CHALMERS
UNIVERSITY OF TECHNOLOGY

Using pre-learned semantic representations of biomedical concepts for analyzing electronic medical records

Master's thesis in Complex Adaptive Systems

TOBIAS SUNDELL

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

Using pre-learned semantic representations of biomedical concepts for analyzing electronic medical records

TOBIAS SUNDELL

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2020

Using pre-learned semantic representations of biomedical concepts
for analyzing electronic medical records
TOBIAS SUNDELL

Supervisors: Magnus Kjellberg, Sahlgrenska University Hospital
Yasaman Ettefagh, Department of Electrical Engineering
Examiner: Marija Furdek Prekratic, Department of Electrical Engineering

Master's Thesis 2020
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2020

Using pre-learned semantic representations of biomedical concepts
for analyzing electronic medical records
TOBIAS SUNDELL
Department of Electrical Engineering
Chalmers University of Technology

Abstract

With the increasing availability of electronic medical record data, machine learning methods have been developed to analyze this data. Such methods can for example be used to predict a patient's diagnoses or if the patient's state will deteriorate in the near future. This thesis investigates the benefit of using pre-learned semantic representations of biomedical concepts when making such predictions. The semantic representations are pre-learned from sources other than the analyzed medical records, such as scientific articles. Since concepts tend to retain their semantics across languages, this method has the advantage of being language-independent. Three different machine learning models are used to predict in-hospital mortality and early or unplanned hospital readmission. The models are logistic regression, random forest, and artificial neural networks. The predictions are made on patients from two data sets: an English data set with intensive care unit patients, and a Swedish data set with addiction care patients. Models using semantic representations are compared against baseline models that did not. The results show that the baseline models outperform the models using pre-learned semantic representations. Thus, unless a principally different method of using pre-learned semantic representations than the method employed in this thesis is used, there seems to be little promise in using semantic representations of biomedical concepts for analyzing electronic medical records.

Keywords: machine learning, transfer learning, nlp, emr

Acknowledgements

I first of all want to thank Magnus Kjellberg, my supervisor at Sahlgrenska, for many helpful and interesting discussions, as well as for giving me access to internal data of Sahlgrenska. I also want to thank Yasaman Ettefagh and Marija Furdek Prekratic for many helpful comments and advice during the writing of the report.

Tobias Sundell, Gothenburg, June 2020

Contents

1

Introduction

Health care systems are increasingly being digitized. Nowadays most health care providers store patients' medical records electronically, and efforts are being made to expand the use and communication of such records (e.g. in the USA [1], and the EU [2]). These records can contain both short-term information, such as vital sign measurements and lab test results, and long-term information, such as diagnoses and prescriptions. The adoption of electronic records has been deemed to improve decision-making, decrease the number of medical errors, and improve communication both with patients and other health care providers [3]. Apart from these benefits such records furthermore allow for uses that would be impossible with paper-based records, such as large-scale data mining, information extraction, and artificial intelligence decision-support systems.

One area of research exploiting this increase in available patient data is predicting unknown health-related information of patients using machine learning. These might be predictions regarding a patient's diagnosis, how long a newly admitted patient will need to stay in a hospital, or if a patient's state will rapidly deteriorate in the near future. Such predictions could support decision-making and avoid errors, and improve both patient care and the utilization of resources.

Machine learning methods are known to improve as more data becomes available. However, since electronic medical records contain patients' private information, gaining access to large amounts of such records is often an obstacle when producing machine learning models. The available data might be limited not only to a specific hospital but to single departments within the hospital. A way to improve predictions under these circumstances is to extract knowledge from sources *other* than confidential patient records, but which can still be useful when predicting patient outcomes. This is called *transfer learning*. For instance, electronic medical records contain biomedical concepts such as diagnoses, symptoms, and pharmacological drugs, and it seems likely that such concepts are very important for making predictions about what will happen to the patient. Instead of learning what these concepts mean only from the medical records, it is possible to learn this from other sources, such as scientific articles, and then transfer that knowledge to the problem of predicting patient outcome. Such "learned meanings" are called *semantic representations*. It would furthermore be advantageous if such transfer learning methods work *across languages*, since most of the available data are in English.

Previous work has been done on learning the meaning of biomedical concepts [4],

and some work has been done on utilizing this knowledge for predicting patient outcome [5]. This work showed that transferring knowledge of biomedical concepts to the problem of predicting patient outcome only had modest benefits. During the years since publication, however, progress has been made in extracting knowledge of biomedical concepts, which warrants testing transfer learning with this improved knowledge. Furthermore, no work has been done on transfer learning to other languages.

1.1 Purpose and Scope

The purpose of this thesis is to investigate if predictions on patient outcome using machine learning can be improved by using pre-learned semantic representations of biomedical concepts. In particular, the benefits of using semantic representations learned from sources in one language used on medical records in another language are investigated. A major contribution of the thesis is the development of methods for transfer learning across languages. Although the data sets used in the thesis are in two particular languages, the methods should be relatively simple to adapt to other languages. The thesis expands on previous work in the following ways: (1) it uses improved semantic representations of biomedical concepts, (2) it develops methods of transfer learning to other languages, (3) it uses partly different data sets, and (4) it employs partly different machine learning models.

The purpose is achieved by testing two different pre-learned sets of semantic representations of biomedical concepts on an English data set containing data about patients in intensive care, and on a Swedish data set containing data about patients in addiction care. Three kinds of machine learning models are used: (1) logistic regression, (2) random forests, and (3) simple artificial neural networks. The models are used to predict in-hospital mortality and unplanned or early readmission. In addition to the models using semantic representations, several similar baseline models will be used for comparison. In particular, a model that uses biomedical concept without the pre-learned semantic representations will be used. If the models using semantic representation perform better compared to the other models on the data sets, this would indicate that semantic representations do improve performance on similar predictive tasks on similar data sets.

The scope of the thesis is limited by the following factors:

- Relatively simple and fast models will be used. The goal is not to achieve state-of-the-art performance, but to compare how similar models perform given different input data.
- The study is limited to the two data sets (MIMIC-III [6] and a proprietary data set from Sahlgrenska University Hospital).
- The models will not necessarily be directly applicable to clinical settings. For example, discharge notes are used when predicting if a patient will be readmitted shortly after discharge. But if the patient has already been discharged, the usefulness of the information that the patient is likely to be readmitted might be diminished, since it might have been important when making the

decision whether the patient should be discharged or not.

1.2 Thesis Outline

The thesis is divided into the following parts:

- Chapter 2 introduces the theory the rest of the thesis is based on. It explains (1) electronic health and medical records, (2) machine learning and the types of models used in the thesis, (3) natural language processing, (4) transfer learning and the relevant techniques for transfer learning, and (5) common practices and considerations when training machine learning models. The chapter ends with a more in-depth treatment of the relevant previous work than was given in this chapter.
- Chapter 3 describes the methods used in the thesis. It consists of (1) how the biomedical concepts are extracted from free-text and how they are embedded using the pre-learned semantic representations, (2) a description of the data sets used, and (3) the models used to make predictions on the data sets.
- Chapter 4 shows the results of extracting concepts and making predictions on the data sets. It also includes a discussion of these results.
- Chapter 5 contains the most important conclusions that can be drawn from the thesis.

2

Theory

This chapter presents the theory required for the rest of the thesis. Section 2.1 describes Electronic Medical Records (EMRs) and Electronic Health Records (EHRs), what they are and how they are being used. Sections 2.2-2.4 describe relevant methods of machine learning, Natural Language Processing (NLP), and transfer learning, respectively, both in general and specifically in relation to EMRs. Section 2.5 discusses the relevant practices and techniques used to train machine learning models; and finally section 2.6 describes relevant previous work on analyzing EMRs using methods similar to those in this thesis.

The book [7] is the main source for the key machine learning concepts presented in this chapter.

2.1 Electronic Medical Records and Electronic Health Records

EMRs and EHRs are variously defined and sometimes used interchangeably. The definitions given below are the ones that will be used in this thesis.

An EMR stores information regarding a patient's medical history from a single health care provider (for example a hospital). It can contain many different kinds of information. Examples include the patient's diagnoses, drug prescriptions, results of lab tests, medical images, and hand-written clinical notes. The kinds of information produced can differ between departments. For example, an Intensive Care Unit (ICU) continually produces measurements of values signifying the patient's present state, while a psychiatric department does not. Different health care providers can have different EMR systems, collecting different kinds of information and using different formats. Hence the systems are not necessarily interoperable.

An EHR stores a subset of information from a patient's EMRs and is intended to contain more long-term health information and to be used across health care providers. For instance, short-term lab tests results should not be included, but diagnoses should. The EHR should contain all relevant health history of a patient.

EMRs and EHRs have been increasingly adopted by health care providers during the last decades. The simplest implementations of electronic records replace older paper-based records and facilitate storage and communication of such records. Electronic records have many potential advantages, such as the possibility of storing larger

amounts of information automatically and in real-time. However, the adoption of electronic records has encountered several obstacles, and while many hospitals have more or less advanced EMR systems, work is still being done to establish standards for EHRs [2].

Because of the huge and increasing amounts of EMR data for every patient, it is not feasible to manually inspect and analyze anything but small parts of it. Thus, automatic analysis of EMRs using machine learning has become a very active field of research.

2.2 Machine Learning

Machine learning has been defined in various ways. One illuminating definition, which will be adhered to in this thesis, is: "a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty" [7]. The data from which the machine learning method learns to detect patterns is called *training data*.

A simple example of a machine learning algorithm is *linear regression*. In this case the training data is N d-dimensional input vectors $\mathbf{x}_i \in \mathsf{R}^d$, each with a corresponding output $y_i \in \mathsf{R}$, where $1 \leq i \leq N$. Each dimension of the input data is called a *feature*. The objective is to fit a hyperplane $f(\mathbf{x}_i) = \hat{y}_i$ (i.e. find the hyperplane parameters) such that the total squared error $\sum_{i=1}^{N}(f(\mathbf{x}_i) - y_i)^2$ is minimized. In the special case $d = 1$, this is simply fitting a line to a set of 2-dimensional (x, y) points, and is illustrated in figure 2.1. Thus, using the training data $\mathbf{x}_i$ a mathematical model $f(\mathbf{x})$ is constructed (the hyperplane), which can then be used to make predictions $\hat{y} = f(\mathbf{x}')$ for previously unseen data $\mathbf{x}'$.

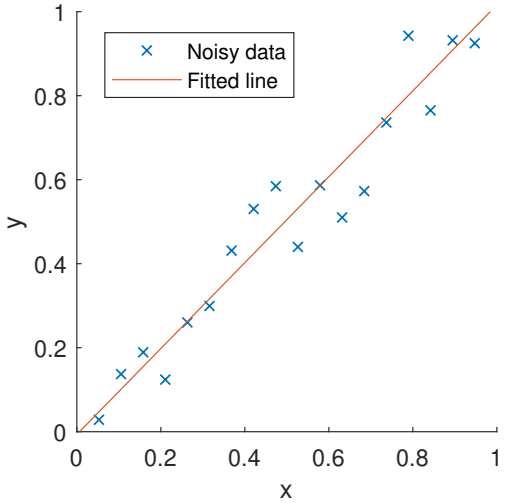


Figure 2.1: Example of a linear regression. The data are sampled from $y = x + \epsilon$, where ϵ is normally distributed noise with mean 0 and standard deviation 0.1.

Machine learning has traditionally been divided into the following approaches:

- **Supervised learning** learns a mapping $y = f(\mathbf{x})$, where each training input $\mathbf{x}_i$ has a known output y_i. Linear regression is an example of supervised learning.
- **Unsupervised learning** learns how to separate the input samples $\mathbf{x}_i$ into different groups based on their positions relative to each other in the d-dimensional space. In this approach, the input samples to do not have corresponding known outputs (labels) y_i.
- **Reinforcement learning** learns to adapt a model to a dynamic environment by getting positive and negative rewards from the environment as a response to the model behavior.

The only approach relevant to this thesis is supervised learning.

In supervised learning, the output y can either be continuous, i.e. $y \in \mathsf{R}$, or a discrete set of classes, i.e. $y \in \{1, 2, \ldots, C\}$. The former case is called a *regression* problem, while the latter is called a *classification* problem. In this thesis only classification problems will be discussed and, for simplicity, only *binary* classifications problems, i.e. $y \in \{0, 1\}$. The machine learning algorithms used, *logistic regression*, *Artificial Neural Networks (ANNs)*, and *random forests*, are described in the following sections in the case of binary classification problems.

2.2.1 Logistic Regression

Logistic regression is very similar to linear regression, except instead of predicting a real value $y \in \mathsf{R}$ from the input $\mathbf{x}$ it predicts the probability that $\mathbf{x}$ belongs to a certain class. In the binary classification case, the probability can easily be used to compute the estimated class $\hat{y} = \{0, 1\}$ by establishing a decision boundary using a threshold θ:

$$\hat{y} = \begin{cases} 0 & \text{if } p(y = 1) \leq \theta \\ 1 & \text{if } p(y = 1) > \theta \end{cases} . \tag{2.1}$$

A common threshold is $\theta = 0.5$.

