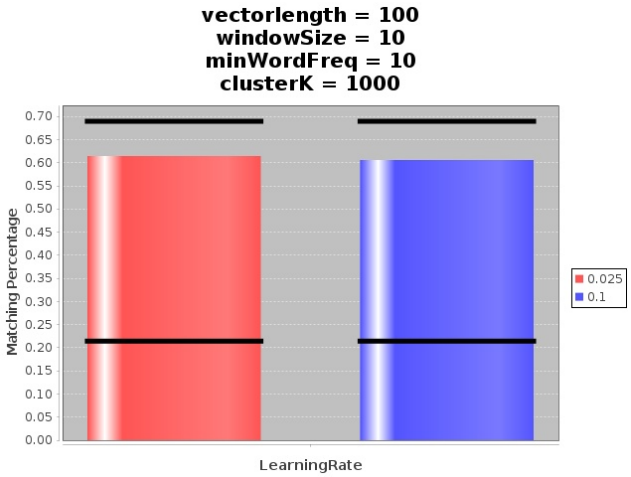


Figure 3.7: Mapping percentage with the learning rate parameter of generalized Word2Vec approach in experiment 2.

Minimum Word Frequency

Both in experiment 1 and experiment 2, we found that the minimum word frequency had no substantial influence on the matching percentage. See figure 3.11 for the matching percentage of experiment 1 and figure 3.9 for experiment 2. Similar results are achieved with all the other parameter settings.

We expected the minimum word frequency to have a better accuracy with a higher value as we would be sure they have enough instances to be learned from. Because we applied generalization, 99.82% of the instances are above the minimum

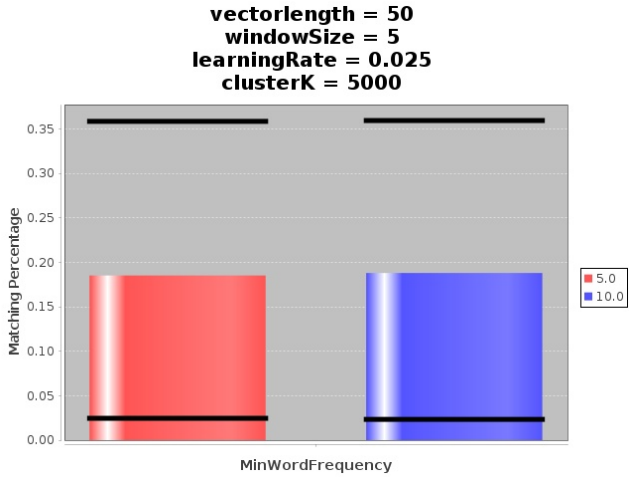


FIGURE 3.8: Mapping percentage with the minimum word frequency parameter of generalized Word2Vec approach in experiment 1.

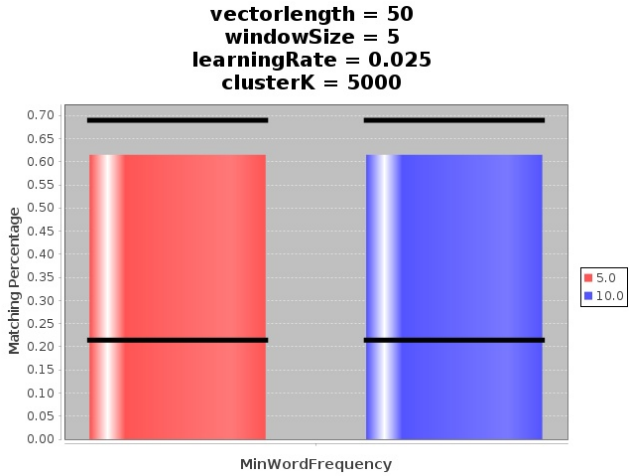


FIGURE 3.9: Mapping percentage with the minimum word frequency parameter of generalized Word2Vec approach in experiment 2.

word frequency of 10 to 69.90% before generalization. We therefore expect the minimum word frequency to have lost its influence. A future direction to explore is to ignore the generalization and solely work with minimum word frequency.

ClusterK

See figure 3.11 for the influence in the matching percentage of experiment 1 by the parameter clusterK. Similar results are achieved with all the other parameter settings.

We see a trend that with a lower clusterK there is a higher matching percentage.

