KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Learning a Disease 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

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.

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
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

Symbols

42	"The Answer to the Ultimate Question of Life, the Universe, and Everything" according to [**?**]
c	Speed of light
E	Energy
m	Mass
π	The number pi

Chapter 1

Introduction

The first contains a general introduction to the work. The goals are defined and the modus operandi is explained.

1.1 Lorem Ipsum 4–5

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

1.2 Lorem Ipsum 6–7

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Chapter 2

Electronic Health Records Analytics

2.1 Introduction

In this chapter we explain the context in which we will be working for this thesis.

In section 2.2 we explain what electronic health records are and how these are represented. In section 2.3, we go further on how the electronic health records can be used to retrieve useful medical information. We also explain different approaches on how to retrieve this information from the health records.

2.2 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].

Large countries like the US and the UK, are each investing more than 20 billion dollars into EHR systems [47]. Those systems are adopted by around 70% of the physicians. Which means a large number of physicians are using other methods or systems. Also, each country develops his own system which results in a good nationwide coverage but introduces different system around the world. 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.

2.2.1 Disease Codes

To make EHRs practical it is important to adhere to standards for data formatting. A well documented 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.
A 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.

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) [6]. 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 numberic code where new terms are assigned sequentially. It does not provide clear hierarchical categories like ICD which are hard to understand without a medical background. MedDRA is mainly used by pharmaceutical companies.

2.3 EHR Analytics

EHRs provide a massive amount of data which could be used to create useful insights. The data contains the medical history of a patient including medical measurements, diagnoses, prescribed drugs, and demographics. 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
- 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.

In the following sections we talk about current EHR analytic methods.

2.3.1 Querying

Analytics in epidemiology on EHRs is typically done through querying a database [22]. A specialist can have a certain idea about correlations between conditions or patients. He can support this idea by finding 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 considered. Some complex relations cannot be found because of the limitations of the querying language. A query language is equivalent to first-order logic. Which means non-linear relations in the data cannot be found.

2.3.2 Big Data Analytics

More advanced methods are applied on EHRs than querying. In general, they try to find patterns in the EHR data which then can be used to predict outcomes of treatments [27].

Several predictive methods from machine learning can be used and show promising results [11]. Those results are achieved by using non-optimized methods which are applied on the EHR data. We also note that methods used as Multi-layer Perceptron networks are not ideal for prediction of time-series, see section 6.4. We conclude that there is still a lot of room for improvement.

More specialized approaches are also applied on EHR data [51] besides the above mentioned machine learning techniques. An EHR of a patient can be transformed into a matrix structure, see figure 2.1. On these matrix structures, large-scale data mining algorithms can be applied. Those make it possible to mine temporal patterns in EHR data. The found patterns can be used for prediction later on.

It is also possible to define patient similarities [48]. So when a patient is similar to a previous known case, his treatment can be based on those previous experiences. This is a similar approach as in recommender systems [**?**].

2.3.3 Statistical Analysis

A more statistical approach is used to find patterns in EHR data on a dataset of the Danish population [26].

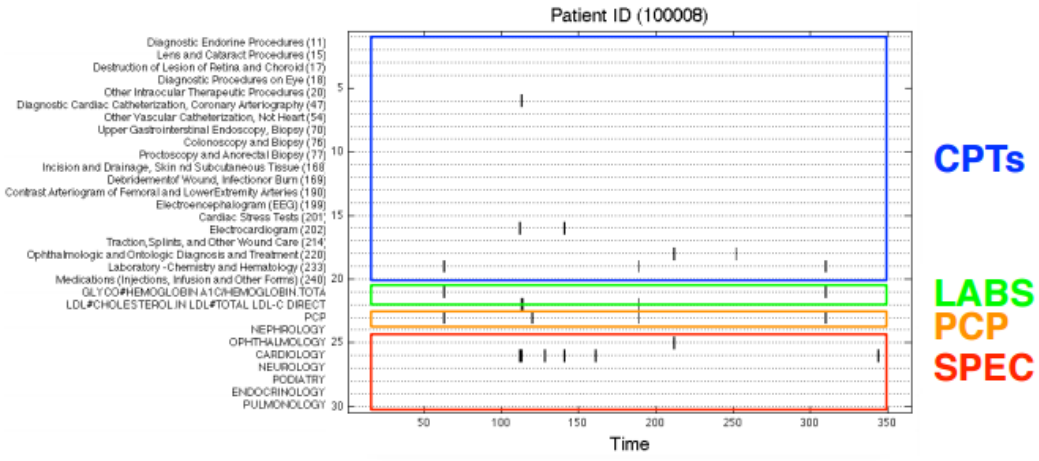


Figure 2.1: Example of an EHR transformed into a matrix structure [51]

First we describe the dataset. The dataset which is used to apply the statistical analysis on, are EHRs collected over 15 years on over 6 million patients in Denmark. The size of this dataset makes it possible to retrieve significant results.