The model used to predict $p(y = 1)$ is based on a linear combination of $\mathbf{x}$: $w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$, where w_i are model parameters to be learned. If an element $x_0 = 1$ is added to $\mathbf{x}$ it can be written more succinctly as $\mathbf{w}^T\mathbf{x}$. To ensure that this linear combination produces a probability $0 \leq p(y = 1) \leq 1$, it is limited to the interval $[0, 1]$ using a sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$, so that

$$p(y = 1) = \sigma(\mathbf{w}^T\mathbf{x}) . \tag{2.2}$$

An illustration of a 1-dimensional fitted logistic regression model with decision boundary at $\theta = 0.5$ is shown in figure 2.2. All the samples to the left of the decision boundary are predicted to belong to class 1, and everything to the right to be class 0.

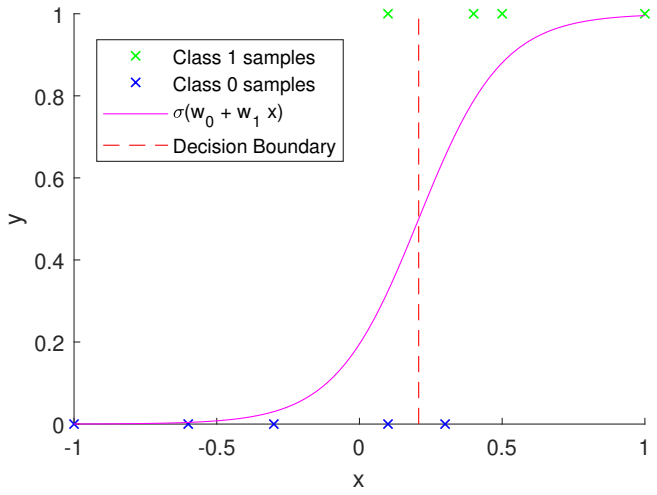


Figure 2.2: Example of a logistic regression.

The model is fitted to the training data by minimizing an error or *loss* function. Logistic regression uses the *cross entropy* loss function. Its derivation and motivation is beyond the scope of this thesis (for details see e.g. [7]). In the binary case, letting $p_i = p(y_i = 1) = \sigma(\mathbf{w}^T \mathbf{x}_i)$, it has the equation

$$L(\mathbf{w}) = \sum_{i=1}^{N} \left(y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right) \ . \tag{2.3}$$

An optimization method is then used to find the minimum of this loss function, i.e. $\min_{\mathbf{w}} L(\mathbf{w})$. The most commonly used optimization methods for logistic regression are so-called *quasi-Newton methods*, which are all based on *Newton's method*.

Newton's method iteratively finds an optimum of a twice-differentiable function. At an iteration k, an optimal parameter estimate $\mathbf{w}_k$ is used to produce an improved estimate $\mathbf{w}_{k+1}$ using the equation

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \frac{\nabla L(\mathbf{w}_k)}{\nabla^2 L(\mathbf{w}_k)} \tag{2.4}$$

where ∇ denotes the gradient of a function, and ∇^2 denotes the Hessian. Usually the method is stopped either after a maximum number of iterations k_{max} or when the difference between iterations falls below some selected tolerance ϵ, i.e. when $|L(\mathbf{w}_{k+1}) - L(\mathbf{w}_k)| < \epsilon$. The quasi-Newton methods differ from Newton's method in that they approximate the Hessian $\nabla^2 L(\mathbf{w}_k)$ instead of calculating it directly since this is a computationally intensive operation for large d.

2.2.2 Artificial Neural Networks

There are many different kinds of ANNs. The only kind that will be considered here is *feed-forward*, *fully-connected* networks.

An ANN consists of artificial neurons. An artificial neuron is a simple mathematical function loosely modeled on biological neurons. It takes a d-dimensional vector $\mathbf{x} \in \mathsf{R}^d$ as input, and outputs a single number $y \in \mathsf{R}$. Internally it performs a linear combination of the input $\mathbf{x}$, adds a bias term, β, and applies a so-called *activation function* f to the result. Expressed mathematically,

$$y = f(\mathbf{w}^T\mathbf{x} + \beta) \ . \tag{2.5}$$

The activation function could be anything, but common ones are Rectified Linear Unit (ReLU), tanh and sigmoid functions. Clearly, if the activation function is a sigmoid, the artificial neuron is equivalent to logistic regression.

ANNs consists of many neurons. In feed-forward networks, the neurons are organized in sequential layers, and a neuron in one layer only has connections to neurons in the next layer; no connections between neurons in the same layer or between non-neighboring layers are allowed. An illustration is shown in figure 2.3. In a fully-connected network, each neuron in a layer is connected to every neuron in the next layer. The illustrated network is fully-connected. The first layer is called the *input layer*. It does not consist of neurons but of the current input vector $\mathbf{x}$. The last layer is the *output layer*. Any layers in between the input and output layers are called *hidden layers*. There can be an arbitrary amount of hidden layers as well as an arbitrary number of neurons in any layer.

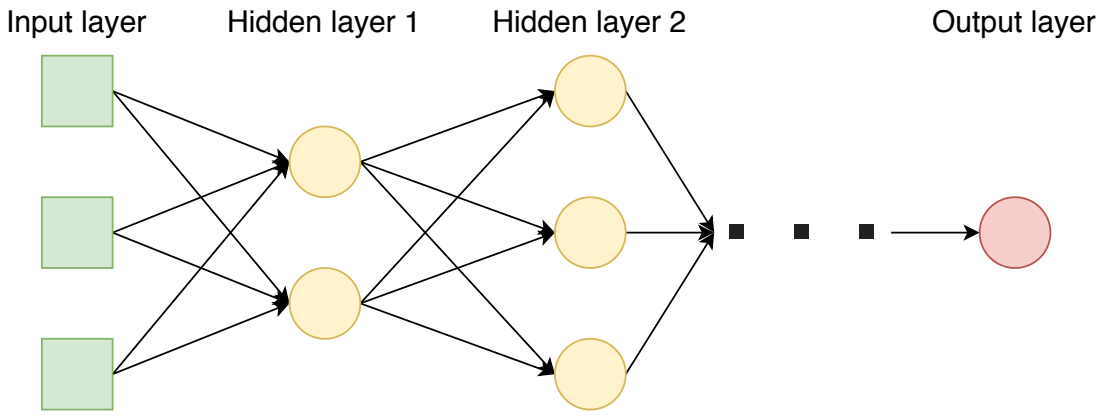


Figure 2.3: Example of an ANN. The squares represent input features, circles represent neurons, and the arrows represent connections between them. There can be any number of hidden layers.

To produce an output from a given input, the values of subsequent layers are fed forward to the next layer consecutively, and the value of the output layer is taken as the output of the whole network. If a network is used for binary classification, the output layer generally consists of a single neuron with a sigmoid activation function (thus essentially performing a logistic regression on the output from the previous layers). This ensures that the output $p \in [0, 1]$ can be interpreted as the probability that the input $\mathbf{x}$ belongs to class 1.

To train an ANNs a loss function to be minimized has to be defined. For binary

classification binary cross entropy (the same as for logistic regression) is commonly used. If there are N vectors of input $\mathbf{x}_i$ with corresponding known classes y_i and the ANN outputs predictions p_i, the loss function is

$$L(\mathbf{W}) = \sum_{i=1}^{N} (y_i \log(p_i) + (1 - y_i) \log(1 - p_i)) \tag{2.6}$$

where $\mathbf{W}$ denotes all model parameters (i.e. the neurons' weights and biases). The loss function is usually optimized using gradient descent methods (such as Adam or RMSprop), a method very similar to the Newton's method described above. Gradient descent takes steps in the descent direction, using the equation

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \gamma \nabla L(\mathbf{W}_k) \tag{2.7}$$

where γ is the learning rate. It is often selected experimentally. A too large learning rate might hinder convergence, while convergence might be too slow with a too small learning rate. Many extensions to the simple gradient descent method have been developed, usually to facilitate convergence, but will not be discussed here.

Since calculating the gradient for each parameter in $\mathbf{W}$ separately is very computationally intensive, the *backpropagation* algorithm is used instead. Backpropagation takes advantage of the feed-forward structure of the network, since in that case the gradient for a neuron in layer l depends only on the gradient of the neurons in layer $l + 1$. Thus, in backpropagation, the gradients are calculated sequentially from the output layer to the input layer, instead of calculating the gradient for a specific variable directly from the loss function.

A feed-forward fully-connected ANN is a universal function approximator: an arbitrarily complex ANN can approximate any continuous function with arbitrarily small error, where the error can be decreased by increasing the network complexity [8]. Of course, available computational resources and training data limits the possible network complexity in practice. A suitable network architecture that produces accurate predictions has to be found experimentally.

2.2.3 Random Forest

Random forest is a machine learning method based on decision trees. As the name suggests, a random forest is a multitude of different decision trees. In the binary classification case, each decision tree produces a probability that an input vector $\mathbf{x}$ belongs to class 1. The forest then produces the final probability estimate as the average of the trees' estimates. How decision trees work and how a forest is grown using the training data will be discussed in the following sections.

2.2.3.1 Decision Trees

A decision tree is most easily described using a figure. An example is shown in figure 2.4. To classify the input vector $\mathbf{x}$ as either class 0 or class 1, the tree is followed

from the root node to a leaf node according to the condition at each non-leaf node. Each leaf node has an estimated probability that the input belongs to class 1. When such a node is reached this probability is the output of the decision tree. A decision tree is not required to use all input features $x_1, x_2, ..., x_d$.

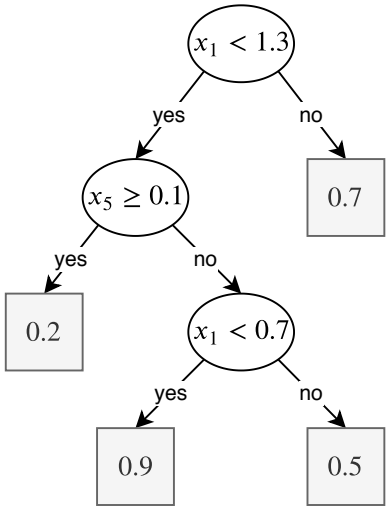


Figure 2.4: Example of a decision tree. Ellipses represent non-leaf nodes, squares represent leaf nodes. The condition shown in each non-leaf node determines which path is followed. The probability shown in each leaf node is the output of the decision tree if that node is reached.

2.2.3.2 Growing a Forest of Decision Trees

Growing an optimal decision tree is an NP-complete problem. Hence, the algorithms used to grow decision trees are approximations. It is common to use a greedy algorithm for this. The algorithm builds the tree recursively from the root, and at each node selects the condition that is currently optimal (although not necessarily optimal for the whole tree). Optimality is measured either as maximum information gain or minimal expected error rate (Gini impurity). At each node, the training data are split according to the node condition so that the children are created using only the split data. Decision trees grown this way however tend to have less accuracy than other models and to be unstable to small changes in the training data. Growing many decision trees as part of a forest is a way to solve both problems.

In a Random Forest, many trees are grown and their predictions are averaged. Randomness is introduced to ensure that the trees are different from each other, since given the same data the greedy algorithm would produce identical trees. Random Forests grow individual trees based on random subsets of both the training data and the training data features. Since the trees get to see different parts of the training data, they will learn to identify different patterns. While each pattern might not be as strong a predictor as the strongest patterns in the training data, taken together they often constitute an accurate and robust classifier.

All the discussed machine learning methods take d-dimensional vectors of numbers as input. Text, composed of words and punctuation, cannot be readily represented

as such vectors. The field of NLP has developed methods of representing text as vectors.

2.3 Natural Language Processing

Many techniques and methods for NLP have been developed. Two of the simplest ones are *one-hot word vectors* and *bag of words*. Another method is *word embeddings*.

2.3.1 One-Hot Word Vectors

A one-hot vector is a vector of n binary numbers $\mathbf{x} \in \{0,1\}^n$ where exactly one element is 1 and the others are 0. Such vectors can be used to represent individual words. If n is the number of unique words in a corpus of texts T and each word has a unique index $1 \leq i \leq n$ (e.g. "heart" might correspond to $i = 5$), the word with index i can be represented by the vector with $x_i = 1$ and $x_{j \neq i} = 0$. The vector $\mathbf{x}$ can readily be used as input to the machine learning methods described above. Several one-hot vectors can be combined to produce a bag of words.

2.3.2 Bag of Words

Bag of words represents a specific text t in a set of corpus T by a vector $\mathbf{x} \in \{0,1\}^n$. In contrast to one-hot vectors, many elements can have the value 1. The elements in $\mathbf{x}$ are determined as

$$x_i = \begin{cases} 1 & \text{if word } i \text{ is in } t \\ 0 & \text{otherwise} \end{cases} . \tag{2.8}$$

Clearly, however, the number of instances of a word and word order are not represented by bag of words. Also, n might become very large and some words might be present in all texts of T, so several methods have been developed to deal with these problems. Generally, stop words such as "a", "it", "the", "from", etc. are removed from the texts. Words can be lemmatized, i.e. replaced with their canonical forms, so that "sleeping" becomes "sleep", "cars" become "car", etc.

