Vrije Universiteit Amsterdam

Master Thesis Business Analytics

Deep Learning and transfer learning in Biomedical relation extraction

Author:
I.A. Wensing

Supervisors:
MSc. M. Gullaksen
Prof. Dr. S. Bhulai

Second reader:
Dr. M. Hoogendoorn

Date
February 17, 2020

Deep Learning and transfer learning in Biomedical relation extraction

Author:
I.A. Wensing

Supervisor:
Prof. Dr. S. Bhulai

External Supervisor:
MSc. M. Gullaksen

Second reader:
Dr. M. Hoogendoorn

Vrije Universiteit Amsterdam
Faculty of Science
Business Analytics
De Boelelaan 1111
1081 HV Amsterdam

Accenture Amsterdam
Applied Intelligence
ITO
Gustav Mahlerplein 90
1082 MA Amsterdam

Acknowledgements

This thesis has been written as the graduate project to fulfil the requirements for the master Business Analytics at Vrije Universiteit Amsterdam. The thesis is written during a 6 month internship at Accenture Applied Intelligence.

First of all, I would like to thank my supervisor Martin Gullaksen from Accenture, the weekly meetings we had were always very useful. Furthermore, my supervisor Sandjai Bhulai from the VU for the feedback on my thesis and providing me with good suggestions during my thesis. Finally, I would like to thank my boyfriend Jan Tijink for taking the time to read my thesis and giving substantive feedback.

Abstract

Biomedical literature is growing exponentially every year, making it harder for researchers to find relevant literature for their research. With this huge growth in biomedical information, the challenge of keeping up to date with new discoveries is getting extremely hard. Databases consisting of structured biomedical knowledge are manually curated and they are often unable to keep pace with the rich amount of information available in fast growing biomedical texts. To cope with this huge amount of literature, text mining and knowledge extraction methods can be very useful.

Biomedical relation extraction is the task of extraction biomedical relations from biomedical text. Existing methods mostly rely on hand-crafted features, and by the lack of annotated data, the methods have inadequate performance. The existing methods are mainly applied to a single application in Healthcare, for instance focusing on Drug-Drug Interactions. In this thesis a method is proposed to extract different types of biomedical relations by joining data from existing knowledge bases to unstructured text from *PubMed*. By using this method, a large dataset was created that contained information about drugs and diseases and different types of relations like *Treats*, *Causes* and *Interacts*.

Recently, Neural networks have shown to achieve better or similar performance as models that depend on hand-crafted features. With the current trend of transfer learning in NLP (BERT), neural networks can also be used without the requirement of a large annotated dataset. In this thesis, we propose several methods that are all based on embedding features and do not require any explicit feature engineering. Our experimental results show that especially the transfer learning models have very promising performance on this task. The performance of this model outperforms all of the Deep learning models that before achieved state-of-the-art performance on many biomedical NLP tasks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the large number of biomedical research articles published every day, new information in the biomedical domain is growing exponentially every year [18], making it harder for researchers to find relevant literature for their research. As of December 2018, PubMed consisted of more than 29.1 million records going back to 1966 [29]. With this huge growth in biomedical information, the challenge of keeping up to date with new discoveries is getting extremely hard. Aside from the fact that there is simply too much information to process, the information is also very specific, so most of the time only a specialist can truly understand the article. To help researchers cope with this huge amount of literature, text mining and knowledge extraction methods can be very useful.

Text mining is the process of exploring and analysing large amounts of unstructured text data. By finding patterns and trends in the text, information can be derived. Text mining is distinct from Natural Language Processing (NLP), where NLP is concerned with the interaction between computers and languages. Text mining and knowledge extraction are concentrating on specific problems, where in the process NLP techniques may be used.

Knowledge extraction is the task of structuring unstructured data, this is done by creating a knowledge base derived from text. This knowledge base represents knowledge in a structured form, from which computers can retrieve existing information or even deduce new knowledge. DBpedia [22] and YAGO [40] are examples of knowledge bases with connections covering almost the entire internet. These knowledge bases are currently the key for most search engines on the internet. Here users do not have to stroll around a site to search for their answer, but with a simple query they can find their answer immediately. For instance, when searching for the birth-place of Obama, the answer will immediately pop-up. This might not seem very complicated, however, imagine that before the search engine used a knowledge base, you had to search for the birth-place on some site yourself.