They start with finding pairs of diagnoses which have a strong 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.

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.

The found clusters will be used to validate our approach described in chapter 3. You can find an example of a clustered trajectory in figure 2.2.

2.4 Conclusion

We conclude that EHRs contain important information of a patients medical history and current state. When a large amount of EHRs is available, empirical 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.

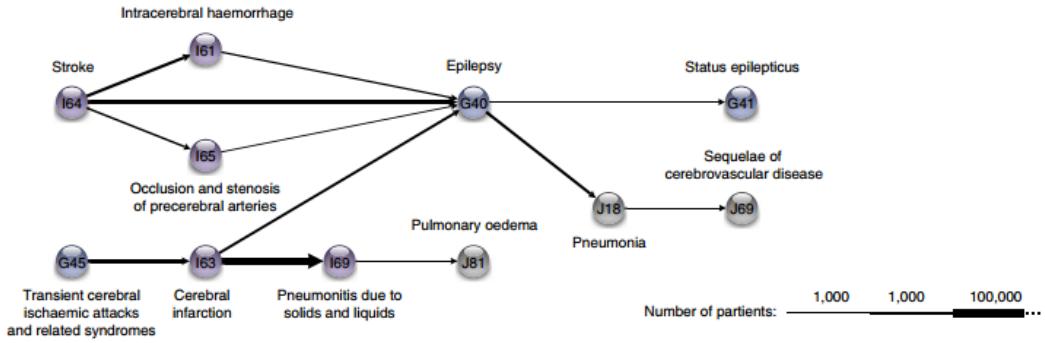


FIGURE 2.2: Cerebrovascular disease trajectory cluster for the Danish population [26]

In the next chapter we explain the needed 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 3

Generalized Word2Vec

3.1 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 2.

We start with explaining some background knowledge in section 3.2 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 3.3. This is then used to introduce Deepwalk as an extension on Word2Vec in section 3.4.

After introducing those concepts, we explain our generalized word2vec approaches in section 3.5.

3.2 Background Knowledge

3.2.1 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.

3.2.2 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.

3.2.3 K-nearest Neighbors

The k-Nearest Neighbors (knn) algorithm is a simple machine learning algorithm [14]. 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.

3.2.4 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) [46]. In this section we explain the workings of this approach in more detail [35].

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 3.1 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 $[3, 8]$.

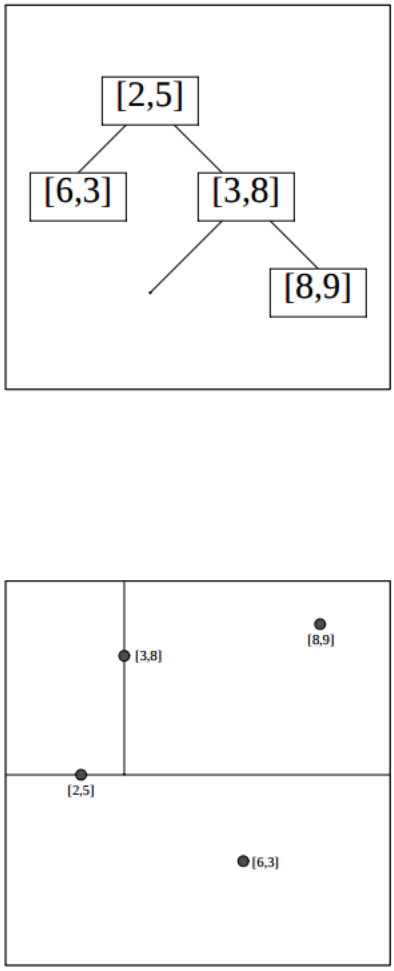


FIGURE 3.1: Representation on how a binary kd tree splits up the plane [35]

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.