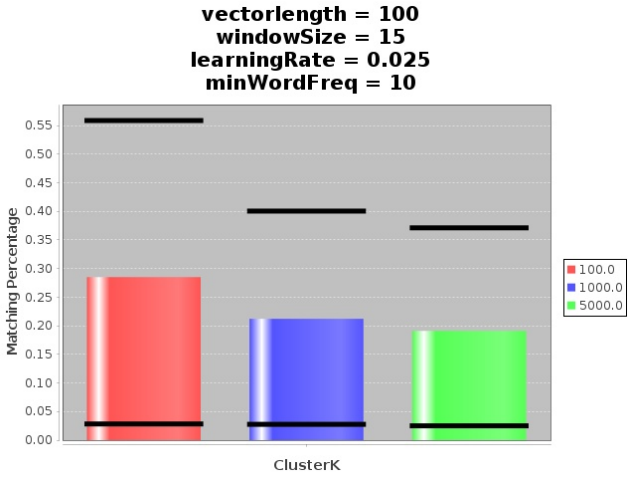


Figure 3.10: Mapping percentage with the clusterK parameter of generalized Word2Vec approach in experiment 1.

This means that in the smaller clusters, there is a higher density of instances which correspond to the Danish clusters. This shows that our method is not completely random especially in the case of some disease codes as a maximum matching percentage of 55% is found.

See figure 3.11 for the influence in the matching percentage of experiment 2 by the parameter clusterK. Similar results are achieved with all the other parameter settings.

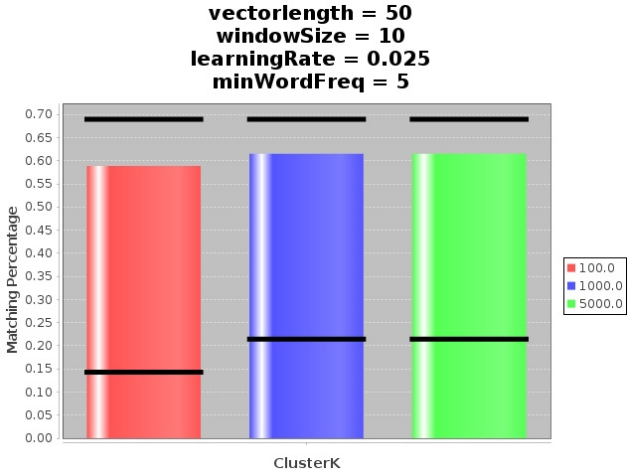


Figure 3.11: Mapping percentage with the clusterK parameter of generalized Word2Vec approach in experiment 2.

We see that most matches with the Danish clusters are found in the smaller Word2Vec

clusters. This corresponds with the results from experiment 1.

3.5.4 K-Nearest Neighbors Word2Vec

Vectorlength

Similar as generalized word2vec

Window Size

Similar as generalized word2vec

Learning Rate

Similar as generalized word2vec

Minimum Word Frequency

Similar as generalized word2vec

ClusterK

K

TODO

3.5.5 DeepWalk

TODO

Vectorlength

Similar as generalized word2vec

Window Size

Similar as generalized word2vec

Learning Rate

Similar as generalized word2vec

Minimum Word Frequency

Similar as generalized word2vec

ClusterK

TODO

Walklength

Brunak 4

3.5.6 Model Comparison

TODO

3.6 Conclusion

Chapter 4

Conclusion

This thesis started with explaining the new research area of Electronic Health Record Analytics. We explored the possible impact of this area as it allows to find medical patterns on a large scale. Those patterns range from drug discovery, disease progression for individuals, and reducing medical costs. At the moment several research groups are working on utilizing EHRs to find medical patterns using several methods like querying, statistics, data mining, and artificial intelligence approaches.
Our research sought to explore the usage of new machine learning approaches to find correlations between different diagnoses. The correlations found can be used in combination with prediction or classification methods.

We introduced a new way on how to apply Word2Vec methods by explaining the link between sentences of words and sequence of medical records. We call this approach a generalized Word2Vec approach and it can be applied on medical data to the correlations between different diagnoses.
To make sure the generalized Word2Vec methods can be applied to large-scale medical data, we applied the generalization concept also on DeepWalk. DeepWalk makes it possible to generate a smaller dataset from the original dataset and then apply a Word2Vec approach on this smaller dataset.
Besides the exploration on generalizing Word2Vec approaches, we also improved Word2Vec by tackling on of its shortcomings. This shortcoming of Word2Vec is that it is unable to handle unseen instances once it has built his lookup table. We combine a k-nearest neighbors method with Word2Vec and make an estimation of the correlation to other diagnoses for the unseen instance.