Of course, the same principle used in bag of words can be used for entities other than words. For example, if a set of texts contain biomedical concepts, the concepts can be extracted to create bags of concepts in the same fashion.

2.3.3 Word Embeddings

Another way is to represent the words using so-called embeddings, with vectors $\mathbf{x} \in \mathsf{R}^d$. Embeddings can be created in different ways. The simplest way is using random embeddings, where each word is assigned a random point in R^d-space, and d is arbitrary. The primary benefit of using randomized word embeddings over one-hot word vectors is a possible reduction of dimensionality if $d < n$. Randomized word embeddings, however, are not used to represent whole texts.

It is also possible to compute word embeddings using a data set. This is a machine learning problem in itself. The data set from which the embeddings are computed need not be the data set containing the words to be embedded, making word embeddings widely used in NLP *transfer learning* methods.

2.4 Transfer Learning

Transfer learning refers to the practice of transferring knowledge learned by solving a task (the *source task*) on a data set (the *source data set*) to solving another task (the *target task*) on another data set (the *target data set*). For example, a model trained to recognize cats in images will probably have learned patterns useful for recognizing dogs. Transfer learning has garnered lots of interest and been used in many different settings the past decades [9]. Two methods of transfer learning for NLP will be discussed in this section: word2vec and the newer Bidirectional Encoder Representations from Transformers (BERT).

2.4.1 word2vec

word2vec is a technique to construct vector representations (embeddings) of words [10]. It uses either of two source tasks: Continuous Bag Of Words (CBOW) or Continuous Skip-Gram, on a source data set consisting of texts. CBOW is the task of predicting a word given a bag of words containing its surrounding words. For example, given the sentence "the cat is on the mat", the task could be to predict the word "cat" given a bag of the words "the", "is", "on", and "mat" (assuming that stop words have not been removed). Skip-gram, conversely, predicts the surrounding bag of words given a single word. For example, the task could be to predict the bag of words "the", "is", "on", and "mat", from the word "cat". Thus, words which occur in the same contexts will get similar embeddings. These tasks are achieved using a feed-forward fully-connected ANN with one hidden layer. Which of the two tasks achieve the best performance has to be found experimentally. After the network has been trained, the embeddings are extracted from the parameters of the network.

It was found that word2vec was able to learn both syntactic and semantic relationships between words. When trained on large amounts of general text, it was for instance able to learn that

$$vector(\text{"}biggest\text{"}) - vector(\text{"}big\text{"}) + vector(\text{"}small\text{"}) \approx vector(\text{"}smallest\text{"})$$

and

$$vector(\text{"}king\text{"}) - vector(\text{"}man\text{"}) + vector(\text{"}woman\text{"}) \approx vector(\text{"}queen\text{"}) \ .$$

Of course, the techniques used by word2vec are applicable to other kinds of entities than just words. As long as an entity occurs in a context, the word2vec approach is applicable. In this thesis, word2vec embeddings of *biomedical concepts* will be used, which have learned biomedical relationships.

2.4.2 Bidirectional Encoder Representations from Transformers

BERT is another technique that can be used for transfer learning in NLP [11]. In contrast to word2vec, BERT is not primarily designed to produce word vector representations, although such word vectors can be extracted from a trained BERT model. Rather, an ANN is trained on the source task on the source data set, and then the whole ANN is transferred to the target task on the target data set. Only the output layer is replaced by a task-specific output layer. The knowledge learned from the source data is encompassed by the ANN parameters.

BERT is trained on two source tasks. In the first a fraction of the words in a sentence are masked and the task is to predict which those words are. The second task is to predict the next sentence after a given sentence. Thus, BERT is designed to be trained on general text, and is not suitable to train using only concepts.

The BERT ANN without an output layer is shown in figure 2.5. It consists of an embedding layer (which produces embeddings suitable for the rest of the ANN), and a number of transformer encoder layers. The transformer encoder layer learns to encode the inputs. An output layer that makes use of the output of the transformer encoder layers has to be added for each specific task. BERT does not consist of only fully-connected layers but is based on transformers [12]. It is is a much more complex model than the relatively simple word2vec, and the details will be omitted here.

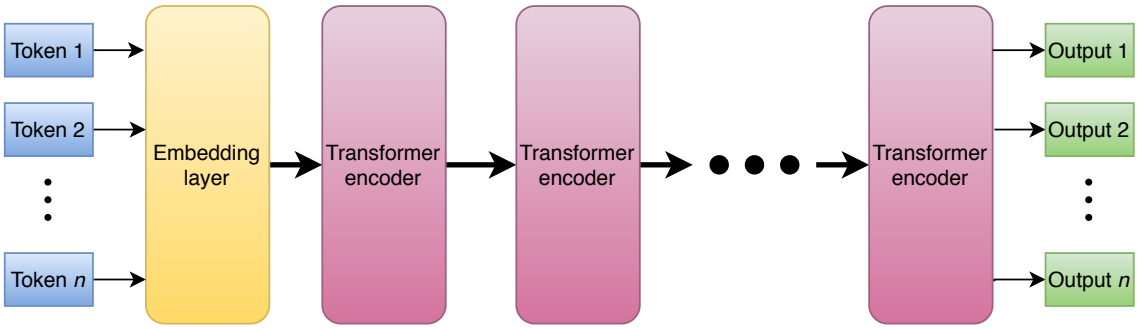


Figure 2.5: The BERT ANN architecture. There are a variable number of transformer encoder layers. Output from the transformer encoders should be fed into a task-specific output layer.

BERT takes *tokens* as input, where tokens are either words or sub-word sequences (for example "playing" could be represented by the tokens "play" and "##ing"). All tokens in the sentence are used when encoding a single token, resulting in *contextualized* encodings, i.e. a token will have different encodings when occuring in different contexts. BERT does not have the same limitations as word2vec does: it takes both a larger context and word sequence into account.

Although not designed to produce fixed word vector embeddings, the authors note that it is possible to extract token embeddings from the output of the transformer-encoder layers [11]. This is done by inputting only the tokens for the target word to the ANN and extracting the output of one or several encoder layers. Through

experimentation it was found that in a network with 12 encoder layers the concatenated output of the last four layers yields the best embeddings, and the sum of the last four layers perform almost as good while benefiting from a large reduction in dimensionality. Embeddings of sequences of tokens can be procured by taking the mean value of each feature of the token embeddings.

Some of BERT's power is lost when a fixed sequence embedding is extracted since the context is omitted. However, because of the different source tasks, model complexity, and the number of surrounding words used when training the encoders, such extracted embeddings can be expected to perform better than similar word2vec embeddings [11].

2.5 Machine Learning Training Practices

In this section some common practices used to solve challenges posed when training machine learning models are discussed.

2.5.1 Data Set Selection

One of the most important aspects determining the performance of a machine learning method is the amount and quality of available data. A machine learning model learns to make accurate predictions by detecting patterns in the data. Obviously those patterns must be present in the training data and be distinguishable from noise. The performance of machine learning models are often primarily limited by the available data.

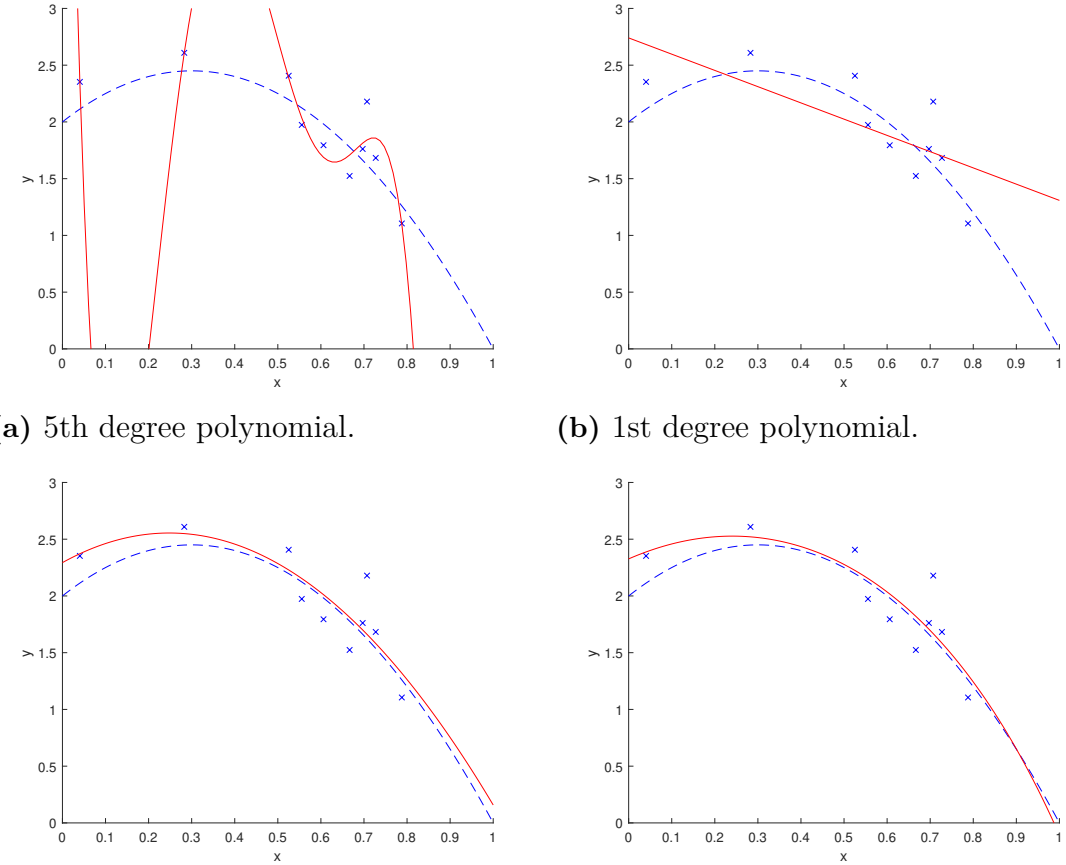
Training data should be collected uniformly from the correct distribution. For example, if the problem is to recognize photographs of cats, photographs of all kinds of cats, taken from all kinds of different distances, under different lighting conditions and camera rotations, etc. would be the ideal, although probably impossible in practice. A sample of common kinds of cat photographs would have to suffice, which means that the model likely would have worse performance on uncommon kinds of cat photographs.

In the health care case, patient data are often only available from a single or a few hospitals and departments. Data from different hospitals and departments will have more or less different patterns, and models trained on data from one source will not necessarily be generalizeable to data from another source. However, so far research has mostly focused on developing models for data from only one or a few sources [13].

2.5.2 Overfitting and Underfitting

Overfitting occurs when an unsuitably complex model is fitted to noisy data such that the model learns the noise and not the patterns in the process that generated the data. An example of this is shown in figure 2.6a, where a 5th degree polynomial is fitted to data sampled from a 2nd degree polynomial with added noise. While

the error for the data used to train the model is low, the error for new samples is expected to be large.



(a) 5th degree polynomial.

(b) 1st degree polynomial.

(c) 2nd degree polynomial.

(d) 5th degree polynomial with regularization.

Figure 2.6: Example of overfitting, underfitting, and regularization. The blue dashed curve is the original 2nd degree polynomial function, the blue crosses are samples from the original function with added noise, the red lines are different fitted polynomial models. **(a)** is an example of overfitting. **(b)** is an example of underfitting. **(c)** and **(d)** are examples of good fits, considering the few samples.

Underfitting, conversely, occurs when an unsuitably simple model is fitted to data generated by a process that is too complex for the model to fit. An example is shown in figure 2.6b, where a 1st degree polynomial is fitted to data generated from the 2nd degree polynomial with added noise.

Underfitting is mitigated by using a more complex model that can fit the data better. Overfitting is mitigated by using more training data (which is more difficult to fit to), simpler models, and/or regularization techniques. The latter two are illustrated in figures 2.6c and 2.6d, respectively.

2.5.3 Regularization

Regularization techniques attempt to limit the complexity of a model during training. A common technique is to add a regularization term to the loss functions, imposing a penalty on the model complexity. For example, using the logistic regression loss function $L(\mathbf{w})$ from section 2.2.1,

$$L'(\mathbf{w}) = L(\mathbf{w}) + \lambda C(\mathbf{w}) \tag{2.9}$$

where λ is a regularization penalty parameter, and $C(\mathbf{w})$ is some measure of complexity. Two common measures of complexity are the ℓ^1 norm $C(\mathbf{w}) = \sum_i |w_i|$, and the ℓ^2 norm $C(\mathbf{w}) = \sqrt{\sum_i w_i^2}$, both of which treat parameter magnitude as a measure of complexity. These regularization techniques can also be used for ANNs, and any other model that optimizes a loss function. Figure 2.6d illustrates fitting a 5th degree polynomial to data sampled from a 2nd degree polynomial with added noise using ℓ^2 regularization.