3.2.5 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 [42]. A perceptron takes multiple binary inputs and has a single binary output (see figure 3.2). Each input has a corresponding real numbered weight w_j. The output is decided on the following equation:

$$output = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if if } \sum_j w_j x_j > \text{ threshold} \end{cases} \tag{3.1}$$

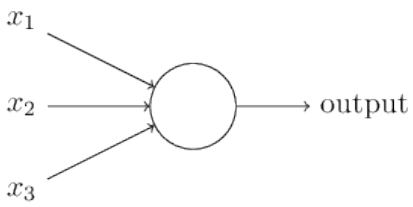


Figure 3.2: Simple presentation of a perceptron [36]

We can build a network by connecting multiple perceptrons (see figure 3.3). 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 [24].

Now we have seen how a general network is constructed, we look at some vocabulary.

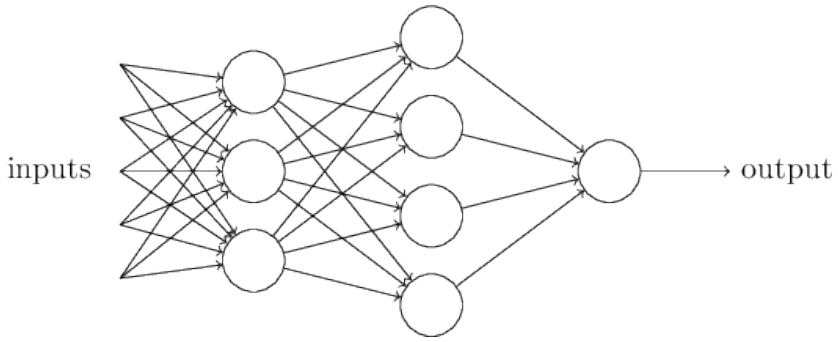


FIGURE 3.3: More complex network made by connecting multiple perceptrons [36]

In figure 3.4, 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.

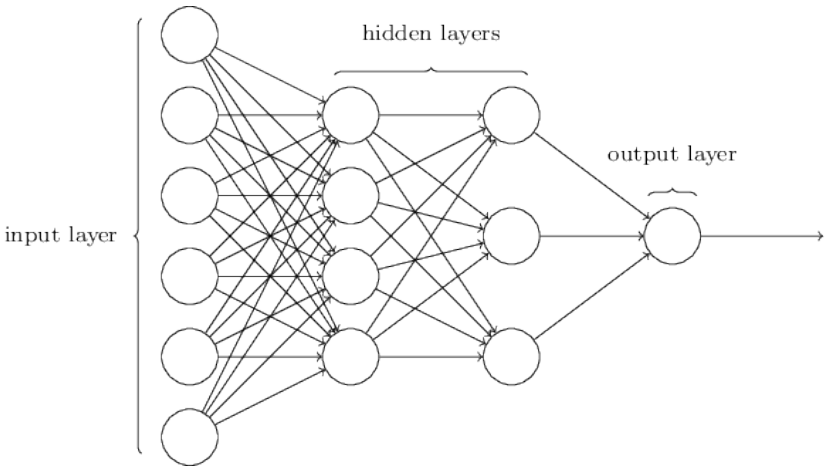


FIGURE 3.4: General vocabulary of a multilayer network [36]

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. 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. Note that the change in weights, should only effect the output by a small bit (see figure 3.5). 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.

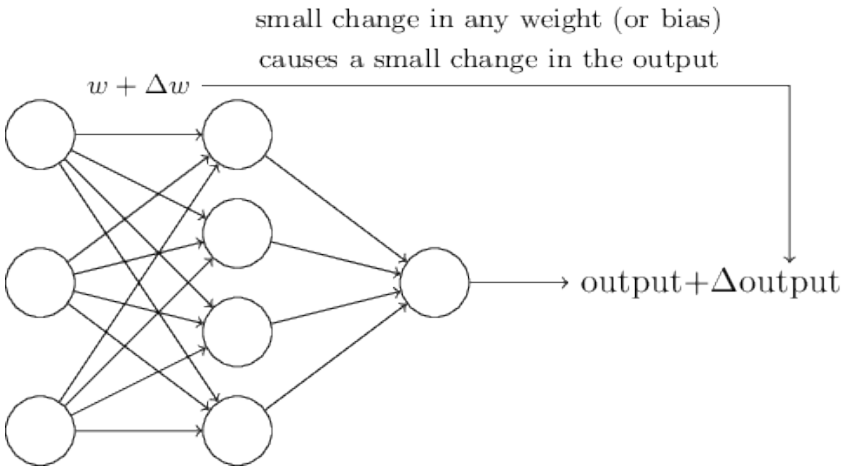


Figure 3.5: Small change on the weights, only has a small impact on the output [36]

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:

$$\frac{1}{1 + exp(-\sum_j w_j x_j - b)} \tag{3.2}$$

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

$$\Delta output \approx \sum_j \frac{\partial output}{\partial w_j} \Delta w_j + \frac{\partial output}{\partial b} \Delta b \tag{3.3}$$

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.