EXPERIMENTS + Limitations of the experiment
CONCLUSION

For more information about another possible experiment which also combines the Word2Vec approaches with prediction or classification methods, we refer to the next chapter. In this chapter we also talk about some possible improvements and future directions to explore.

Chapter 5

Future Work

5.1 Introduction

In this chapter we discuss some possible improvements or further research which are possible to add to the current approaches.
In section 5.2 we discuss a possible improvement on how we handled generalization of the medical data. In section 5.3, we introduce a optimization which can be applied on our Word2Vec approaches by utilizing data distribution. In section 5.4, we explain a specific neural network which can use the found patterns by the Word2Vec approaches to predict or classify patients' disease trajectories.

5.2 Generalization

In our Word2Vec approaches, we applied generalization on the medical states to reduce the amount of infrequent states. This was needed to retrieve more general n-grams. For this generalization, we divided some attributes into specific intervals.

Instead of dividing some attributes into specific intervals, we could apply normalization to it. Based on the distribution of the data, we can make more sensible intervals and assign them to the attributes.

5.3 Distributed Word2Vec

Word2vec can be made distributed as the underlying idea is quite simplistic, it counts occurrences of n-grams. Counting occurrences based on labels, is a well known problem and is often solved by MapReduce algorithms [17].

5.4 Patient Classification

As mentioned in section 2.3, a trained 2-layer neural network can be placed before another neural network and function as a lookup table. In this section, we discuss a

possible neural network which allows us to further investigate the effectiveness of our Word2Vec approach to better classify patients. More concrete: we should check if a better accuracy is acquired with the lookup table in front of the neural network or without.

5.4.1 Problem Definition

The medical history of a patient is seen as a time series with as datapoints EHRs. Based on the time series, we want to classify it into different disease trajectories. A patient who is classified into a specific disease trajectory, can be treated more specifically as we would have a better view what the next stages of the trajectory could be.

The medical data of multiple patients is a 3 dimensional tensor, see figure 5.1. This data structure is the input structure for a neural network.

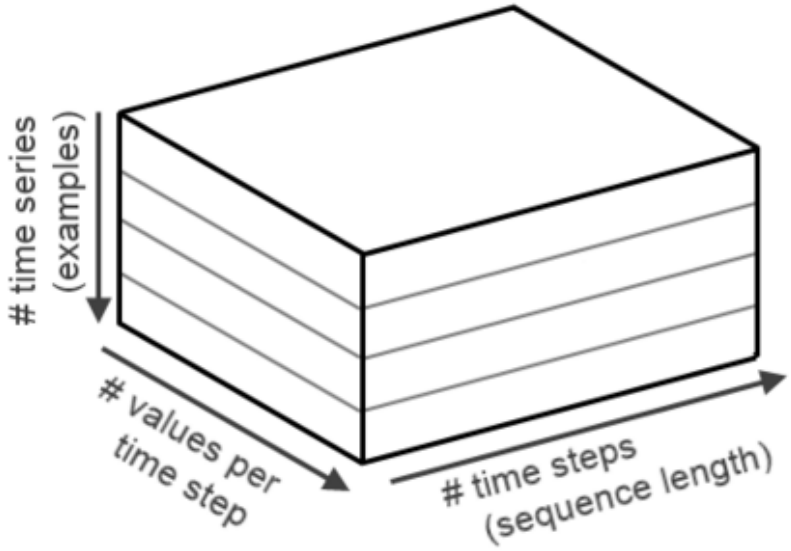


Figure 5.1: Overview of the data structure for medical data with a time aspect [6]

Medical data has some problems which we will discuss.
It often consists of long time periods. This means there could be a long range of dependencies between events. In the context of training neural networks, this can cause a problem known as the vanishing gradient problem [47].
Patients do not have regular intervals in their medical data. The irregular intervals need to be transformed into regular intervals otherwise the time aspect will not be consistent throughout the data.
The standardization of the attributes needs to be taken into account. Preferably some sort of normalization should be applied as well.