In order to generate a knowledge base, relations between two entities are extracted, where an entity can be e.g., a person, date, organisation, location, values, medical names. A relation can be practically any connection between two entities. In the biomedical domain, the entities mainly describe diseases, drugs, symptoms, genes or other biomedical terms. The relations can be a drug that is used as a treatment for a disease or a disease that can cause a symptom. These relations are mainly written down in documents, and the goal is to extract these relations. An example of a sentence that describes a biomedical

relation is given below:

> Both patients had a history of RA (disease) and were being treated with Methotrexate (drug).

There are many different approaches to extract knowledge from text. In early research, relation extraction was mainly done by pattern based methods ([7], [2]), where patterns would be extracted by looking at similarities in sentences that would describe similar relations. More recently, several Machine Learning methods were used to solve the problem. The problem with Machine Learning models is they require a large annotated dataset in order to perform well. In the last couple of years, some annotated data sets came available [38], making it possible to train more advanced models like Deep Learning methods. Even more recently, Google introduced BERT [12], BERT is a transfer learning approach for natural language processing applications. BERT achieved state-of-the-art performance on almost every NLP task including relation extraction.

In this thesis we will investigate the use of deep learning models and transfer learning in the field of biomedical relation extraction. In order to do this, a large annotated dataset is used and combined with data obtained by distant supervision. The research question to achieve this is as follows:

> *Is it possible to extract relations between entities in biomedical literature by using deep learning or transfer learning?*

The thesis is structured in the following manner: In Section 2 the available literature on the subject is discussed. Section 3 describes the dataset, how it is created, as well as some data analysis. Next, in Section 4 the outline of the methodology and set-up of the experiments are presented. The results are shown and discussed in Section 5. Finally, the conclusions of the research are drawn in Section 6.1 and further research options are discussed in Section 6.2.

Chapter 2

Literature review

In this section we will dive into the theory and literature used in the research. First, an overview is given of which NLP processes are used, then relation extraction will be explained in more detail. Different approaches on how to handle problems where there is little annotated data available are discussed. Finally, different models that can be used in relation extraction will be explained.

2.1 Text mining

In general terms, Text Mining can be defined as the process of exploring and analysing large amounts of text data. Information can be derived from the text by finding patterns and trends in the data. This can be accomplished by using different methodologies like Natural Language Processing (NLP). NLP is an Artificial Intelligence component where the computer learns to understand and analyse human language. To understand human language, NLP applies several techniques to find the meaning of a word or sentence. In order to perform more complex NLP tasks, first some basic linguistic processing steps are performed (tokenisation, sentence splitting, parsing etc.). The output of these linguistic processing steps can be used in more complex language processing algorithms like sentiment analysis, relation extraction or Named Entity Recognition.

2.1.1 Linguistic Processing

To use the more complex language processing algorithms, more basic linguistic processing tasks should be completed first. The linguistic processing tasks consists of tokenisation, sentence splitting, Part-of-speech tagging and parsing. These will now be explained in more detail.

tokenisation is the task where text is split up in units, called tokens. These tokens are the words, numbers and symbols in a text. In most languages these tokens are separated by a white space. tokenisers are depending on the language, but also on the type of text, i.e., there are specific tokenisers for biomedical text mining. Since most linguistic processes like POS-tagging and stemming require tokens as input, it is very important to use a good tokeniser [46]. Errors made in this step can have impact on the performance of more complex tasks like relation extraction or Named-Entity Recognition.

Sentence Splitting is the process of determining the boundaries of a sentence. This is mainly determined by the punctuation like full stops, commas or question marks. This may seem a trivial task, however, in many cases this is not as straight-forward as thought. For example, when names (i.e., Mr.) or abbreviations are in a sentence it is hard to split the sentence by punctuation rules. Sentence splitters have various approaches on how to solve this issue, like using a list of abbreviations and specific rules for full stops after specific words. It might be necessary to use a sentence splitter specifically trained on biomedical text, since different abbreviations are used than in regular text documents.