Random forests can be regularized by limiting individual tree complexity. Since the trees can grow very complex, leading to overfitting, the growth of the trees is commonly limited by one or more parameters.

- **Maximum depth** defines the maximum depth an individual tree is allowed to have.
- **Maximum leaf nodes** defines the maximum number of leaf nodes an individual tree is allowed to have.
- **Minimum split size** defines the minimum train data size that must remain after a split. The idea is that the tree should not create rules specific to only very small amounts of training data.

The number of trees in the forest is also a parameter. In most cases, the benefit of more trees plateaus after a certain number and does not lead to overfitting. Thus, this parameter only has to be set "large enough".

Another regularization technique is *early stopping*. This technique is applicable to iterative training algorithms. Model complexity usually increases during the training process, which thus goes from underfitting to overfitting. Early stopping stops the training process at the iteration of peak performance.

2.5.4 Holdout Validation and Cross Validation

Because of the possibility of overfitting, if all available data are used to train the algorithm, there would be no way to gauge model performance on new data, which is generally much more important than performance on training data. Therefore, parts of the available data are used for *validation* rather than training. Validation is simply measuring the performance on this validation data to get an indication of how well the model performs on data it has not been trained on.

The simplest method of getting validation data is using a fraction of the available data, for example 0.1 or 0.15. This is called *holdout validation*. Usually validation

data are sampled randomly from the available data. The data set should be large enough to give statistically significant results.

Another method of validating a model is *cross validation*. In this method the available data are split into k parts (or folds) for k-fold cross validation. The model is then trained and validated k times, each time using a different part for validating and the rest for training. The final validation result is then taken as the mean of the k different validations. Cross validation gives a better gauge of the model performance on new data but requires more computation compared to holdout validation.

The validation data are used for model and hyperparameter (e.g. the λ regularization parameter) selection. Because of this there is a risk that the final model is also overfitted to the validation data. It is common practice, therefore, to also reserve a fraction of the available data, called the *test data set*, for final testing of the model. The test data set should never be used before the final testing in order to avoid influence on model and hyperparameter selection.

Validation and testing of a model however requires a suitable metric, which in many cases is not obvious.

2.5.5 Evaluation Metrics

The most suitable metric used to evaluate a model depends on what the model will be used for. An obvious possible metric for training algorithms with a loss function is the loss. However, this metric is often not very useful in practice. For classification problems the accuracy is a more intuitive metric. It is defined as

$$Acc = \frac{N_{correct}}{N} \tag{2.10}$$

where $N_{correct}$ is the number of correctly classified samples and N is the total number of samples. Although simple and intuitive, accuracy is not always the most useful metric. False negatives might be more important than false positives, or vice versa, for instance.

Actual / **Predicted**	**Positive**	**Negative**
Positive	Number of True Positives (TP)	Number of False Positives (FP)
Negative	Number of False Negatives (FN)	Number of True Negatives (TN)

Table 2.1: Confusion Matrix.

The number of true positives, false positives, false negatives, and true negatives can be visualized using a *confusion matrix*. It is illustrated in table 2.1. Interesting metrics that can be derived from these numbers are *positive predictive value* or *precision*

$$PPV = \frac{TP}{TP + FP} \ , \tag{2.11}$$

true positive rate or *recall*

$$TPR = \frac{TP}{TP + FN} \ , \tag{2.12}$$

and *false positive rate*

$$FPR = \frac{FP}{FP + TN} \ . \tag{2.13}$$

Another commonly used metric is F_1 score, which combines precision and recall:

$$F_1 = 2 \times \frac{PPV \times TPR}{PPV + TPR} \tag{2.14}$$

Since the models discussed outputs probabilities and not predictions directly, the threshold θ (see section 2.2.1) used to determine predictions from the probabilities can be varied. The Receiver Operating Characteristic (ROC) curve plots TPR versus FPR as θ is varied. An example is shown in figure 2.7. A model guessing at random corresponds to the straight line $TPR = FPR$. As a model improves over random guessing, the curve gets stretched towards the top-left corner. Selecting a specific threshold θ corresponds to picking a point on the curve, yielding a specific trade-off between true positive and false positive rates. A perfect model reaches the point at the top left, i.e. $TPR = 1$ and $FPR = 0$.

The ROC curve can be reduced to a single number by taking the Area Under Curve (AUC) of the ROC. The random guess model has 0.5 ROC AUC. The higher the number the better the model.

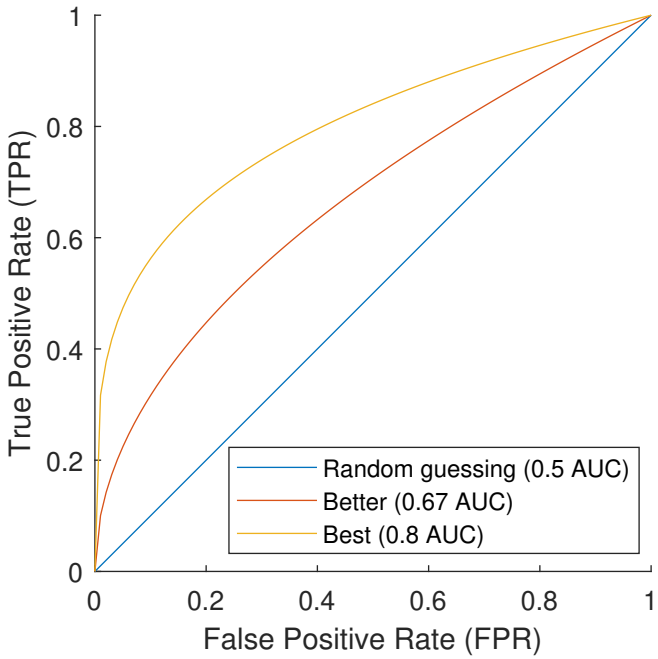


Figure 2.7: Examples of ROC curves. The more the curve bends towards the top-left corner, the better the ROC.

2.5.6 Data Standardization

Data standardization is translating and scaling of a data set so that it has mean 0 and variance 1. That is, if x_{ij} is feature j of sample i, $\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij}$ is the mean value of feature j of all N samples, and $\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{ij} - \mu_k)^2$ is the variance of feature j of all N samples, then each sample is transformed by

$$x'_{ij} = \frac{x_{ij} - \mu_j}{\sqrt{\sigma_j^2}} \,. \tag{2.15}$$

Data standardization tend to facilitate convergence, and some machine learning models assume that the data are standardized. Take for instance ℓ^1 regularization for logistic regression, which penalizes the magnitude of the parameters ($C(\mathbf{w}) = \sum_i |w_i|$). For simplicity, assume there is only one feature x_{i1}, and hence one regression coefficient w_1. The output of the model is then $p(y_i = 1) = \sigma(w_0 + w_1 x_{i1})$. If feature 1 is scaled by s, w_1 has to be scaled by $1/s$ in order for the model to make the same prediction (since $(w_1/s)x_{i1}s = w_1 x_{i1}$). But this changes the magnitude of w_1 and thus of the penalty incurred by that feature by the regularization. Clearly, a lower variance implies a higher regularization penalty for a feature. Thus, if ℓ^1 (or ℓ^2) is used, the data should be standardized so that all features have similar regularization penalties.

The mean μ_j and variance σ_j^2 are calculated only on the training data set. This is because the validation set (and test set) are supposed to represent unknown data of which the mean or variance are not known. When validating, however, the validation

set (and test set) are standardized using the mean and variance from the training set.

2.5.7 Class Imbalance

It is not seldom the case that the classes in a classification problems are *imbalanced*, that is, there is not a roughly equal number of samples for each class in the data. In the following, assume there are two classes, and that class 0 have many more samples than class 1. If the loss function is written as the sum of the loss of the samples

$$L = \sum_{i=1}^{N} l_i \tag{2.16}$$

where l_i is the loss of sample i, the total loss will clearly be lower if the model prioritizes achieving lower mean loss for class 0 than for class 1. In extreme cases, for example when 0.99 of the data are class 0 and 0.01 are class 1, the model might simply learn to classify all samples as class 0. But in such cases it is probably of more practical importance to find potential class 1 samples than achieving a total high accuracy.

A simple way of alleviating class imbalance is adding a weight to each sample loss l_i according to its class.

$$w_0 = \frac{N}{N_0} \qquad\qquad w_1 = \frac{N}{N_1}$$

where N_0 is the number of class 0 samples and N_1 is the number of class 1 samples. Then, if the losses are split into l_{0i} for class 0 losses and l_{1j} for class 1 losses,

$$L' = w_0 \sum_{i=1}^{N_0} l_{0i} + w_1 \sum_{j=1}^{N_1} l_{1j} \ . \tag{2.17}$$

Clearly, if both classes have the same mean loss m over all their samples, the contribution to the total loss of each class is the same:

$$L' = N\frac{1}{N_0}\sum_{i=1}^{N_0} l_{01} + N\frac{1}{N_1}\sum_{j=1}^{N_1} l_{1j} = Nm + Nm \tag{2.18}$$

In this sense the classes are balanced in the loss function, although not in the number of samples.

2.6 Related Work

This section presents previous work relevant to the thesis subject. A lot of work has been done in these areas so an exhaustive treatment is beyond the scope of the

thesis. The focus will be on the most relevant previous work, especially that whose results the thesis makes use of.

2.6.1 Biomedical Concept Embeddings

Many different biomedical concept embeddings have been produced, in various ways and from various sources. A survey of word embeddings, including concept-only embeddings, can be found in [4].

In the thesis two relatively recent ones will be used, both of which have achieved good results on concept similarity evaluations. cui2vec [14] uses an approach similar to word2vec to create concept embeddings from medical journals (PubMed), health insurance claims, and clinical notes, which is a larger source data set compared with other embeddings. ClinicalBERT [15] is a BERT model trained on general text (Wikipedia and BookCorpus) and then on clinical notes (MIMIC-III). Although the BERT model does not directly produce embeddings, embeddings from ClinicalBERT can be extracted as described in section 2.4.2.

2.6.2 EMR Analysis

In recent years deep learning methods (commonly defined as neural networks with many hidden layers) have been increasingly used to analyze EMRs, see [13] for an overview. Compared to traditional machine learning methods, such as logistic regression and random forests, deep learning generally achieves better results and requires less feature engineering (i.e. manually determining which data features to use as input to the models). Much of the work has been focused on how to combine the different kinds of EHR data, such as measurements and clinical notes, in a single model.

The previous work most similar to this thesis is [5], where first published word2vec concepts embeddings from journal abstracts [16] were tested against concept embeddings calculated from the target data set, as well as against a bag of words baseline and recurrent neural networks for predicting mortality, inpatient admission, and future emergency room visits. They found that both kinds of concept embeddings had similar performance and outperformed bag of words on small data sets (125, 250 and 500 patients) but that bag of words caught up and achieved similar or even better performance on larger data sets (1000 or more patients). The pre-trained concept embeddings used by this study however used a much smaller data set than cui2vec, and did not use BERT, justifying this thesis' purpose of testing more recent concept embeddings for similar tasks and developing methods for applying them on Swedish EMRs.

2.6.3 Named Entity Recognition

Named Entity Recognition (NER) is the field of recognizing *named entities* in unstructured text and categorising them according to some set pre-defined categories. Such categories can be geographical places, given names, physical objects, etc. In a biomedical context they may be disease or disorder, symptoms, body parts, etc.

Although NER is not directly relevant to the thesis as such, some NER systems are based on identifying pre-defined biomedical concepts in unstructured text for which the categories are known. Being able to identify biomedical concepts in text is a pre-requisite for embedding the concepts and using them in machine learning models.

Several medical NER systems are based on pre-defined lists of biomedical concepts. Most such systems use lexical analysis, such as MetaMap [17] and its less accurate but faster variant MetaMap Lite [18], and cTAKES [19]. While this approach often yields relatively accurate results, it is relatively slow and the lexical analysis has to be modified for every language. Another approach, employed by Narrative Information Linear Extraction (NILE) [20], is to shallowly match concept phrases word-by-word. The authors found that this approach can be orders of magnitude faster than the lexical analysis approach while still yielding comparable results. It is also language agnostic, requiring only language-specific dictionaries (although NILE additionally incorporates semantic analysis which is language-specific).

3

Methods

This chapter describes the methods used to evaluate the benefit of using concept embeddings when analyzing EMRs. Although many parts of an EMR contain concepts, such as lists of the patient's diagnoses and prescriptions, the majority and widest variety of concepts can be found in the clinical notes. Section 3.1 describes how concepts were extracted from clinical notes and embedded, and how the embeddings were aggregated into a single fixed-length feature vector. Section 3.2 describes the two data sets the analysis tasks were performed on. Finally, section 3.3 describes the models used and how they were trained. The code used for MIMIC-III is publicly available[1].

3.1 Concept Extraction and Embedding

This section describes the methods used to find and extract biomedical concepts in Swedish and English text, and embed these concepts using pre-trained embedding vectors. All text used was pre-processed such that punctuation was removed and all characters transformed to lower case. First the Unified Medical Language System (UMLS) is introduced. UMLS has term-concept mappings and is used by the subsequent methods.