Medical data has a high dimensionality. A lot of parameters need to be taken into account to retrieve accurate results. This causes a well known problem: Curse of Dimensionality [32]. It causes the data to be sparse and have more infrequent cases. Therefore, more data is needed to cover all cases. Especially in medical data where outliers are important.

5.4.2 Approach

Here we describe our approaches for the problems mentioned in the previous section. We solve the vanishing gradient problem with a special form of recurrent neural network, see section 5.4.3.
By applying our Word2Vec approach, the input is projected on another vector space using a lookup table. This vector space causes normalization. The standardization is also done in our Word2Vec approach.
As we mentioned, the Curse of Dimensionality causes the need for more data. The neural network in section 5.4.3 often handles high dimensional data [14] [38] [51] [39]. In a sense, because it keeps track of the time aspect of the data, it uses the data more thoroughly and thus handles the high dimensionality better.

A lot of these problems are covered in an extensive work of Graves et al [22] about using advanced neural networks to label sequences.

Padding and Masking

The transformation of irregular intervals into a regular ones, is done with padding and masking.

If we do not use any masking and padding, our data can only be of equal length sequences and at each time step a classification label. Our data on the other hand, consists of several different length sequences and only has one label for each sequence, namely the classification of the whole sequence.

The method of padding is simple. It adds empty events (ex. zeros) to the shorter sequences until all sequences are of equal length for both input and output.
Using padding, changes the data quite drastically and would cause problems during the training because of that. For this problem we use the method of masking. With masking, we have two additional arrays which contain the information about whether an input or output is padding or not. See figure 5.2, picture 2 shows how the masking is done for a many to one case (which is the case that corresponds to labeling a complete sequence).

5.4.3 Neural Network

We still have to solve 2 problems mentioned in section 5.4.1.
In this section we explain in more detail why Long-Short Term Memory (LSTM)

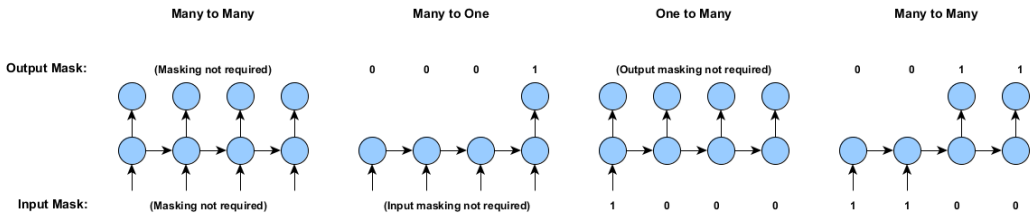


Figure 5.2: Multiple masking methods [6]

[26] networks handle the vanishing problem and long-term dependencies.

First we shortly repeat the structure of a neural network in figure 5.3. We see the input x, different layers with their perceptrons, σ as the activation function, and y as output.

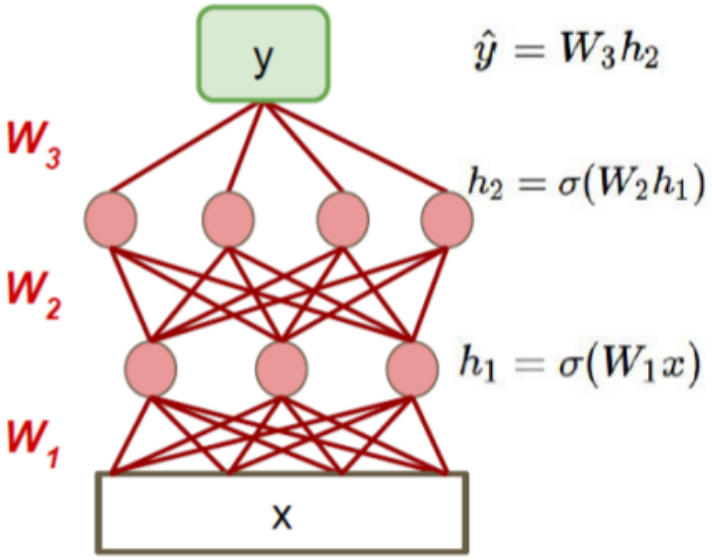


Figure 5.3: General structure of a neural network [53]

Recurrent Neural Network

A standard neural network does not have any persistence. They will classify their input but when they get a stream of inputs (ex. speech), they will classify each word independently of each other and without any regards of the previous words. A recurrent neural network (RNN) addresses this problem by introducing networks with loops [37]. This way, the output of a previous input has effect on the next input. In figure 5.4, we transform those loops into multiple copies of the same network

which makes it easier to reason about.