Part-of-speech tagging is the process of tagging words with their part of speech tag (i.e., noun, verb, adjective). The POS-tag of a word depends on the syntactic context of the word. A POS-tagger assigns a tag to the word based on the word itself and its surrounding words. Currently, most POS-taggers are Machine Learning models, since it is hard to cover all of the grammatical rules. These models are trained on huge datasets such as the *Wall Street Journal* corpus.

Parsing is concerned with deriving the syntactic structure of a sentence, based on grammar. It mainly describes how elements in a sentence are related to each other. There are different types of parsing techniques, one of them is dependency parsing. Dependency parsing extracts a dependence parse between different words, this dependency parse describes the grammatical structure of the sentence. By using the output of dependency parsing, the *shortest dependency path* between two words can be derived. This is a sequence of words with the shortest path between the two words based on the dependency parse.

2.1.2 Representation methods

After the text is processed by several linguistic processes, the data consists of several tokens, POS-tags and dependency relations. To use this information in some complex algorithms, the data should first be represented in a numerical format. There are different approaches on how to represent a piece of text as a fixed-length vector:

Bag of Words is an option to represent the sentences in vector format. The length of the vector is equal to the full size of the vocabulary, the value in the vector counts how often the word occurs in the sentence, an example is given in Figure 2.1. By using this representation, the machine can understand the sentence. This approach is used in many NLP tasks, however, this method has three major downsides:

1. The representation loses all of the grammatical information of the sentence, since there is no order involved anymore. The context of the word is not included in the representation.

2. Most of the values in the vector are zero, and therefore the vector is highly sparse.

3. The representation is biased to the words that occur more often.

The last point is solved by introducing the *term frequency–inverse document frequency* weight (TF-IDF). The weight is a statistical measure that evaluates how relevant a word is in a document. It measures whether a word is occurring more often in a document or

sentence than in general, so if words occur very frequently in every document, this will not impact the representation. The value of word x in document y is given by:

$$w_{x,y} = tf_{x,y} \cdot log(\frac{N}{df_x}) \tag{2.1}$$

where $tf_{x,y}$ is the frequency of word x in y, df_x is the number of documents containing the word x and N the total number of documents.

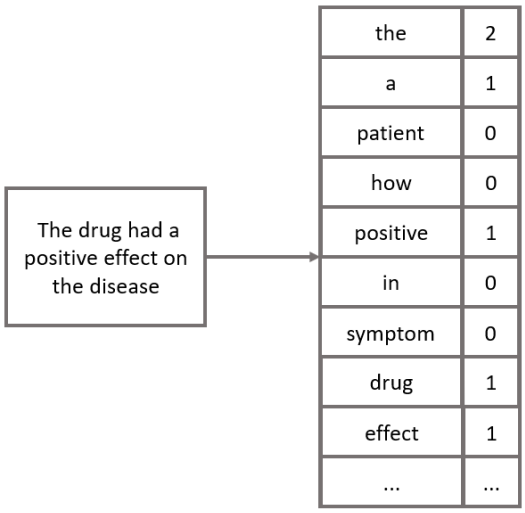


Figure 2.1: A Bag-of-Words representation of a sentence.

Word-Embeddings is a technique where words are mapped to a vector of real numbers. It tries to map the full vocabulary to a lower dimensional vector space. This will increase the efficiency of the network. Embeddings make use of a fully-connected layer, and the final embedding follows from a matrix multiplication between the one-hot encoded vector of the words and an embedding weight matrix. By training this model it uses the context of the words, since Firth [14] claims that a word is characterised by the company it keeps. By using this information to train the model, the syntactic meaning of the words is also covered by the model. This guarantees that both syntactic and semantic similar words are located in similar locations in the vector space. In most cases, a word embedding model is trained on a large dataset, where the model is normally trained on articles and text files in one domain. In Figure 2.2 an example is given for different biomedical words in a two-dimensional space. Words that have a similar meaning will have coordinates close to each other. The word embeddings can capture the meaning of a word, however, words can have different meaning when they occur in different contexts. The word embeddings can only capture one meaning per word.