3.2.6 Backpropagation

Backpropagation is an algorithm which is used to train neural networks [44]. 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 3.6.

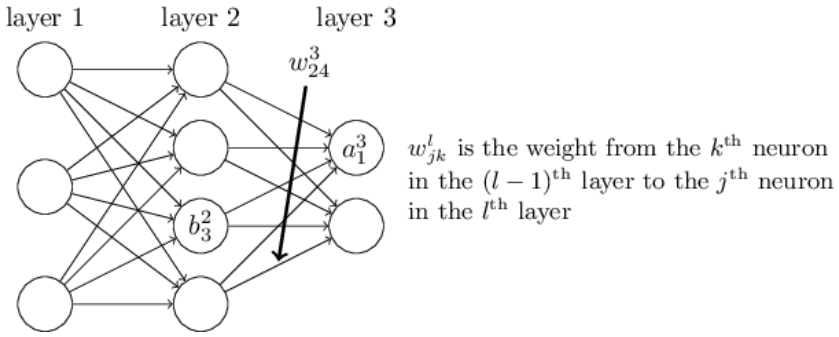


FIGURE 3.6: Visual representation of the terminology for a neural network [36]

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

$$a^l = \sigma(w^l a^{l-1} + b^l) \tag{3.4}$$

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 activation and on how much the activation function is influenced by z^L_j:

$$\delta^L_j = \frac{\partial C}{\partial a^L_j}\sigma'(z^L_j) \text{ with } z^L_j = \sum_k w^L_{jk} a^{L-1}_k + b^L_j \tag{3.5}$$

This can be written as a neat vector equation:

$$\delta^L = (a^L - y) \circ \sigma'(z_j^L) \tag{3.6}$$

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:

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \circ \sigma'(z_j^L) \tag{3.7}$$

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:

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l. \tag{3.8}$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l. \tag{3.9}$$

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.

3.3 Word2Vec

3.3.1 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.

3.3.2 Skip-gram

There are two main models used for word2vec [34], namely Continuous Bag-of-Words (CBOW) and the Skip-Gram model.
The first one tries to predict a word if a context is given (ex. predict Paris when capital France is given). And the second one does the inverse of this approach [41].

Empirical results have shown that the Skip-Gram model tends to do better on larger datasets [50] 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:

$$\arg\max_{\theta} \prod_{(w,c\in D} p(c|w;\theta) \tag{3.10}$$

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-gram [21]. In figure 3.7, n-gram is explained on the sentence "This is a sentence".

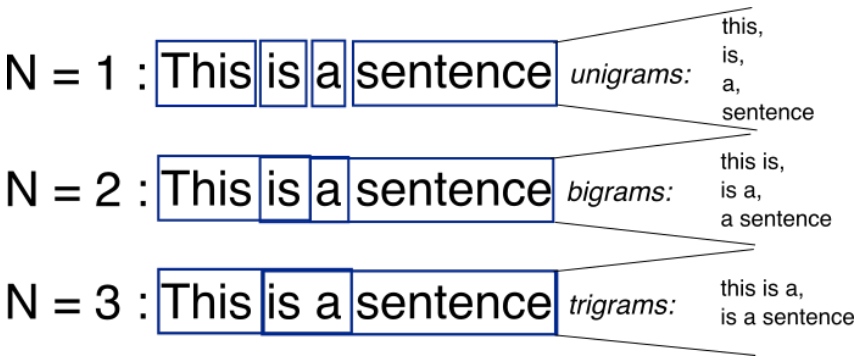


FIGURE 3.7: Explanation of n-gram [4]

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 results in more training examples and thus can lead to a higher accuracy but will have a longer training time.