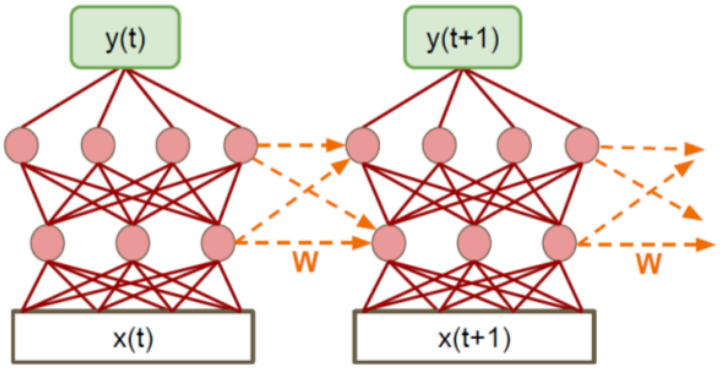


FIGURE 5.4: Unrolled recurrent neural network [53]

The problem with RNNs is mainly that they have trouble learning long-term dependencies which is often essential in time series [12].

Long Short Term Memory

A LSMT network is a specific RNN which is capable of learning long-term dependencies [19]. We will explain the difference with a standard RNN and why a LSTM can learn these long-term dependencies.

A recurrent network is, as we said, a chain of connected neural networks. Those networks can have a simple structure like a single *tanh* layer, see figure 5.5.

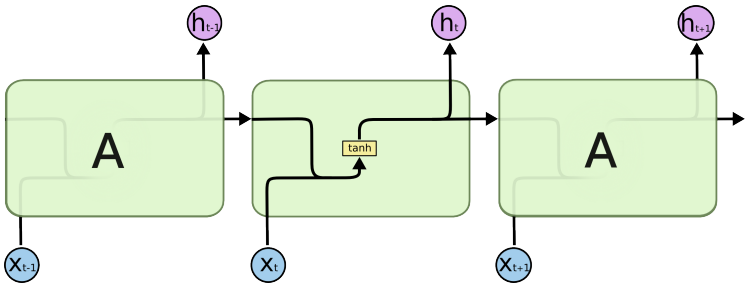


FIGURE 5.5: Unrolled recurrent neural network with a single tanh layer [45]

It is important to see the difference with a LSTM. The repeating network does not have a single neural network layer, but has 4 layers which each fulfills a specific goal, see figure 5.6.

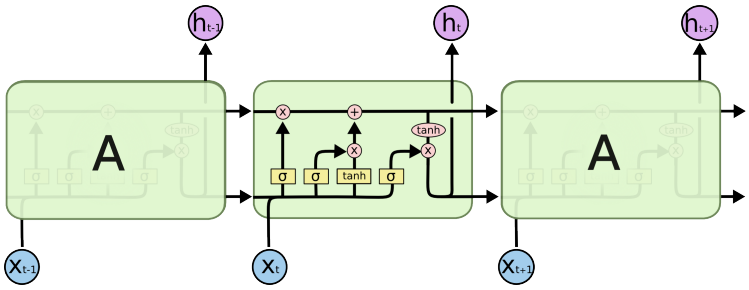


Figure 5.6: Unrolled LSTM network where each network has 4 layers [45]

The main idea behind LSTM is that each repeated network has its own cell state. It functions as a memory which can be updated with each new input. On figure 5.7, you can see the cell state C through time. It can be compared with a conveyor belt which interacts with the input at certain gates. This way the state is updated throughout several inputs and this way a LSTM can keep track of long-term dependencies.

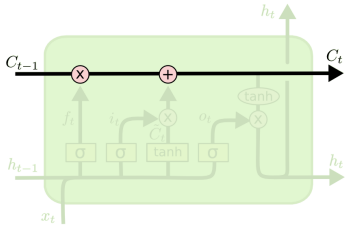
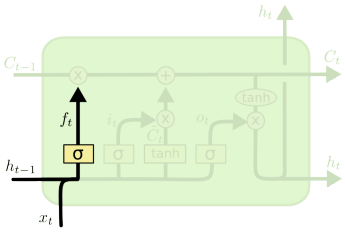


Figure 5.7: Representation of the cell state for a LSTM network [45]

In the following figures, we show the different gates and their functions in changing the cell state depending on the input and the output of the previous network. Next to each figure, the formulas are shown on how the cell state is updated. There should be no surprises as they are not much different than the standard formulas of neural networks.