2.1.3 Named Entity Recognition

Named Entity Recognition (NER) is the process of identifying entities in text, traditionally the set of named entities consists of persons, organisations, locations, and date and time expressions. In the case of biomedical NER, the goal is to identify biomedical entities,

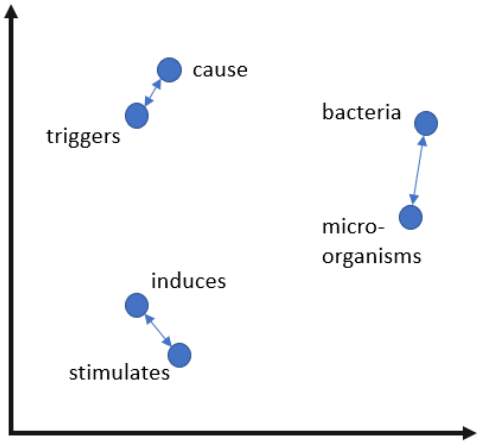


Figure 2.2: An example of biomedical words in a word embedding space.

i.e., drugs, diseases or proteins. When information is extracted from biomedical text, the NER task is very important since it detects the entities involved in a possible relation. Research [39] has shown that finding biomedical named entities is much more difficult than finding normal named entities because technical biomedical terms can have unusual characteristics. Furthermore, different names can refer to the same entity and therefore extra knowledge about the entity is needed in order to recognise the right entity.

The different approaches to NER can roughly be divided into two groups; rule-based and machine learning methods. Rule-based models have a list (called gazetteer) of known entities (companies, locations, etc.). Using a list of rules, it is determined whether an entity in the text refers to the same entity as in the gazetteer. When only the gazetteer is used to recognise the entities (check whether a word exists in the gazetteer), several entities names are too ambiguous, i.e., a name of a city can refer to the location, but it can also refer to a company or a football club in that city. Therefore gazetteers are combined with linguistic pre-processing information such as POS-tags, capitalisation and contextual information. When a machine learning model is used as an approach for NER, the entities are tagged by extracting a feature set from the text. These features can include Morphological features like capitalisation and special characters, POS-tags, context features, syntactic information from the parse of the sentence and word embeddings. Song et al. [39] compares different NER methodologies for biomedical documents. Several machine learning approaches are compared, where the conditional random fields (CRF) performed best.

2.2 Relation extraction

Relation extraction is the task of extracting structured semantic relationships from an unstructured text. A semantic relation is a relation between two entities. When two entities are recognised by Named Entity Recognition, relation extraction is the next task. The input of the relation extraction task the dataset should consist of several sentences including an annotated relation. When an annotated training set is available, several features can be extracted from the sentence (i.e., tokens, POS, Named entities), these can be

used for the training of a model. The label used for the model is the relation. This is a classification task, where all of the relations that occur in the training set can be predicted.

Known relations are collected in a *knowledge graph*. In such a graph, structured information of entities and their relations are encoded. A knowledge graph can contain millions of entities and billions of relations [23], but they are still far from complete. By using relation extraction new information can be discovered to expand the knowledge graph.

There are several approaches to the relation extraction task. In early days, bootstrapping methods such as DIPRE [7] and Snowball [2] were introduced. In bootstrapping methods they start off with a set of seed facts, from these seed facts patterns are extracted, and with these patterns new tuples are extracted. This is an iterative process where eventually many new relations can be extracted. A problem with these methods is that there is no information about the confidence of the relations. Furthermore, the precision tends to be low. Recently, the most used approaches are supervised learning methods. Features like word embeddings, dependency paths, chunks, or parse trees are extracted from the text and used as input for the classifier. These methods can achieve very high accuracy, however, the lack of data is a big problem. Annotating the data is very time consuming and expensive and therefore distant supervision (Section 2.3) approaches were introduced. This gave the opportunity to create data on large scale, making it possible to use deep learning models for the relation extraction task.