3.1.1 UMLS

There exist many different medical coding standards and terminologies. Some are specific to certain areas, such as diagnoses or drugs, others are more general. UMLS [21] combines over 200 terminologies and provides tools for accessing them. The terminologies are mainly in English but also in some other languages, including Swedish. Different terminologies contain different terms and IDs for the same concepts. UMLS provides a global Concept Unique Identifier (CUI) (of the form "C1234567") for each concept, and contain mappings from terminology-specific IDs to CUIs. Thus, the CUIs of the concepts in a clinical note can be found by matching UMLS concept terms with the text of the clinical notes.

For English text, the concept terms from Systematized Nomenclature of Medicine Clinical Terms (SNOMED CT) were used. It is "the most comprehensive multilingual clinical healthcare terminology available" and designed to be used for EMR

[1] `https://github.com/jrrr/emr-semantic-representations`

systems [22]. Both SNOMED CT's Preferred Terms (PT) and Synonyms (SY) were used.

For Swedish text, in addition to the Swedish translation of SNOMED CT the translations of Medical Subject Headings (MeSH) and International Classification of Primary Care (ICPC) were used, since the Swedish version of SNOMED CT is less comprehensive than the English one. MeSH is primarily used for indexing and searching medical documents. ICPC is primarily used for classifying data from health care encounters. All available terms from these sources were used.

3.1.2 Finding Concepts in Text

The method of finding and extracting concepts from text must be fast enough to handle gigabytes of clinical notes in a reasonable time. MetaMap and MetaMap Lite (see section 2.6) were found to be much too slow. NILE's string matching approach would be significantly faster. However, since NILE also uses English-specific semantic analysis to determine whether the concepts were e.g. negated in the text or not, modifications would have to be made to adapt it to Swedish text. It was furthermore hypothesized that, since it is not uncommon for clinical notes to contain misspellings, accuracy could be improved by allowing slight misspellings, which NILE does not support. Therefore a new, language agnostic concept finder program was developed. It uses SymSpell[2] to first attempt to correct misspelled terms, followed by string matching using hash tables similar to NILE's approach. An overview of the program is shown in figure 3.1. The code is publicly available[3].

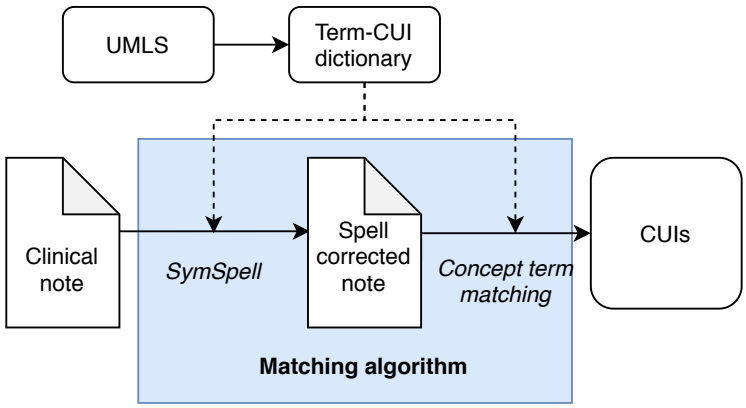


Figure 3.1: The concept finder program.

3.1.2.1 Matching Algorithm

The first part of the matching algorithm consists of running SymSpell to correct the spelling of the clinical notes. SymSpell corrects the spelling of a text by replacing unknown input words by words from a dictionary if they are similar enough. In this case, the dictionary used contains the words from the concept terms. Similarity is

[2]SymSpell, `https://github.com/wolfgarbe/symspell`
[3]`https://github.com/jrrr/conceptfinder`

measured as edit distance, i.e. the number of removals, insertions, or substitutions of characters required to modify one word into another. For example, the edit distance between "heart" and "heat" is 1. SymSpell has a running time that is independent of the number of dictionary entries, giving a fast enough performance for this application (see the SymSpell web page for more details). It supports frequency counts for the dictionary entries, so that if multiple dictionary entries have the same edit distance to the input word, the most frequent dictionary entry is used. However, since in this case the frequencies of the concept terms are unknown, the first entry with minimum edit distance is used. The maximum edit distance used when correcting words is a parameter and could impact the performance of the concept finder.

After SymSpell has corrected misspelled terms, concept term matching is performed using a tree of hash tables. The tree is built using a dictionary containing concept terms with corresponding CUIs. An illustration is shown in figure 3.2. The illustrated tree was created from 5 concept terms for 4 unique concepts: "heart attack", "mycardial infarction", "diabetes", "diabetes insipidus", and "diabetes mellitus". The matching algorithm proceeds as follows:

1. Start at the first input word, $i \leftarrow 1$.
2. Starting at position i find the longest possible match in the hash table tree. Let i_{match} be the last word of the match found.
   - If there is no match, perform step 2 with $i \leftarrow i + 1$.
   - If there is a match, remember the match and perform step 2 with $i \leftarrow i_{match} + 1$.
3. When there are no more input words, return all remembered matches.

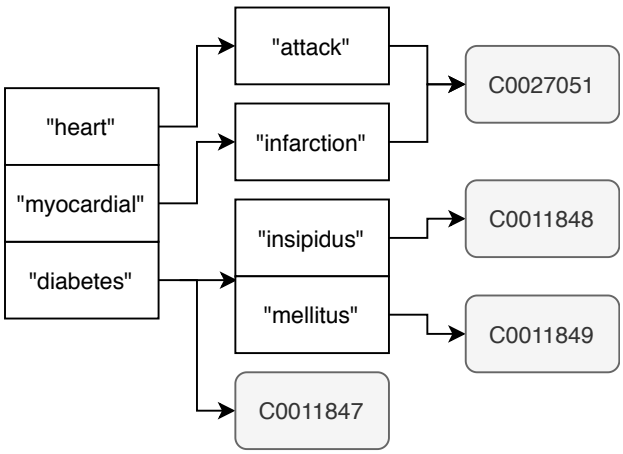


Figure 3.2: Illustration of the concept finder hash table tree. The boxes are elements of hash tables, adjacent boxes belong to the same hash table. Boxes with rounded corners are leaf nodes containing the CUI of the parent hash table entry, if it has any.

3.1.2.2 Term-CUI Dictionaries

UMLS specifies which semantic group(s) a concept belongs to. Many of the groups (e.g. "Birds") are likely irrelevant when analyzing EMRs. Thus, only concepts belonging to any of the following groups were added to both the English and Swedish term-CUI dictionaries:

- Disease or Syndrome
- Sign or Symptom
- Pathologic Function
- Clinical Drug
- Pharmacologic Substance
- Antibiotic

For the Swedish dictionary an additional group was added, because the Swedish EMRs are from a psychiatric department:

- Mental or Behavioral Dysfunction

Since words may be inflected in the clinical notes, the concept finder also supports word forms. For example, "heart" and "hearts" are two forms of the same word. These word forms are specified when building the term-CUI dictionary. This feature is likely more important for Swedish since the definite form is marked by inflection where in English it is marked by article ("hjärtat" corresponds to "the heart") and therefore is a more inflected language. Because of this and the lack of a morphological lexicon containing English word forms this feature was only used for Swedish.

SALDO [23] was used to lookup different word forms for the Swedish concept terms. A table was created mapping each word to its set of word forms. For example, both "heart" and "hearts" would be mapped to the set $\{$"*heart*", "*hearts*"$\}$. Then every word of the concept terms were replaced with the corresponding set of word forms.

It should be noted that different words can have overlapping word forms. This, however, is not allowed in the hash tree: each entry (word form) is only allowed to have *one* hash tree child. Initially it was thought to be possible to simply merge words with overlapping word forms, but this led to each word having on average over 5000 forms. Therefore, as a quick fix, when generating the word form lookup table, if a word form has been previously used for another word, that word form is not used for the current word. While potentially slightly decreasing the number of true positives it was deemed unlikely to increase false positives.

3.1.3 Extracting Embeddings from ClinicalBERT

The method of extracting embeddings from a pre-trained BERT model was described in section 2.4.2. Embeddings were extracted for the same CUIs as in the cui2vec pre-trained embeddings, in order to compare embedding quality and not CUI coverage. To extract an embedding for a concept, a single concept term is required,

while UMLS contains many terms for each concept. However, UMLS also contains a ranking of preference for concept terms (`MRRANK.RRF`), and the highest ranked concept term was used for each CUI. Following [15], embeddings were extracted by taking the sum of the outputs of the last 4 encoder layers when given a concept term as input. If the concept term consisted of several tokens, each with its own embedding, the mean of the token embeddings was taken as the concept embedding. An illustration of this process is shown in figure 3.3. The embeddings constructed this way will be referred to as the *ClinicalBERT embeddings*.

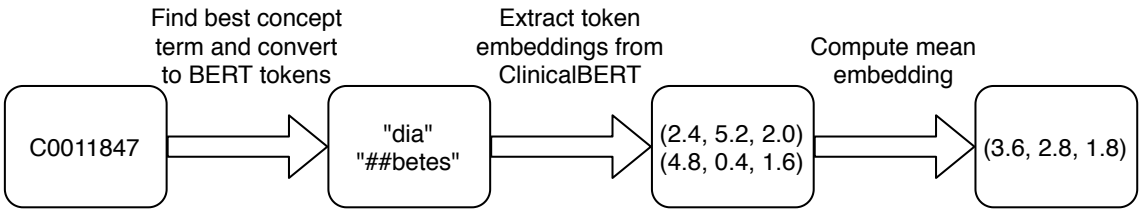


Figure 3.3: Illustration of the embedding extraction process.

3.1.4 Aggregating Concept Embeddings

The concepts found and mapped to CUIs were embedded using both cui2vec and ClinicalBERT embeddings. However, the models require fixed-length input, while there is a variable number of concepts in the clinical notes, as well as a variable number of clinical notes per hospital stay. The concept embeddings had to be aggregated into a fixed-length vector. Following [5], the concept embeddings were aggregated using element-wise *min*, *max*, and *mean*, concatenated together. The authors found that this method of aggregating worked well. This is illustrated in figure 3.4.

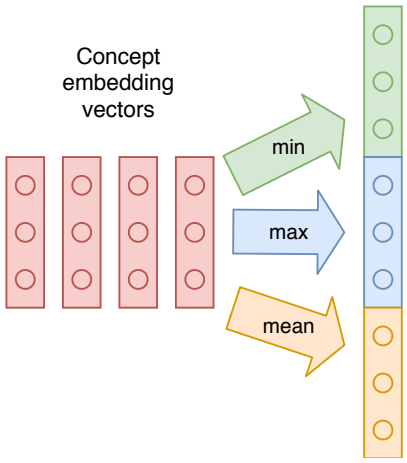


Figure 3.4: Illustration of aggregating multiple concept embeddings into a single vector.

3.2 Data Sets

Two sources of EMR data were used in this thesis. The first was the freely accessible MIMIC-III data set, containing ICU patients and English language clinical notes. The second was proprietary EMR data provided by Sahlgrenska University Hospital, concerning patients admitted for addiction care and Swedish language clinical notes. Predictions were made per *hospital admission*.

3.2.1 MIMIC-III

The MIMIC-III data set contains patients admitted to an ICU at Beth Israel Deaconess Medical Center in Boston, Massachusetts between 2001 and 2012 [6]. It contains information about patients who during a hospital admission also stayed in an ICU. The majority of the data pertains to the ICU stays, but there are also some data about the hospital admission, such as time of admission and discharge and the reason for admission. The data are de-identified to protect the patients' privacy.

The data from MIMIC-III that was used as input to the models in this thesis are clinical notes and physiologic measurements of 17 variables (such as blood pressure, for a complete list see [24]). The measurements are used as a baseline, to give an indication of the other models' performance. Information about admission and discharge times, whether an admission was planned or not, and whether the patient died in hospital were additionally used to determine the classes of the inputs.

The pre-processing, extraction of measurements, and task construction were all based on the publicly available code from [24]. The code used is published[4].

3.2.1.1 Pre-Processing

The pre-processing will be briefly described here (see [24] for more details). First several ICU stays are excluded:

1. Hospital admission with ICU transfers or multiple ICU stays.
2. ICU stays of patients younger than 18.

This resulted in 33,798 patients and 42,276 ICU stays. Note that since hospital admissions with multiple ICU stays were excluded, there is a one-to-one relation between hospital admissions and ICU stays, and the terms can be used interchangeably.

3.2.1.2 Input Features and Tasks

Several sets of input features were used:

- **cui2vec embeddings**: the concepts extracted from the clinical notes were embedded and aggregated as described in section 3.1 using the cui2vec pre-learned embeddings [14].