Parameterization

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

$$p(c|w;\theta) = \frac{e^{v_x * v_w}}{\sum_{c' \in C} e^{v_{c'} * v_w}} \tag{3.11}$$

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:

$$\arg\max_{\theta} \sum_{(w,c)\in D} \log p(c|w;\theta) = \sum_{(w,c)\in D} (\log e^{v_c * v_w} - \log \sum_{c'} e^{v_{c'} * v_w}) \tag{3.12}$$

3.3.3 Negative Sampling

To compute the vectors using the Skip-gram model more efficiently, we introduce negative sampling [18].
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 of several vectors chosen randomly. This makes the Skip-gram model usable in terms of speed.

3.3.4 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 goal function the function described in the previous section. The training can be done with Gradient Descent for example.
The trained 2-layer neural network can be placed in front of another neural network [37]. It will convert the words to their vector representation and feed 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.

3.4 DeepWalk

DeepWalk is an approach where graph structured data is transformed into sequences of vertices [40]. 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 3.8, 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.

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 $\Phi \in \mathbb{R}^{|V| \times d}$
1: Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
2: Build a binary Tree T from V
3: **for** $i = 0$ to γ **do**
4: $\mathcal{O} = \text{Shuffle}(V)$
5: **for each** $v_i \in \mathcal{O}$ **do**
6: $\mathcal{W}_{v_i} = RandomWalk(G, v_i, t)$
7: SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
8: **end for**
9: **end for**

FIGURE 3.8: Overview of the DeepWalk algorithm [40]

Secondly, those vertices are used for word2vec. This process is explained in section 3.3.

3.5 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.

3.5.1 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 contains certain values depending on the EHRs. It can contain values for the timestamp, demographics, blood pressure, diagnoses, and others.

3.5.2 Generalized Word2Vec

As explained in section 3.3, 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 for 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.

3.5.3 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 3.2.3). 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.
With our approach, the knn are found using a kd tree (see section 3.2.4). Those knn are found from all the known vectors in their old representation v_{old} in the lookup table. From the new vector representations v_{new} of the knn, a weighted average is taken to find the v_{new} for $v_{unknown}$.

In short: we look for the knn of the $v_{unknown}$ from all the known vectors in their original representation v_{old}. Based on the found knn, we take an weighted average of the new representation v_{new} from those found knn. This weighted average is the v_{new} for the $v_{unknown}$.

3.5.4 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 data 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.
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.

3.6 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 object than words, namely vectors representing an EHR. We also extended Word2Vec to find a representation for a new instance based on knn. The same extension can be done for Deepwalk.

Chapter 4

Validation

4.1 Introduction

In this chapter
DiseaseMapping (generalization) OSIM Clusters TensorFlow DL4J

4.2 Dataset

To validate the approaches mentioned in chapter 3, we used a dataset generated by OSIM2. This dataset is used by 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. [26]. Although these found clusters are not a golden standard, it is a first validation point for our approaches.

4.3 Software

4.3.1 TensorFlow

TensorFlow is an open-source machine learning software library released at the end of 2015 [7]. 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.

4.3.2 DeepLearning4Java

DeepLearning4Java (DL4J) is an open-source machine learning software library released by Skymind [49].
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.

4.4 Experiment Setup

4.4.1 Generalization

As described in chapter 3, 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 4.4.2.

To see the effect of our generalization, see table 4.4.1. You can see the 3 most common vectors from our dataset before and after the generalization.

4.4.2 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

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

bruise on your left leg.

The OMOP dataset uses MedDRA disease codes for the diagnoses. We described the MedDRA disease codes in chapter 2. 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. [26] 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 4.1, 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.

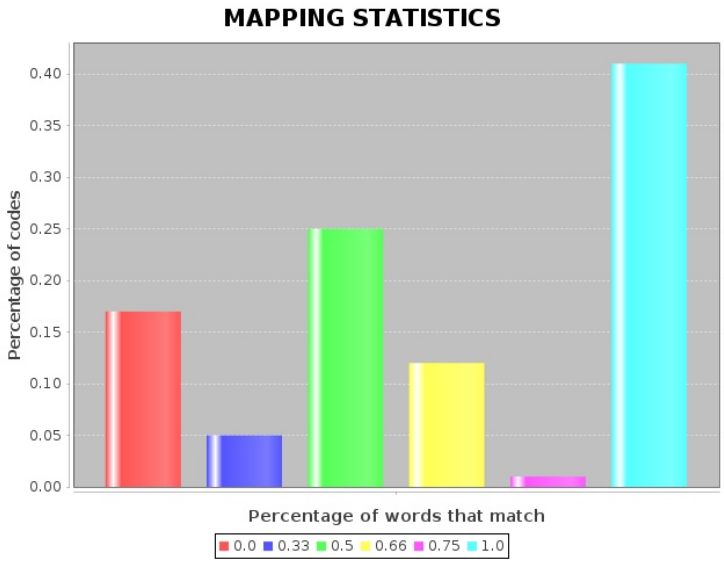


Figure 4.1: Statistics of the mapping approach

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 [8] 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.