Many literature studies perform research in relation extraction for biomedical research, in particular the interaction between two drugs where the task is called drug-drug Interaction extraction (*DDI-extraction*), since a large annotated dataset is available for this specific task [36]. Early studies on this dataset mainly use feature-based methods, where syntactic and lexical features are extracted from the text and used as input for the classifier, in most cases Support Vector Machines are used. Björne et al. [4] tested different feature sets, where syntactic features were enhanced with features extracted from public databases with domain knowledge. Kim et al. [19] introduced a rich feature-based method to extract DDIs. The main challenge in feature-based models is to select the right features for the task. Feature-extraction is a time-consuming task and also skill dependent. The syntactic parse tree and dependency graph contain much information for the relation extraction task, therefore some papers ([37], [11]) introduce kernel-based methods. The kernel-based methods are better in capturing the syntactic information from the dependence graph and parse tree than the feature-based models. However, the performance of these methods is highly dependent on the selected feature set and the kernel function used.

In recent years, Deep neural networks gave promising performances on the DDI-extraction task. In Deep neural networks, the features are learned automatically by using word embeddings. Zhao et al. [51] proposes a Convolutional Neural network that outperformed all previously introduced methods. This led to more research using different Deep neural networks, Sahu et al. [34] applied a Long Short Term Memory network (LSTM) model with attention pooling. Zhang et al. [50] explored a Recurrent Neural Net including the shortest dependency path. Currently, state-of-the-art methods make use of a combination of different Deep neural networks ([49],[41]). Sun et al. [41] currently outperform all other methods by using a Recurrent Hybrid Convolutional Neural network, where con-

textual information is fused with the word embeddings. In Section 2.4 all of these methods are explained in more detail.

Most recent developments in natural language processing and in particular relation extraction, is the release of BERT [12]. BERT is a pre-trained model that can be used for several NLP tasks. It almost directly achieved state-of-the-art performance on all NLP tasks including relation extraction. This is further discussed in Section 2.6.

2.3 Distant Supervision

In most relation extraction problems, there is no annotated dataset available. Since most recent relation extraction methods are based on supervised learning, it is necessary to obtain or create an annotated dataset. Manually annotating data is very time-consuming, furthermore, in more complex domains only an expert can annotate the data, making it even more expensive. When no annotated data is available, distant supervision [26] can be a solution. In distant supervision relations are extracted from an existing knowledge base. Sentences that contain both entities from the relation in the knowledge base are labelled with the relation as label. In this approach they make use of the *distant supervision* assumption:

> If two entities participate in a relation, all sentences that mention these two entities express that relation.

Distant Supervision seems like a very efficient method to create large amounts of training data, however, there are also some downsides to this method. There is no guarantee that the data is correctly labelled, and therefore the data can consists of noisy samples. When two entities are related, it is likely that they appear in the same sentence without describing the relation, because they are related to the same topic. Therefore, Riedel et al. [32] relaxed the distant supervision assumption by introducing *expressed-at-least-once* assumption:

> If two entities participate in a relation, at least one sentence that mentions these two entities might express that relation.

This assumption makes the problem much more difficult, since it is unknown which of the sentences describes the relation. However, by using a ranking model they showed that they could reduce the model errors by 33%. Several other methods were introduced to handle the noise in the data, where most recently Deep neural networks were used to select the most plausible instances from the data [31]. Since most Machine Learning models need a large amount of data to achieve good performance, distant supervision can be very useful because it can generate large amounts of data.

2.4 Neural networks for relation extraction

With the increasing amount of data available and the growth in computational power, Artificial Neural networks *(ANN)* went through a huge rise in popularity. Neural networks are computer models based on the neural networks of the human brain. Neural networks can learn from raw features without complicated feature engineering. When a Neural

network is used for an NLP-task, mostly word-embeddings in combination with other features are used. A neural network consists of several layers, where every layer contains a number of neurons, the neurons send information to the next layer. All the neurons have weight inputs that are adjustable parameters during the training and an activation function to define the output.