[4] `https://github.com/jrrr/mimic3-benchmarks/tree/emr-semantic-representations`

- **ClinicalBERT embeddings**: the same as cui2vec, only using the ClinicalBERT pre-learned embeddings [15].
- **Physiological measurements**: different physiological measurements taken from the patients during the ICU stays, selected by [24]. They use 17 basic variables, which are expanded with 6 statistics (minimum, maximum, mean, standard deviation, skew and number of measurements), using 6 different time ranges (first 10% of time, first 25% of time, first 50% of time, last 50% of time, last 25% of time, last 10% of time), resulting in 714 features.
- **Bag of Concepts**: binary vectors were created from the concepts extracted from the clinical notes (see section 3.1), as described in section 2.3.2.
- **Bag of Words**: binary vectors were created from the words in the clinical notes after stop words were removed, as described in section 2.3.2.

Additionally, ClinicalBERT and Bag of Concepts were concatenated together in order to discover if they contained any complementary information. ClinicalBERT and Bag of Concepts were also concatenated with physiological measurements in order to discover if concept embeddings are easier to combine with other data than Bag of Concepts. Since cui2vec is likely to perform similar to ClinicalBERT in this context, it was not used in combination with any other features due to limited time.

Two tasks were used:

- **In-hospital mortality:** whether the patient died during the hospital admission. This task is the same as in [24]. Only data from the first 48 hours of the ICU stay were used. However, since only the date and not the time of a clinical note is available in MIMIC-III, notes from the first 48 to 72 hours (depending on the time of day of admission) were used. Stays shorter than 48 hours, containing no events before 48 hours, or with an unknown length of stay were removed, resulting in 20,030 admissions. Of these approximately 13% resulted in in-hospital mortality.
- **Unplanned readmission:** whether the patient has an unplanned readmission to the hospital within 30 days of discharge. Note that this is *hospital* and not *ICU* readmission. The task is thus to use data from the ICU stay in order to predict hospital readmission. It would be more natural to predict ICU readmission but MIMIC-III does not state whether ICU admissions are planned or not. Data from the complete ICU stay were used for this task. Approximately 6% of the admissions resulted in unplanned readmission.

3.2.1.3 Evaluation

The split into train, validation and test sets selected by [24] was used. This split has the advantage that no patient is present in more than a single set. This ensures that potential overfitting to certain patients does not influence validation scores. The validation and test sets were approximately 15% of the total data, and the train set 70%. The validation set was used to select hyperparameters, and the test set was only used to produce the final scores at the end.

3.2.2 Swedish Addiction Care EMRs

The Swedish EMRs concern patients admitted to hospital for addiction care at Sahlgrenska University Hospital between 2014 and 2019. In contrast to MIMIC-III, this data set does not contain any physiological measurements. It contains data such as patient demographics, diagnoses, prescriptions, and clinical notes. Only the clinical notes were used. During pre-processing hospital admissions without any clinical notes or matched concepts were removed, resulting in 9438 admissions. Since this data set is proprietary, the code used cannot be published.

3.2.2.1 Input Features and Tasks

The input features used were the same as for MIMIC-III, except no measurements were available:

- **cui2vec embeddings**: the concepts extracted from the clinical notes were embedded and aggregated as described in section 3.1 using the cui2vec pre-learned embeddings [14].
- **ClinicalBERT embeddings**: the same as cui2vec, only using the ClinicalBERT pre-learned embeddings [15].
- **Bag of Concepts**: binary vectors were created from the concepts extracted from the clinical notes (see section 3.1), as described in section 2.3.2.
- **Bag of Words**: binary vectors were created from the words in the clinical notes after stop words were removed, as described in section 2.3.2.

For this data set, only one task was used:

- **Early readmission:** whether the patient was readmitted within 14 days of discharge. Approximately 14% of the admissions resulted in early readmission.

3.2.2.2 Evaluation

Since this data set was relatively small, 10-fold cross validation was used. The final scores were computed as the average of the scores from each fold, and a 95% confidence interval was computed from the standard deviation of the folds' scores. No test set was used. This was motivated by the fluctuating scores of the 1/10-sized folds. Thus, either the test set would have to be much larger, resulting in less training data, or random effects would impact the test scores to an intolerable degree.

3.3 Models and Training

Primarily two models were used: logistic regression and random forest, trained on the data sets described in section 3.2. Both are simple and fast models, while still principally different. It was believed that if neither improved by using concept embeddings, other kinds of models would also be unlikely to do so. Furthermore, two simple ANNs were developed to explore whether a similar but slightly more

complex model than logistic regression would improve performance, and whether a weighted sum of embeddings with learned weights could perform better than the fixed embedding aggregation described in section 3.1.4.

3.3.1 Logistic Regression

scikit-learn's[5] logistic regression implementation was used to perform the logistic regression. The `lbfgs` solver was used, with a stopping tolerance of 10^{-4}. ℓ^2 regularization was used, and the parameter λ was tuned manually to optimize ROC AUC. All training data except bag of concepts were standardized and missing values of the MIMIC-III measurement features were imputed to the mean of that feature in the training data set. The classes were balanced using the weighing technique described in section 2.5.7.

3.3.2 Random Forest

scikit-learn's random forest classifier implementation was used for the random forest model. The forest grew 200 trees (or estimators), and the maximum number of leaf nodes were tuned manually to optimize the ROC AUC. The classes were balanced using weights. As above, missing measurement values in the MIMIC-III data were imputed but no standardization was performed as this is not required for random forests.

3.3.3 Artificial Neural Network

Finally, two ANNs were implemented. The first was a simple feed-forward fully-connected ANN. Beginning from a single layer with a single neuron, i.e. equivalent to logistic regression, no improvement is observed with increasing neurons and layers during early tests.

The second consisted of an embedding layer that combined embeddings from an input bag of concepts using learnable weights, according to the equation

$$\mathbf{x}' = \sum_i x_i w_i \mathbf{e}_i \tag{3.1}$$

where x_i are the input features, w_i are the learnable weights, and $\mathbf{e}_i$ are the concept embeddings. This can be written more succinctly as

$$\mathbf{x}' = \Big(E(\mathbf{x} \odot \mathbf{w})^T\Big)^T \tag{3.2}$$

where $\odot$ represents element-wise vector product, $\mathbf{x}$ and $\mathbf{w}$ are row vectors containing the inputs x_i and weights w_i, respectively, and E is the matrix consisting of $\mathbf{e}_i$ as rows. However, early tests indicated that this ANN also did not improve performance over simple logistic regression. Because of this and time constraints, both ANNs were discontinued before the final, more elaborate testing.

[5] scikit-learn, `https://scikit-learn.org/stable/index.html`

Both ANNs were trained using the Adam optimization algorithm, with a learning rate of 10^{-3} and the ℓ^2 regularization parameter λ was tuned manually to optimize ROC AUC. Early stopping regularization was also used, and the learned network parameters that achieved highest ROC AUC during training was used to generate the final metric scores.

4

Results and Discussion

This chapter first presents the results of employing the methods described in chapter 3. Apart from the models' predictive performance, statistics of the extracted concepts and, when meaningful, the most significant features found by logistic regression are shown. Section 4.3 contains a discussion of the results, in particular in relation to the purpose of the thesis.

The following is a short explanation of the the tables. Each row shows the results of training and evaluating the model using the specified set of features. A plus sign in the features tab denotes concatenation of feature vectors. λ for logistic regression and ANN is the ℓ^2 regularization hyperparameter. MLN for random forest is the Maximum Leaf Nodes hyperparameter. Abbreviations are used for the different features:

- BERT denotes the ClinicalBERT embedded concepts.
- M denotes the physiological measurement variables.
- BoC denotes bag of concepts.
- BoW denotes bag of words.

All metrics except ROC AUC use a fixed decision boundary of 0.5, while ROC AUC aggregates all possible decision boundaries. Also note that the most important metric is ROC AUC. The hyperparameters were tuned to optimize this metric, occasionally at the expense of other metrics.

4.1 MIMIC-III

The number of notes per hospital stay in the MIMIC-III data set is shown in figure 4.2. For each number of notes on the x-axis, the y-axis shows how many hospitals stays that have that number of notes. The mean number of notes was 24.9 for all time, and 10.6 for the first 48-72 hours. The number of concepts per hospital stay, i.e. in all notes pertaining to a hospital stay, are shown in figure 4.3. The mean number of concepts was 352.0 total in all notes, 245.3 unique in all notes, 125.0 total in the first 48-72 hours, and 88.4 in unique in the first 48-72 hours. It is the in-hospital mortality task that only uses notes from the first 48-72 hours, while the unplanned readmission task uses all notes.

Figure 4.1 shows the lengths of the matched concept terms when extracting the

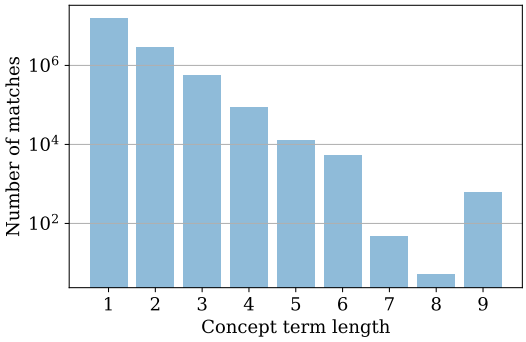


Figure 4.1: Concept term match lengths on the MIMIC-III data set.

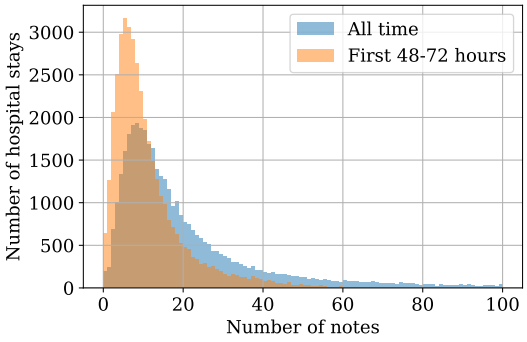
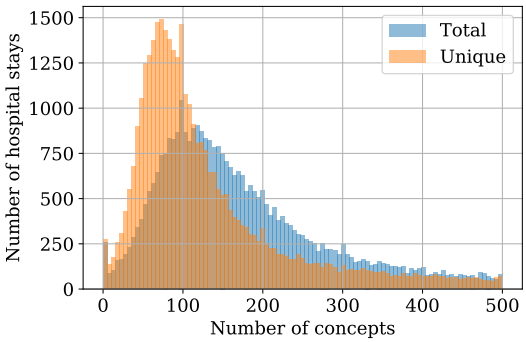
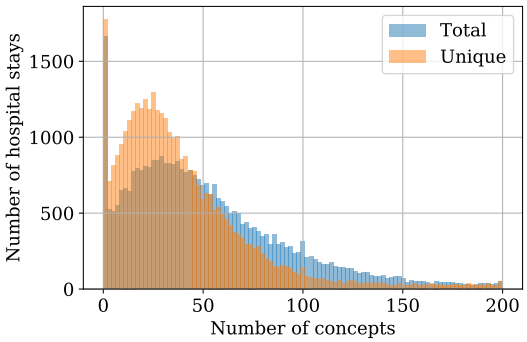


Figure 4.2: Number of notes per hospital stay in the MIMIC-III data set.



(a) In all notes.



(b) In notes from the first 48-72 hours.

Figure 4.3: Number of concepts per hospital stay in the MIMIC-III data set.

concepts from the notes associated with hospital admissions. The mean length was 1.22.

4.1.1 In-Hospital Mortality

This section contains the results of predicting in-hospital mortality on the MIMIC-III data. Table 4.1 shows the results of logistic regression, while table 4.2 shows the results of random forest. Tables 4.3 and 4.4 show the most significant (largest regression coefficient $\mathbf{w}$, see section 2.2.1) concepts and words, respectively, found by the logistic regression models using either bag of concept or bag of words.

Features	λ	Accuracy	Precision	Recall	F_1	ROC AUC
cui2vec	10^{-3}	0.691 **0.704**	0.236 **0.209**	0.650 **0.654**	0.347 **0.317**	0.735 **0.754**
BERT	10^{-3}	0.730 **0.728**	0.268 **0.231**	0.660 **0.679**	0.382 **0.345**	0.779 **0.782**
M	10^{-4}	0.784 **0.793**	0.345 **0.305**	0.721 **0.709**	0.467 **0.426**	0.845 **0.851**
BERT + M	10^{-4}	0.798 **0.801**	0.360 **0.318**	0.769 **0.781**	0.490 **0.452**	0.861 **0.869**
BoC	10^{-2}	0.735 **0.747**	0.260 **0.239**	0.598 **0.636**	0.362 **0.347**	0.773 **0.795**
BoC + M[1]	10^{-2}	0.813 **0.808**	0.375 **0.323**	0.736 **0.748**	0.497 **0.452**	0.862 **0.872**
BoC + BERT[2]	10^{-2}	0.746 **0.743**	0.281 **0.237**	0.647 **0.648**	0.392 **0.347**	0.789 **0.793**
BoW	10^{-3}	0.774 **0.783**	0.336 **0.297**	0.739 **0.736**	0.462 **0.424**	0.849 **0.855**

Table 4.1: Logistic regression in-hospital mortality prediction results on MIMIC-III. The first row is the score for the validation set, the second row is the score for the test set.