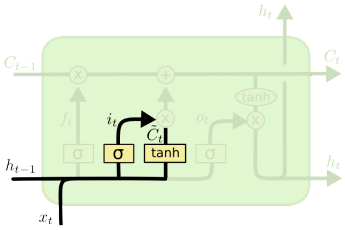
We start with the forget gate layer of a LSTM. Based on x_t and h_{t-1}, it outputs a number between 0 and 1 for each number in the cell state C_{t-1}. This is shown in figure 5.8. When the output is a 0, the number in the cell state is completely erased. With the output 1, the number does not change.

Next we look to the input gate layer. This gate decides which values will be updated in the cell state and outputs those in i_t. It is then combined with the vector $\widetilde{C}_t$, which contains the new candidate values based on the input x_t and h_{t-1}. This is shown in figure 5.9.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

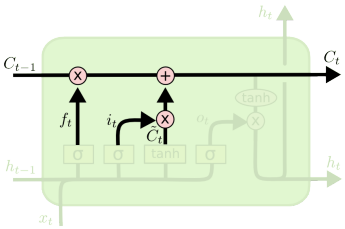
FIGURE 5.8: Forget layer of a LSTM network [45]



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

FIGURE 5.9: Input layer of a LSTM network [45]

We can now combine the previous results and adjust the cell state. We multiply the old state with f_t so we forget the needed elements. Then we add $i_t * \widetilde{C}_t$ which are the new candidate values multiplied by the amount on how much we want to update each state value. See figure 5.10.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

FIGURE 5.10: Update process of the cell state of a LSTM network [45]

Finally, we need to output h_t to the next network. This is based on the input and the cell state C_t. See figure 5.11.

Variants Long Short Term Memory

In "LSTM: A Search Space Odyssey" [23], research is done between different variants of LSTM networks. It was concluded that the forget gate and the activation function are the most important. Other variants do not have a large influence and mainly add a lot of extra complexity.

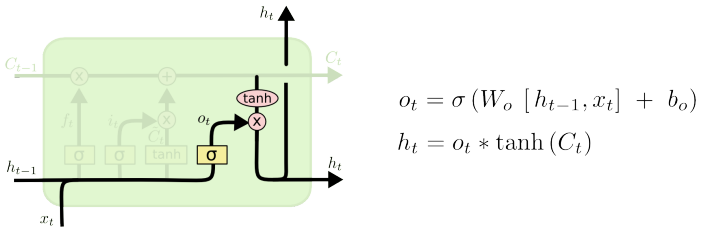


Figure 5.11: Decide the output of a LSTM network [45]

5.5 Conclusion

Before this chapter, we already mentioned some possible roads to explore like a better disease code mapping strategy or extensive parameter tuning.

In this chapter we talked shortly about two possible improvements, a better generalization approach and a better use of data distribution to speed up to process. We mainly focused on LSTM neural networks and explained why they should be suitable to predict or classify disease trajectories. They are able to handle the Curse of Dimensionality, long time dependencies, and take the time aspect of medical data into account. Our Word2Vec approaches can be used in combination with this kind of neural network. The prediction/classification accuracy of the LSMT neural network makes it possible to validate the working of our approaches by testing if the accuracy increases with the use of our Word2Vec approaches.

Bibliography

[1] Google Code Archive - Long-term storage for Google Code Project Hosting. https://code.google.com/archive/p/word2vec/. (Accessed on 05/16/2016).

[2] HealthIT.gov | the official site for Health IT information. https://www.healthit.gov/. (Accessed on 05/15/2016).

[3] MedDRA. http://www.meddra.org/. (Accessed on 05/03/2016).

[4] The Speech Recognition Wiki. http://recognize-speech.com/. (Accessed on 05/16/2016).

[5] THIN research team UCL. https://www.ucl.ac.uk/pcph/research-groups-themes/thin-pub. (Accessed on 05/27/2016).