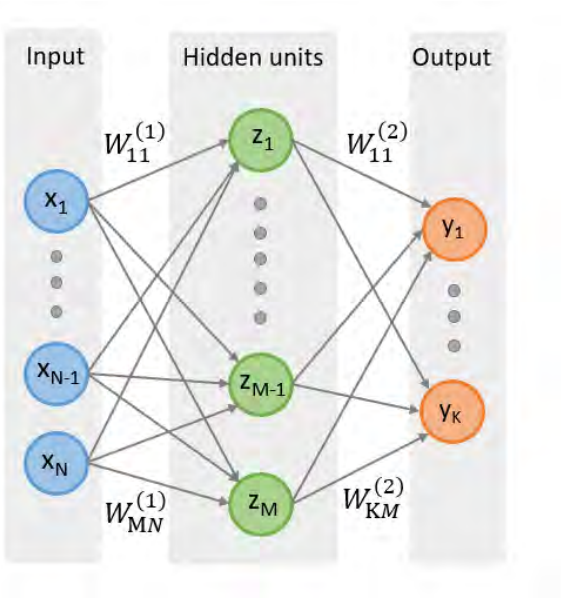


Figure 2.3: A neural network with one hidden layer.

2.4.1 Forward Propagation

The initial information in the network is given by the input X, this information then goes through the network and finally produces the output y, this process is called Forward Propagation. In Figure 2.3 an example of a neural network with multiple input nodes, two output nodes and one hidden layer with multiple nodes is shown. The layers are connected by the weights W, where $W^{(1)}$ represent the weights between the input layer and the first hidden layer. The weights between the layer l and layer $l+1$ are defined by W^l. The values of the hidden layer depend on the values of X, where every neuron in the hidden layer is a linear combination of the values of X, defined by the weights of the neurons, a bias term and an activation function h_l. The most simple activation function is the linear activation, where the transformation is linear. However, in order to learn more complex patterns in the data non-linear activation functions are used. There are different types of activation functions like *Sigmoid*, *Rectified Linear Unit* (ReLU) and *Hyperbolic Tangent* (TanH). A problem where both Sigmoid and Tanh suffer from is that the gradient can vanish or explode, making it harder for the algorithm to learn. ReLU does not have this problem since it prevents the gradients from moving towards zero. If the output of the ReLU returns a negative value, the gradients is set equal to zero, and therefore the weights are not updated. Only the neurons with a positive output of the ReLU function need to be updated, making the training of the model very easy and fast. The values of the hidden layers are calculated by the weights and the activation function in Equation 2.2.

$$a_j = \sum_{i=1}^{N} w_{ji}^{(1)} x_i + w_{j0}^{(0)} \tag{2.2}$$

$$z_j = h_0(a_j) \tag{2.3}$$

The value of the output is determined by the values of the last hidden layer and the final activation function σ:

$$y_k(x, w) = \sigma \left(\sum_{j=1}^{M} w_{kj}^{(2)} h_0 \left(\sum_{i=1}^{N} w_{ji}^{(1)} x_i + w_{j0}^{(0)} \right) + w_{k0}^{(2)} \right) \tag{2.4}$$

When more layers are considered, we can generalise this term by a vector representation in Equation 2.5, where the output a^l of layer l is given by:

$$z^l = W^l a^{l-1} + b^l \tag{2.5}$$

$$a^l = h^l(z^l) \tag{2.6}$$

2.4.2 Loss Function

To evaluate the predictions made by the model, the loss can be calculated. The value of the loss function is the value that needs to be minimised by the model. The loss function must consider all aspects of the model and put it into one single number that describes the performance of the model. Therefore, it is very important to select a loss function that can capture the properties of the model, but should also be interpretable by stakeholders. The value of the loss function depends on the realised value y and the predicted value $\hat{y}$. The loss function depends on the nature of the output, for instance regression problems mostly use *Mean Squared Error* (MSE). In the relation extraction task, the output is categorical and in most cases there are multiple classes. *categorical cross-entropy* described in Equation 2.7 is used when the output values are categorical. N is the number of samples and C is the number of classes.

$$E_{CCE} = \sum_{i}^{N} \sum_{j}^{C} y_{ij} \log(\hat{y}_{ij}) \tag{2.7}$$

2.4.3 BackPropagation

Originally, *backpropagation* was introduced in 1970, however after David Rumelhart, Geoffrey Hinton, and Ronald Williams [33] published their findings, backpropagation finally gained recognition. They showed that training neural networks by using backpropagation is much faster than previous approaches. Currently, backpropagation is the main method to efficiently train a Neural network.