4.5 Results

-what to test

4.6 Conclusion

http://omop.org/OSIM2

Chapter 5

Conclusion

The final chapter contains the overall conclusion. It also contains suggestions for future work and industrial applications.

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.

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.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In

hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Chapter 6

Future Work

6.1 Introduction

In this chapter
Normalization Distributed RNN

6.2 Generalization

In our word2vec approach we applied generalization on the medical 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.

6.3 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 [15].

6.4 Patient Classification

As mentioned in section 3.3, 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 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.

6.4.1 Problem Definition

The medical history of a patient is seen as a time series with as datapoints an EHR. 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.

The medical data of multiple patients is a 3 dimensional tensor, see figure 6.1. This data structure is the input structure for a neural network.

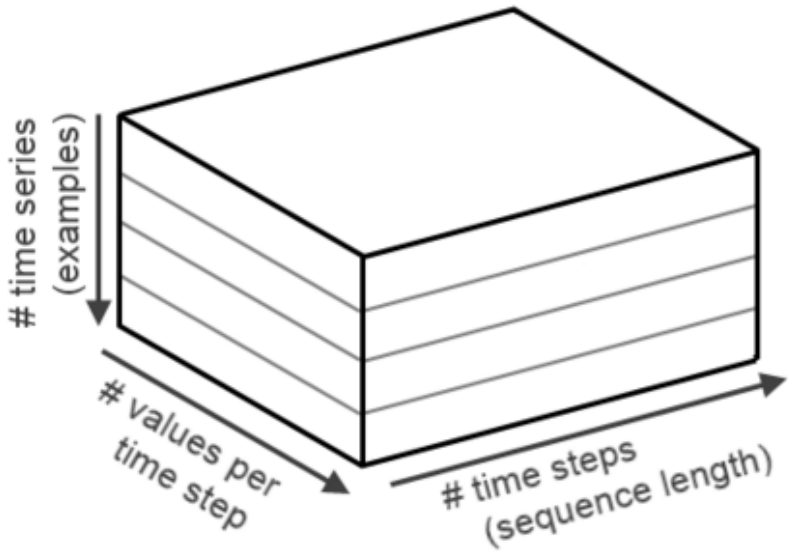


Figure 6.1: Overview of the data structure for medical data with a time aspect [5]

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 [39].
Patients do not have regular intervals in their medical data. The irregular intervals need to be transformed to regular intervals otherwise the time aspect won't 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 the well known problem: Curse of Dimensionality [29]. It causes the data to be sparse and therefore, more data is needed to cover all cases.. Especially in medical data where outliers are important.

6.4.2 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 6.4.3.
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 6.4.3 often handles high dimensional data [12] [32] [43] [33]. In a sense, because it keeps track of the time aspect of the data, it uses the data more thoroughly and thus has a better method to handle the high dimensionality.

A lot of these problems are covered in an extensive work [19] about using advanced neural networks to label sequences.

Padding and Masking

The transformation of the irregular interval to a regular one, is done with padding and masking.

If we do not use any masking and padding, our data can only be of equal length time series and at each time step a classification. Our data consists of several inputs, the different time steps of a time series, and has one output associated with it, namely the classification of the time series.

The method of padding is simply by adding empty events (ex. zeros) to the shorter time series until all examples are of equal length for both input and output. Using padding, changes the data quite drastically and would case 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 6.2, picture 2 on how the masking is done for a many to one case.

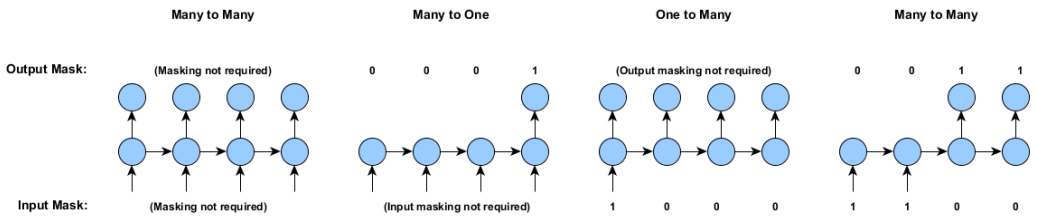


FIGURE 6.2: Multiple masking methods [5]

6.4.3 Neural Network

In this section we explain in more detail why a Long-Short Term Memory (LSTM) [23] networks handles the vanishing problem and long-term dependencies.