[1]Rescaled to have a standard deviation of $\sqrt{0.01}$. This results in the same optimal λ as for BoC (see section 2.5.6 for an explanation why).

[2]Rescaled to have a standard deviation of $\sqrt{0.1}$. This results in the same optimal λ as for BoC.

Features	MLN	Accuracy	Precision	Recall	F_1	ROC AUC
cui2vec	500	0.870 **0.894**	0.415 **0.480**	0.072 **0.114**	0.122 **0.185**	0.726 **0.748**
BERT	300	0.864 **0.885**	0.387 **0.386**	0.127 **0.156**	0.192 **0.222**	0.741 **0.766**
M	300	0.869 **0.883**	0.503 **0.461**	0.498 **0.488**	0.500 **0.474**	0.860 **0.869**
BERT + M	200	0.867 **0.875**	0.476 **0.421**	0.507 **0.502**	0.491 **0.458**	0.855 **0.864**
BoC	300	0.841 **0.867**	0.365 **0.377**	0.352 **0.393**	0.359 **0.385**	0.772 **0.803**
BoC + M	750	0.886 **0.902**	0.601 **0.571**	0.272 **0.277**	0.374 **0.373**	0.858 **0.863**
BoC + BERT	500	0.870 **0.896**	0.375 **0.538**	0.048 **0.089**	0.085 **0.153**	0.744 **0.763**
BoW	100	0.800 **0.813**	0.342 **0.305**	0.567 **0.570**	0.427 **0.397**	0.804 **0.812**

Table 4.2: Random forest in-hospital mortality prediction results on MIMIC-III. The first row is the score for the validation set, the second row is the score for the test set.

CUI	Description	Coefficient
C0003962	Ascites	0.20
C0024730	Mannitol	0.15
C0004238	Atrial fibrillation	0.14
C1705480	Vasopressin	0.12
C0032285	Pneumonia	0.12
C0012634	Disease	0.12
C0221423	Illness	0.12
C0013030	Dopamine	0.12
C0012359	Dilation	0.12
C0151603	Anasarca	0.11

CUI	Description	Coefficient
C0027497	Nausea	-0.18
C0008031	Chest pain	-0.13
C0016204	Flatulence	-0.09
C2741638	Stress ulcer	-0.09
C0030193	Pain	-0.08
C1956346	Coronary artery disease	-0.08
C0018681	Headache	-0.08
C2364135	Discomfort	-0.08
C0264956	Atheroma	-0.07
C0012833	Dizziness	-0.07

Table 4.3: Concepts with the largest coefficients from the bag of concepts logistic regression model on in-hospital mortality. Positive coefficient implies a positive correlation with in-hospital mortality, negative coefficient implies a negative correlation.

Word	Coefficient	Word	Coefficient
family	3.82	clear	-2.85
dnr	3.51	extubated	-2.50
ascites	3.17	extubation	-2.48
metastatic	2.73	normal	-2.41
arrest	2.59	pain	-2.40
dni	2.22	sda	-2.37
worsening	2.17	bradycardia	-2.08
expired	1.85	post	-1.99
pts	1.79	diet	-1.86
diffuse	1.76	insulin	-1.72

Table 4.4: Words with the largest coefficients from the bag of words logistic regression model on in-hospital mortality. Positive coefficient implies a positive correlation with in-hospital mortality, negative coefficient implies a negative correlation.

4.1.2 Unplanned Readmission

This section contains the results of predicting unplanned readmission on the MIMIC-III data. Table 4.5 shows the results of logistic regression. Table 4.6 shows the results of random forest. Table 4.7 shows the results of the two kinds of ANN. Both kinds of ANN use the ClinicalBERT embeddings and are described in section 3.3.3. "2×1" contains two layers with 2 neurons and 1 neuron, respectively. "weights" uses learnable embedding weights and then has a single layer with 1 neuron. Tables 4.8 and 4.9 show the most significant concepts and words, respectively, found by the logistic regression models using either bag of concept or bag of words.

Features	λ	Accuracy	Precision	Recall	F_1	ROC AUC
cui2vec	10^{-4}	0.646 **0.642**	0.095 **0.084**	0.557 **0.591**	0.162 **0.147**	0.648 **0.655**
BERT	10^{-4}	0.681 **0.677**	0.109 **0.096**	0.590 **0.614**	0.185 **0.166**	0.683 **0.689**
M	10^{-4}	0.623 **0.620**	0.094 **0.080**	0.601 **0.604**	0.163 **0.142**	0.659 **0.654**
BERT + M	10^{-4}	0.701 **0.699**	0.115 **0.102**	0.579 **0.607**	0.192 **0.174**	0.709 **0.705**
BoC	10^{-3}	0.708 **0.719**	0.115 **0.105**	0.560 **0.581**	0.190 **0.178**	0.692 **0.697**
BoC + M[3]	10^{-3}	0.726 **0.715**	0.128 **0.102**	0.595 **0.568**	0.210 **0.172**	0.714 **0.715**
BoC + BERT[4]	10^{-3}	0.712 **0.712**	0.112 **0.103**	0.535 **0.581**	0.186 **0.174**	0.687 **0.695**
BoW	10^{-4}	0.739 **0.740**	0.129 **0.115**	0.573 **0.601**	0.211 **0.194**	0.722 **0.727**

Table 4.5: Logistic regression unplanned readmission prediction results on MIMIC-III. The first row is the score for the validation set, the second row is the score for the test set.

[3]Rescaled to have a standard deviation of $\sqrt{0.1}$. This results in the same optimal λ as for BoC.
[4]Rescaled to have a standard deviation of $\sqrt{0.1}$. This results in the same optimal λ as for BoC.

Features	MLN	Accuracy	Precision	Recall	F_1	ROC AUC
cui2vec	300	0.931 **0.942**	0.086 **0.233**	0.014 **0.045**	0.023 **0.076**	0.656 **0.651**
BERT	400	0.938 **0.947**	0.000 **0.000**	0.000 **0.000**	0.000 **0.000**	0.671 **0.665**
M	100	0.855 **0.860**	0.127 **0.114**	0.234 **0.250**	0.164 **0.156**	0.682 **0.677**
BERT + M	300	0.936 **0.945**	0.185 **0.185**	0.014 **0.016**	0.025 **0.030**	0.690 **0.690**
BoC	100	0.786 **0.793**	0.126 **0.118**	0.421 **0.455**	0.194 **0.187**	0.675 **0.684**
BoC + M	200	0.916 **0.923**	0.203 **0.150**	0.128 **0.104**	0.157 **0.123**	0.697 **0.703**
BoC + BERT	300	0.934 **0.943**	0.140 **0.178**	0.016 **0.026**	0.029 **0.045**	0.672 **0.671**
BoW	500	0.939 **0.948**	0.000 **0.000**	0.000 **0.000**	0.000 **0.000**	0.703 **0.703**

Table 4.6: Random forest unplanned readmission prediction results on MIMIC-III. The first row is the score for the validation set, the second row is the score for the test set.

ANN	λ	Accuracy	Precision	Recall	F_1	ROC AUC
2 × 1	10^{-2}	0.609 **0.605**	0.102 **0.085**	0.690 **0.669**	0.178 **0.150**	0.682 **0.679**
Weights	10^{-2}	0.736 **0.744**	0.120 **0.107**	0.524 **0.532**	0.195 **0.178**	0.686 **0.688**

Table 4.7: Artificial neural network unplanned readmission prediction results on MIMIC-III. The first row is the score for the validation set, the second row is the score for the test set.

CUI	Description	Coefficient
C0037982	Spironolactone	0.14
C0151636	Ventricular premature contractions	0.13
C0032952	Prednisone	0.12
C0018946	Subdural hematoma	0.12
C0016382	Flushing	0.11
C0025853	Metoclopramide	0.11
C0543495	Albuterol sulfate	0.11
C0009014	Clonidine	0.11
C0520679	Obstructive sleep apnea	0.10
C0016410	Folic acid	0.10

CUI	Description	Coefficient
C0026056	Midazolam	-0.09
C0009566	Complication	-0.09
C0015672	Fatigue	-0.08
C0032825	Potassium chloride	-0.08
C0037580	Soft tissue swelling	-0.08
C0003864	Arthritis	-0.08
C0002962	Angina pectoris	-0.07
C0025598	Metformin	-0.07
C1956346	Coronary artery disease	-0.07
C0020443	Hypercholesterolemia	-0.07

Table 4.8: Concepts with the largest coefficients from the bag of concepts logistic regression model on unplanned readmission. Positive coefficient implies a positive correlation with unplanned readmission, negative coefficient implies a negative correlation.

Word	Coefficient
po	2.79
donation	2.43
subdural	2.42
trach	2.31
tracheostomy	2.26
admissions	2.18
organ	2.14
ama	2.09
dka	2.05
lasix	1.92

Word	Coefficient
expired	-3.48
osh	-3.03
comfort	-2.95
family	-2.63
outside	-2.16
post	-1.79
hospice	-1.69
cmo	-1.60
husband	-1.59
measures	-1.55

Table 4.9: Words with the largest coefficients from the bag of words logistic regression model on unplanned readmission. Positive coefficient implies a positive correlation with unplanned readmission, negative coefficient implies a negative correlation.

4.2 Sahlgrenska Univerity Hospital EMRs

The number of concepts per hospital admission in the Sahlgrenska University Hospital (SU) data set is shown in figure 4.4. The mean number of concepts per stay (excluding stays with 0 concepts) was 103.4 total and 30.3 unique. Note that the different notes were already concatenated in the data that was made available, so the number of notes per admission is unknown.

Figure 4.5 shows the lengths of the matched concept terms when extracting the concepts from the notes associated with hospital admissions. The mean length was 1.08.

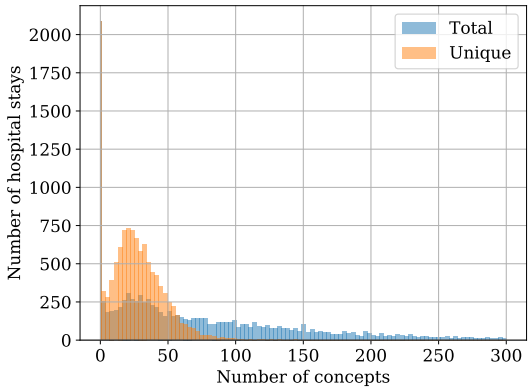


Figure 4.4: Number of concepts per hospital stay in the SU data set.

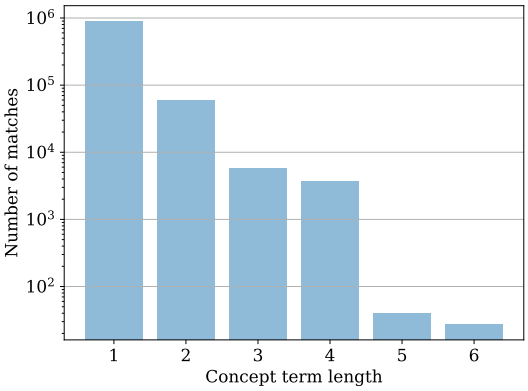


Figure 4.5: Concept term match lengths on the SU data set.

4.2.1 Early Readmission

This section contains the results of predicting early hospital readmission on the SU data. Table 4.10 shows the results of logistic regression. Table 4.11 shows the results of random forest. Tables 4.12 and 4.13 show the most significant concepts and words, respectively, found by the logistic regression models using either BoC or BoW. Note that some concepts, such as "psychological symptom" and "complication", are quite general and abstract.

Features	λ	Accuracy	Precision	Recall	F_1	ROC AUC
cui2vec	10^{-4}	0.593 ± 0.092	0.172 ± 0.065	0.508 ± 0.159	0.254 ± 0.077	0.568 ± 0.054
BERT	10^{-4}	0.611 ± 0.074	0.175 ± 0.086	0.486 ± 0.170	0.256 ± 0.111	0.577 ± 0.052
BoC	10^{-2}	0.633 ± 0.102	0.181 ± 0.081	0.468 ± 0.188	0.259 ± 0.106	0.590 ± 0.050
BoW	10^{-3}	0.816 ± 0.056	0.261 ± 0.071	0.163 ± 0.105	0.192 ± 0.059	0.603 ± 0.038

Table 4.10: Logistic regression early readmission prediction results on SU data. The $\pm$ denote the bounds of a 95% confidence interval.

Features	MLN	Accuracy	Precision	Recall	F_1	ROC AUC
cui2vec	50	0.772 ± 0.053	0.183 ± 0.134	0.184 ± 0.099	0.182 ± 0.112	0.571 ± 0.054
BERT	100	0.849 ± 0.058	0.211 ± 0.222	0.032 ± 0.041	0.054 ± 0.068	0.578 ± 0.069
BoC	50	0.728 ± 0.077	0.203 ± 0.119	0.330 ± 0.185	0.249 ± 0.136	0.591 ± 0.071
BoW	25	0.858 ± 0.019	0.134 ± 0.439	0.007 ± 0.037	0.012 ± 0.055	0.576 ± 0.032

Table 4.11: Random forest early readmission prediction results on SU data. The $\pm$ denote the bounds of a 95% confidence interval.