[6] Using Recurrent Neural Networks in DL4J - Deeplearning4j: Open-source, distributed deep learning for the JVM. http://deeplearning4j.org/usingrnns. (Accessed on 05/21/2016).

[7] WHO | International Classification of Diseases (ICD). http://www.who.int/classifications/icd/en/. (Accessed on 04/27/2016).

[8] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[9] G. V. Bard. Spelling-error Tolerant, Order-independent Pass-phrases via the Damerau-levenshtein String-edit Distance Metric. In *Proceedings of the Fifth Australasian Symposium on ACSW Frontiers - Volume 68*, ACSW '07, pages 117–124, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.

[10] M. Bashiri and A. F. Geranmayeh. Tuning the parameters of an artificial neural network using central composite design and genetic algorithm. *Scientia Iranica*, 18(6):1600 – 1608, 2011.

[11] J. Beel, B. Gipp, S. Langer, and C. Breitinger. Research Paper Recommender Systems: A Literature Survey. *International Journal on Digital Libraries*, pages 1–34, 2015.

[12] Y. Bengio, P. Simard, and P. Frasconi. Learning Long-term Dependencies with Gradient Descent is Difficult. *Trans. Neur. Netw.*, 5(2):157–166, Mar. 1994.

[13] C. Bennett and T. Doub. Data Mining and Electronic Health Records: Selecting Optimal Clinical Treatments in Practice. *CoRR*, abs/1112.1668, 2011.

[14] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. *ArXiv e-prints*, June 2012.

[15] A. G. Chris Nicholson. Using Recurrent Neural Networks in DL4J - Deeplearning4j: Open-source, distributed deep learning for the JVM. http://deeplearning4j.org/usingrnns. (Accessed on 05/03/2016).

[16] T. M. Cover. Nearest neighbor pattern classification. 1982.

[17] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.

[18] G. Declerck, J. Souvignet, J. M. Rodrigues, and M. Jaulent. Automatic annotation of ICD-to-MedDRA mappings with SKOS predicates. In *MIE*, volume 205 of *Studies in Health Technology and Informatics*, pages 1013–1017. IOS Press, 2014.

[19] F. Gers. Long Short-Term Memory in Recurrent Neural Networks, 2001.

[20] C. L. Giles, S. Lawrence, and A. C. Tsoi. Noisy Time Series Prediction Using Recurrent Neural Networks and Grammatical Inference. *Mach. Learn.*, 44(1-2):161–183, July 2001.

[21] Y. Goldberg and O. Levy. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722, 2014.

[22] A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer, 2012.

[23] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber. LSTM: A Search Space Odyssey. *CoRR*, abs/1503.04069, 2015.

[24] D. Guthrie, B. Allison, W. Liu, L. Guthrie, and Y. Wilks. A Closer Look at Skip-gram Modelling.

[25] A. Hameurlain, J. Kung, R. Wagner, H. Decker, L. Lhotska, and S. Link, editors. *Transactions on Large-Scale Data- and Knowledge-Centered Systems IV - Special Issue on Database Systems for Biomedical Applications*, volume 8980 of *Lecture Notes in Computer Science*. Springer, 2011.

[26] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[27] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251 – 257, 1991.

[28] K. J. M. Janssen, A. R. T. Donders, F. E. J. Harrell, Y. Vergouwe, Q. Chen, D. E. Grobbee, and K. G. M. Moons. Missing covariate data in medical research: To impute is better than to ignore. *Journal of Clinical Epidemiology*, 63(7):721–727, 2016.

[29] A. B. Jensen, P. L. Moseley, T. I. Oprea, S. G. Ellesoe, R. Eriksson, H. Schmock, P. B. Jensen, L. J. Jensen, and S. Brunak. Temporal disease trajectories condensed from population-wide registry data covering 6.2 million patients. *Nat Commun*, 5, Jun 2014. Article.

[30] C. K. R. Jimeng Sun. Big Data Analytics for Healthcare. 2013.

[31] A. Karpathy. The Unreasonable Effectiveness of Recurrent Neural Networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/. (Accessed on 05/03/2016).

[32] E. Keogh and A. Mueen. *Encyclopedia of Machine Learning*, chapter Curse of Dimensionality, pages 257–258. Springer US, Boston, MA, 2010.