First we shortly repeat the structure of a neural network in figure 6.3. We see the input, different layers with their perceptrons, and σ as the activation function.

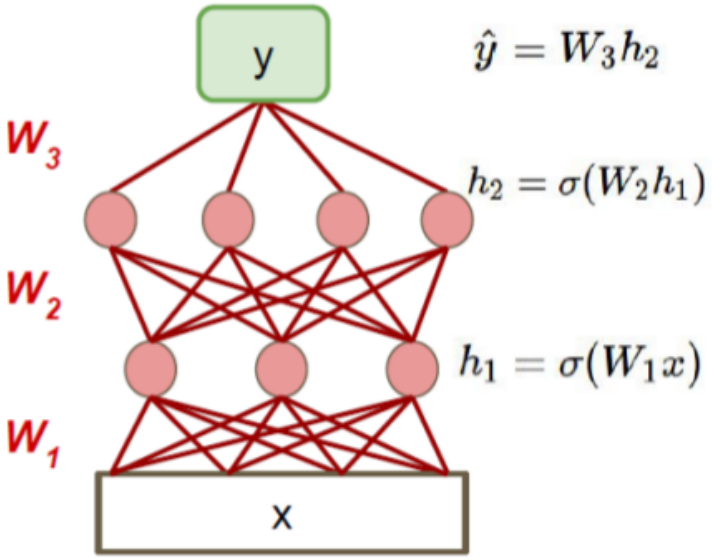


Figure 6.3: General structure of a neural network [45]

Recurrent Neural Network

A standard neural network do 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 [31]. This way, the output of a previous input has effect on the next input. In figure 6.4, we transform those loops into multiple copies of the same network which makes it easier to reason about.

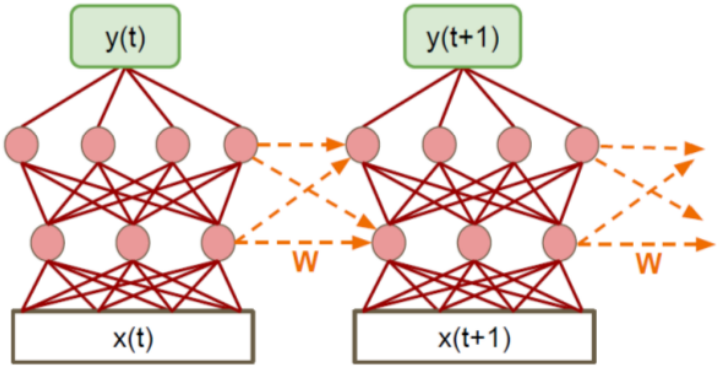


FIGURE 6.4: Unrolled recurrent neural network [45]

The problem with RNNs is mainly that they have troubles learning long-term dependencies which is often essential in time series [10].

Long Short Term Memory

A LSMT network is a specific RNN which is capable of learning long-term dependencies [16]. 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 as a single *tanh* layer, see figure 6.5.

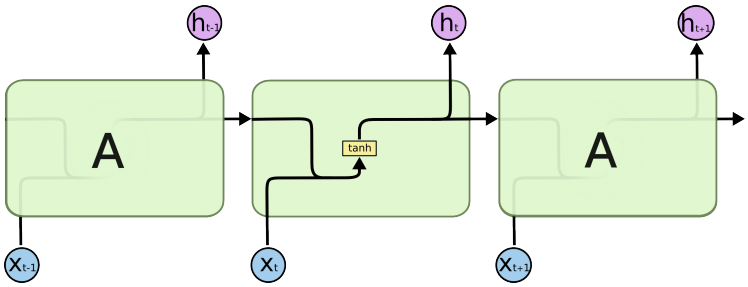


FIGURE 6.5: Unrolled recurrent neural network with a single tanh layer [38]

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 6.6.