The main idea behind backpropagation is to minimise the loss function by changing the weights in the network. It is an iterative approach, approaching a local minimum, however it does not guarantee a global minimum.

2.4.4 Convolutional Neural network

A *Convolutional Neural network* (CNN) is another type of Neural network mainly used in computer vision, but recently also applied in NLP-tasks. The difference between a standard Artificial Neural network and a CNN is that in an ANN all neurons are connected to all neurons in the next layer (called a fully connected layer), where a CNN uses different types of layers. The different layers can be a convolutional layer, a pooling layer or the fully connected layer appearing in the ANN as well.

The convolutional layer is the most important layer of the CNN, it is a sliding window function over a matrix. The sliding window is called a filter, the main idea of the filter is to summarise the value of an element and its surrounding elements. When we consider the example of image recognition, filters can for instance recognise edges in the image. In the next layer the edges can then be combined to find shapes, and finally the shapes can be combined to detect faces or other objects.

A pooling layer is typically applied after the convolutional layer. Pooling is needed for dimension reduction, the pooling layer will summarise the results of the filter. Most common pooling operations are the max-pooling, and the mean-pooling.

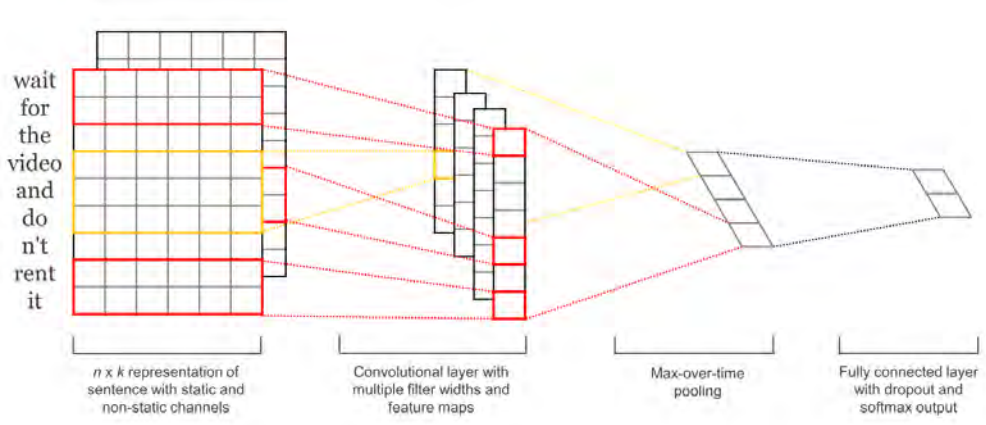


Figure 2.4: In a CNN the data is transformed by a filter to recognise patterns, then a pooling layer is applied to reduce dimensionality. [20]

When a CNN is used for an NLP-task, the inputs are not consisting of pixels, but they exist of sentences or documents represented as a matrix, by using word embeddings for instance. In most cases every row in the matrix represents one single word. When a filter is applied, the filter is mostly looking at the full row (one word) and the number of words that the filter considers can differ. So the filter size is similar to the word-embedding dimensionality and the number of words considered in the filter. CNNs has successfully been applied for different NLP-tasks, and can learn important features from sentences. This is visualised in Figure 2.4.

2.4.5 Recurrent Neural network

A *Recurrent Neural network* (RNN) is a specific type of Neural network that can remember the previous input objects, making it possible to learn temporal sequences. This is important when text data is considered, since the meaning of a word mostly depends on the previous words in the sentence. An RNN has loops within the network to remember what they have learned from prior inputs. The network can either generate one output, or one output per sequence or an output for every point in the sequence. The output is not only defined by the weights of the network, but also by the previous states. As shown in Figure 2.5 the output of the RNN is passed to the next time-step, the output of the final model is both based on the input and the previous outputs.