[33] S. Kimura, T. Sato, S. Ikeda, M. Noda, and T. Nakayama. Development of a Database of Health Insurance Claims: Standardization of Disease Classifications and Anonymous Record Linkage. *J Epidemiol*, 20(5):413–419, Sep 2010. 20699602[pmid].

[34] W. M. Koolen, T. van Erven, and P. Grünwald. Learning the Learning Rate for Prediction with Expert Advice. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2294–2302. Curran Associates, Inc., 2014.

[35] O. Levy and Y. Goldberg. Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 302–308, 2014.

[36] O. Levy and Y. Goldberg. Neural Word Embedding as Implicit Matrix Factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc., 2014.

[37] Z. C. Lipton. A Critical Review of Recurrent Neural Networks for Sequence Learning. *CoRR*, abs/1506.00019, 2015.

[38] Z. C. Lipton, D. C. Kale, and R. C. Wetzel. Phenotyping of Clinical Time Series with LSTM Recurrent Neural Networks. *CoRR*, abs/1510.07641, 2015.

[39] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, 54:187 – 197, 2015.

[40] C. Miani et al. Health and Healthcare: Assessing the Real World Data Policy Landscape in Europe. 2014.

[41] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. *CoRR*, abs/1310.4546, 2013.

[42] A. Moores. Efficient Memory-based Learning for Robot Control. 1991.

[43] M. Nielsen. Neural networks and deep learning. http://neuralnetworksanddeeplearning.com/. (Accessed on 05/03/2016).

[44] C. Olah. Deep Learning, NLP, and Representations - colah's blog. http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/. (Accessed on 05/16/2016).

[45] C. Olah. Understanding LSTM Networks – colah's blog. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. (Accessed on 05/03/2016).

[46] O. M. O. Partnership. OSIM2 - Observational Medical Dataset Simulator Generation 2 | Observational Medical Outcomes Partnership. http://omop.org/OSIM2. (Accessed on 05/28/2016).

[47] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training Recurrent Neural Networks. *CoRR*, abs/1211.5063, 2012.

[48] B. Perozzi, R. Al-Rfou, and S. Skiena. DeepWalk: Online Learning of Social Representations. *CoRR*, abs/1403.6652, 2014.

[49] X. Rong. word2vec Parameter Learning Explained. *CoRR*, abs/1411.2738, 2014.

[50] F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. *Psychological Review*, pages 65–386, 1958.

[51] H. Sak, A. W. Senior, and F. Beaufays. Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition. *CoRR*, abs/1402.1128, 2014.

[52] J. Schmidhuber. Deep Learning in Neural Networks: An Overview. *Neural Networks*, 61:85–117, 2015. Published online 2014; based on TR arXiv:1404.7828 [cs.NE].

[53] J. Simm. IMEC Technical talk.

[54] R. F. Sproull. Refinements to nearest-neighbor searching in k-dimensional trees. *Algorithmica*, 6(1):579–589, 1991.

[55] C. P. Stone. A Glimpse at EHR Implementation Around the World: The Lessons the US Can Learn. 2014.

[56] J. Sun, F. Wang, J. Hu, and S. Edabollahi. Supervised Patient Similarity Measure of Heterogeneous Patient Records. *SIGKDD Explor. Newsl.*, 14(1):16–24, Dec. 2012.

[57] D. D. Team. Deeplearning4j: Open-source distributed deep learning for the JVM.

[58] G. B. Team. Vector Representations of Words. https://www.tensorflow.org/versions/0.6.0/tutorials/word2vec/index.html. (Accessed on 05/16/2016).

[59] F. Wang, N. Lee, J. Hu, J. Sun, and S. Ebadollahi. Towards Heterogeneous Temporal Clinical Event Pattern Discovery: A Convolutional Approach. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 453–461, New York, NY, USA, 2012. ACM.

Master thesis filing card

Student: Milan van der Meer

Title: Learning a Diseases Embedding using Generalized Word2Vec Approaches.

UDC: 621.3

Abstract:

Here comes a very short abstract, containing no more than 500 words. LaTeX commands can be used here. Blank lines (or the command `\par`) are not allowed!

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Thesis submitted for the degree of Master of Science in Engineering: Computer Science, specialisation Artificial Intelligence

Thesis supervisor: Prof. dr. R. Wuyts

Assessors: Prof. dr. ir. H. Blockeel
R. van Lon

Mentor: Dr. E. D'Hondt