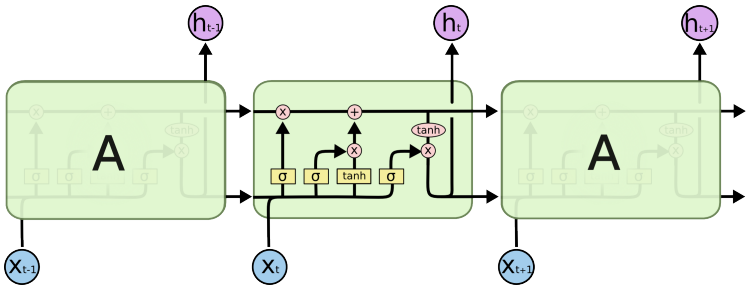


Figure 6.6: Unrolled LSTM network where each network has 4 layers [38]

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 6.7, 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.

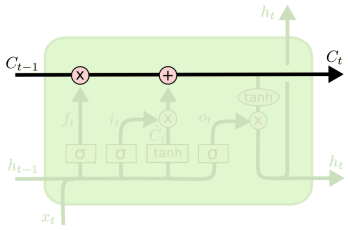
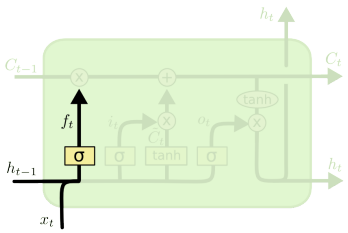


Figure 6.7: Representation of the cell state for a LSTM network [38]

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.

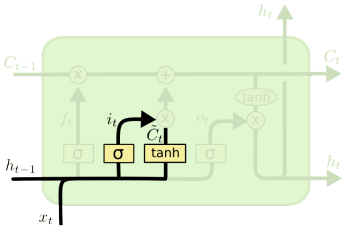
We start with the forget gate layer of a LSTM. Based on x_t and h_{t-1}, it outputs a number between 0 and 1 for each number in the cell state C_{t-1}. This is shown in figure 6.8.



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

Figure 6.8: Forget layer of a LSTM network [38]

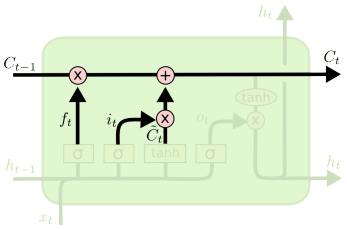
Next we look to the input gate layer. This gate decides which values will be updated in the cell state and outputs those in i_t. It is then combined with the vector $\widetilde{C}_t$, which contains the new candidate values based on the input x_t and h_{t-1}. This is shown in figure 6.9.



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

FIGURE 6.9: Input layer of a LSTM network [38]

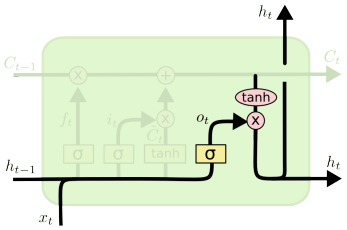
We can now combine the previous results and adjust the cell state. We multiply the old state with f_t so we forget the needed elements. Then we add $i_t * \widetilde{C}_t$ which are the new candidate values multiplied by the amount on how much we want to update each state value. See figure 6.10.



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

FIGURE 6.10: Update process of the cell state of a LSTM network [38]

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 6.11.



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

FIGURE 6.11: Decide the output of a LSTM network [38]

Variants Long Short Term Memory

In "LSTM: A Search Space Odyssey" [20], research is done between different variant of LSTM networks. It was concluded that the forget gate and the activation function

is the most important. Other variants do not have a large influence and mainly add a lot of extra complexity.

6.5 Conclusion

The final section of the chapter gives an overview of the important results of this chapter. This implies that the introductory chapter and the concluding chapter don't need a conclusion.

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] Using recurrent neural networks in dl4j - deeplearning4j: Open-source, distributed deep learning for the jvm. http://deeplearning4j.org/usingrnns. (Accessed on 05/21/2016).

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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).

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

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

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

[17] 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.

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

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

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

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

[22] 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.

[23] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

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

[25] 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.

[26] 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.

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

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

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

[30] 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].

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

[32] 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.

[33] 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.

[34] 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.

[35] A. Moores. Efficient memory-based learning for robot control. 1991.

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

[37] 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).

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

[39] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *CoRR*, abs/1211.5063, 2012.

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

[41] X. Rong. word2vec parameter learning explained. *CoRR*, abs/1411.2738, 2014.

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

[43] 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.

[44] 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].

[45] J. Simm. IMEC Technical talk.

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

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

[48] 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.

[49] D. D. Team. Deeplearning4j: Open-source distributed deep learning for the jvm.

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

[51] 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 Disease 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