CUI	Description	Coefficient
C0338970	Emotionally unstable personality disorder	0.38
C0038436	Post-traumatic stress disorder	0.24
C0008073	Developmental disabilities	0.24
C1290899	Self abuse	0.21
C0338986	Atypical autism	0.20
C1263839	Developmental mental disorder	0.19
C0233397	Psychological symptom	0.15
C0025859	Metoprolol	0.15
C0221480	Recurrent depression	0.15
C0064636	Lamotrigine	0.14

CUI	Description	Coefficient
C0001975	Alcohol	-0.19
C0917801	Insomnia	-0.18
C0009566	Complication	-0.18
C0439857	Dependence	-0.18
C0559546	Adverse reactions	-0.16
C0018681	Headache	-0.15
C0036341	Schizophrenia	-0.13
C0011253	Delusions	-0.12
C0231230	Fatigability	-0.12
C2004491	Scar tissue	-0.12

Table 4.12: Concepts with the largest coefficients from the bag of concepts logistic regression model on early readmission. Positive coefficient implies a positive correlation with early readmission, negative coefficient implies a negative correlation.

Word	Coefficient
välkänd	0.09
personlighetsstörning	0.08
utskrivning	0.07
personal	0.07
lvm	0.06
utskriven	0.05
pox	0.05
därifrån	0.05
senast	0.05
flertal	0.05

Word	Coefficient
år	-0.07
alkohol	-0.07
inget	-0.06
avgiftning	-0.05
bruk	-0.05
dricker	-0.05
också	-0.05
anhöriga	-0.05
haft	-0.05
ibland	-0.05

Table 4.13: Words with the largest coefficients from the bag of words logistic regression model on early readmission. Positive coefficient implies a positive correlation with early readmission, negative coefficient implies a negative correlation.

4.3 Discussion

The optimized metric was ROC AUC, which was also used as the value of comparison. Comparing the models that use emebedded concepts, cui2vec and BERT, against BoC that uses unembedded concepts, it can be seen that BoC consistently outperforms the others. Clearly, then, *there is no benefit of using pre-learned concept embeddings in any of the models and data used*. Of course, other data could have yielded a different result. However, since the data sets used are quite different, from different departments (intensive care and psychiatry) and different hospitals, while still yielding similar results, it is likely that other EMR data sets would not be much different. Thus, despite using more advanced concept embeddings, this result is in line with the finding from [5], that concept embeddings do not improve performance for large enough data sets.

It is unclear *why* the embeddings do not improve performance. The reason could be that the pre-learned information from scientific articles, etc., are simply not very useful in the context of EMR data. Another reason could be that the embedding aggregation removes useful information, such as the presence and absence of individual concepts. However, combining the BERT embeddings with BoC did not perform better than using only the BoC, implying that either such information was not removed or it does not complement the embeddings. Also, ClinicalBERT is pre-trained on general text and additionally on the MIMIC-III data itself, yet performed better than cui2vec also on the data from SU. Since the concepts relevant for ICU care are likely very different from those relevant for addiction care, it seems probable that what ClinicalBERT learned from general text is more important than what it learned from the specifically clinical text of MIMIC-III. Thus, it seems more likely that the embeddings simply do not contain enough information useful for the tasks.

It is also possible that the models used are not capable of extracting information from the embeddings. However, two points speak against this possibility. Firstly, two quite different models (logistic regression and random forests) gave similar results. For instance, in contrast to logistic regression, random forests can combine features non-linearly. Secondly, a simple neural network, at first equivalent to logistic regression, was expanded in both width and depth to increase model complexity and introduce more non-linearities, and yet did not achieve better performance than logistic regression. Thus, it seems unlikely that embeddings would perform better compared to the other features if other models were used.

However, the methods developed might be useful for other purposes than producing the best predictive models. Although producing less accurate predictions, an advantage of the BoC model is that it can give more insight into the data than the BoW model. This is especially noticeable in the SU data. The words in table 4.13 are in many cases seemingly insignificant words (e.g. "också" - "also"), while the concepts in table 4.12 are mostly clearly relevant and seem to give a better understanding of which patients are readmitted early. Furthermore, the concepts and concept groups used by the BoC model can easily be selected so that the importance of different such groups can be investigated, using the methods developed in this thesis.

In hindsight, it would have probably been better to use notes from the first 24-48 hours instead of the first 48-72 hours for the in-hospital mortality task. As can be seen in table 4.4, many of the most significant words directly indicate that the patient *already* died, while the task was to predict which patients *would* die in the *future*. Since patients who died within the first 48 hours were already excluded, notes containing these words are likely from the first 48-72 hours. However, no concepts that indicate this seem to have been extracted, which likely explains some of BoW's advantage over BoC. Nevertheless, this does not affect the relative performance of BoC and the embeddings, so the most relevant result, that embeddings are not beneficial, should not be affected by this.

The gap between BoC and BoW could potentially be closed, or even surpassed, if the concept extractor was improved. For instance, a known problem with identifying concepts in clinical notes is the common use of abbreviations. This is also seen in the results. For example, in table 4.9 "dka" (diabetic ketoacidosis) is one of the most significant words, but this abbreviation is not part of the concept extractor dictionary and was thus not available to the BoC model.

As it stands, the concept extraction was only evaluated by how well the extracted concepts performed on the prediction tasks. It would be valuable to know how well it performs in other contexts. For example the accuracy compared to a data set of texts manually annotated with the present concepts. Such an evaluation was not possible since no such data sets were available. Furthermore, since extracting the concepts was the single most time consuming step, it was not possible to try different hyperparameter values. For example, should more semantic concept groups have been included? Or which is the optimal edit distance? However, even if the concept extraction was improved, it is likely that the embeddings would still perform worse than BoC.

The results would suggest that there is little promise in further research into using pre-learned semantic representations of biomedical concepts in predictive models. It is worth noting, however, that there is a relatively large difference between the cui2vec and the ClinicalBERT embeddings. Indeed, the difference between them is larger than the difference between BERT and BoC. Thus, if some other method of pre-learning embeddings manages to improve as much as ClinicalBERT improves on cui2vec, those embeddings would outperform BoC. Therefore there is some motivation for future work retrying embeddings if and when improved methods are developed. However, it is possible that a language model, for example BERT, trained solely on the target language would still perform better than using such embeddings (for example, ClinicalBERT achieved 0.768 ± 0.027 ROC AUC on 30-day unplanned readmission on MIMIC-III [15]).

Finally, many improvements to the somewhat rudimentary concept extractor are possible. In particular, due to the widespread use of abbreviations in clinical notes, a method of identifying such abbreviations would be a large improvement. Detection of abbreviations could potentially be improved by a more careful selection of concept terms from UMLS, or dictionaries of medical abbreviations could be used in addition to the concept terms. It should also be tested with annotated data sets in order to determine its performance outside of its use in predictive models, and gain insights

into further potential improvements.

5

Conclusion

The thesis set out to investigate whether there is any benefit of using pre-learned semantic representations of biomedical concepts with models predicting patient outcome from EMR data. In particular, it was to be investigated if it is possible and beneficial to use representations learned from sources in one language on EMR data in another language. A secondary goal was to develop the methods required to do this. To this end, models were developed to predict patient outcome on an English EMR data set, MIMIC-III [6] containing patients admitted to an Intensive Care Unit, and a Swedish EMR data set, provided by Sahlgrenska University Hospital and containing patients admitted to addiction care.

A concept extraction method was developed for English and Swedish, and which should be easy to adapt to most languages. Two sets of publicly available pre-learned semantic representations were tested, cui2vec [14] and ClinicalBERT [15], and used to represent the extracted concepts. The semantic representations were compared to several baseline models when predicting in-hospital mortality and early or unplanned hospital readmission. It was found that *using pre-learned semantic representation provided no benefit when predicting patient outcome.*

The main consequence is thus that other avenues of research should be pursued if predictive models on EMR data are to be improved. However, if improved methods of learning semantic representations of biomedical concepts are developed, retrying the thesis' methods might yield a different result.

The most significant positive contribution of the thesis is the concept extraction method. A fast, language-agnostic (except for the required language-specific dictionary) concept extractor program has been made publicly available[1]. This program can be used in any context where concepts need to be extracted from a large amount of text. In particular, it was discovered that the concept extractor can be used to select groups of concepts used in predictive models that, although not giving the most accurate predictions, can give useful insights into which of the selected concepts are most important in that particular case. This can provide an organisation more flexibility than a simple bag of words model.

[1] `https://github.com/jrrr/conceptfinder`

Bibliography

[1] J. Henry, Y. Pylypchuk, T. Searcy, and V. Patel, "Adoption of Electronic Health Record Systems among U.S. Non-Federal Acute Care Hospitals: 2008-2015." https://dashboard.healthit.gov/evaluations/data-briefs/non-federal-acute-care-hospital-ehr-adoption-2008-2015.php, May 2016. [Online; accessed 26-May-2020].

[2] E. Union, "eHealth adoption in primary healthcare in the EU is on the rise." https://ec.europa.eu/digital-single-market/en/news/ehealth-adoption-primary-healthcare-eu-rise, Jun 2019. [Online; accessed 08-May-2020].

[3] C. M. DesRoches, E. G. Campbell, S. R. Rao, K. Donelan, T. G. Ferris, A. Jha, R. Kaushal, D. E. Levy, S. Rosenbaum, A. E. Shields, *et al.*, "Electronic health records in ambulatory care — a national survey of physicians," *New England Journal of Medicine*, vol. 359, no. 1, pp. 50–60, 2008.

[4] F. K. Khattak, S. Jeblee, C. Pou-Prom, M. Abdalla, C. Meaney, and F. Rudzicz, "A survey of word embeddings for clinical text," *Journal of Biomedical Informatics: X*, p. 100057, 2019.

[5] S. Dubois, N. Romano, D. C. Kale, N. Shah, and K. Jung, "Effective Representations of Clinical Notes," 2017.

[6] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, "MIMIC-III, a freely accessible critical care database," *Scientific data*, vol. 3, p. 160035, 2016.

[7] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning Series, MIT Press, 2012.

[8] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The expressive power of neural networks: A view from the width," in *Advances in neural information processing systems*, pp. 6231–6239, 2017.

[9] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, p. 9, 2016.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint*

arXiv:1810.04805, 2018.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," 2017.

[13] B. Shickel, P. J. Tighe, A. Bihorac, and P. Rashidi, "Deep EHR: a survey of recent advances in deep learning techniques for electronic health record (EHR) analysis," *IEEE journal of biomedical and health informatics*, vol. 22, no. 5, pp. 1589–1604, 2017.

[14] A. L. Beam, B. Kompa, I. Fried, N. P. Palmer, X. Shi, T. Cai, and I. S. Kohane, "Clinical concept embeddings learned from massive sources of multimodal medical data," *arXiv preprint arXiv:1804.01486*, 2018.

[15] K. Huang, J. Altosaar, and R. Ranganath, "ClinicalBERT: Modeling clinical notes and predicting hospital readmission," *arXiv preprint arXiv:1904.05342*, 2019.

[16] L. De Vine, G. Zuccon, B. Koopman, L. Sitbon, and P. Bruza, "Medical semantic similarity with a neural language model," in *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pp. 1819–1822, 2014.

[17] A. R. Aronson and F.-M. Lang, "An overview of MetaMap: historical perspective and recent advances," *Journal of the American Medical Informatics Association*, vol. 17, no. 3, pp. 229–236, 2010.

[18] D. Demner-Fushman, W. J. Rogers, and A. R. Aronson, "MetaMap Lite: an evaluation of a new Java implementation of MetaMap," *Journal of the American Medical Informatics Association*, vol. 24, no. 4, pp. 841–844, 2017.

[19] G. K. Savova, J. J. Masanz, P. V. Ogren, J. Zheng, S. Sohn, K. C. Kipper-Schuler, and C. G. Chute, "Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications," *Journal of the American Medical Informatics Association*, vol. 17, no. 5, pp. 507–513, 2010.

[20] S. Yu, T. Cai, and T. Cai, "NILE: Fast Natural Language Processing for Electronic Health Records," 2013.

[21] O. Bodenreider, "The unified medical language system (UMLS): integrating biomedical terminology," *Nucleic acids research*, vol. 32, no. suppl_1, pp. D267–D270, 2004.

[22] T. Benson and G. Grieve, *Principles of Health Interoperability: SNOMED CT, HL7 and FHIR*. Health Information Technology Standards, Springer International Publishing, 2016.

[23] L. Borin, M. Forsberg, and L. Lönngren, "SALDO: a touch of yin to WordNet's yang," *Language resources and evaluation*, vol. 47, no. 4, pp. 1191–1211, 2013.

[24] H. Harutyunyan, H. Khachatrian, D. C. Kale, G. V. Steeg, and A. Galstyan, "Multitask learning and benchmarking with clinical time series data," *arXiv preprint arXiv:1703.07771*, 2017.