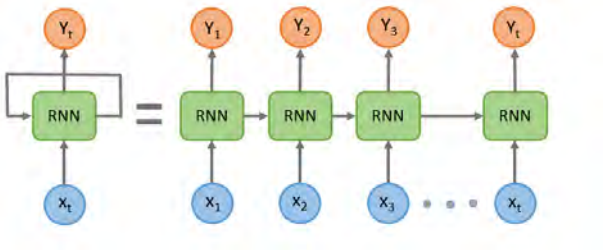


Figure 2.5: The final output y_i is depending on what the model learned from the previous inputs.

An RNN consists of several Neural network layers, one single layer is a simple layer consisting of one tanh layer. This is visualised in Figure 2.6, the output of the previous time frame and the new input are both used as input for the current time frame. The output o_t is passed to the output gate of the current time frame and the value h_t is passed to the next time frame. The input of RNN is a sequence of variable length $\mathbf{x} = (x_1, \ldots, x_T)$, where $x_t \in \mathbb{R}^n$ an input vector at time t. The output of the RNN is an ordered list of the output of the hidden states $h = (h_0, \ldots, h_t)$. The hidden states return the output vector $\mathbf{y} = \{y_1, \ldots y_T\}$. At each time step t, the hidden state h_t is updated by

$$h_t = f(h_{(t-1)}, x_t) \tag{2.8}$$

Where f is a nonlinear activation function.

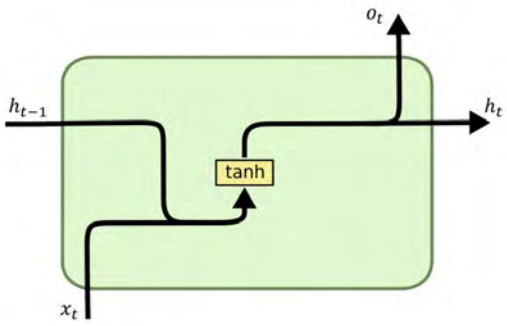


Figure 2.6: The structure of a single layer of a RNN consists of one tanh layer.[1]

When an RNN is used to solve an NLP-task, the input is not the full sentence but a single word. This makes it easier to handle sentences with varying lengths, this was not possible in standard Neural networks. In many NLP tasks the RNN is a many-to-one RNN, where the input is a full sentence and the output is one single value (like predicting the next word, or classify the sentiment) as in Figure 2.7. However, there are also problems where both the model input and the output is a sequence of words, such as when doing translation or predicting the next sentence.

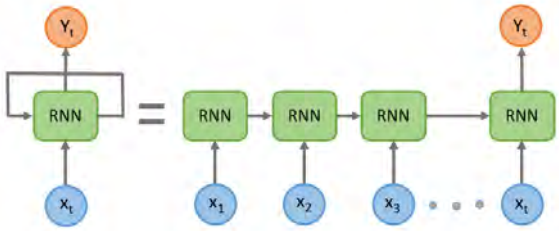


Figure 2.7: A many to one RNN network, where many inputs generate one output value.

2.4.6 LSTM

An RNN can be very useful when temporal sequences need to be learned. Predicting a value based on earlier data points is a task RNNs are quite good at. However, when important events occurred earlier in the sequence, RNNs having trouble remembering the important information. This important information needs to travel through the whole network, where in every step the information gets less value by the network. In theory, an RNN could learn long-term dependencies, however in practice the gradient would vanish or explode. *Long-Short-Term memory* (LSTM) was introduced to solve this problem, an LSTM is a special kind of an RNN, and is capable of learning long-term dependencies.

The RNN has a chain structure as shown in Figure 2.7, where every module in the chain consists of a neural network with a very simple structure, for instance a single tanh activation layer. The LSTM model also has a chain structure, but the structure of the individual cells is more complicated and consists of four layers instead of one layer, this is visualised in Figure 2.8.

The top line in the diagram is called the cell state, where important information can be retained, the other layers can throw away information from the cell state and add new information. The first layers determines by the input and the weights of this layers how much information should be thrown away from the cell state, this is done by a Sigmoid layer that will return a number between 0 and 1. Where 0 means all information will be thrown away and 1 means all information will be retained, this value is given by Equation 2.9. The value of f_t is multiplied by C_{t-1}.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2.9}$$

In the second step new information is added to the cell state, the information that will

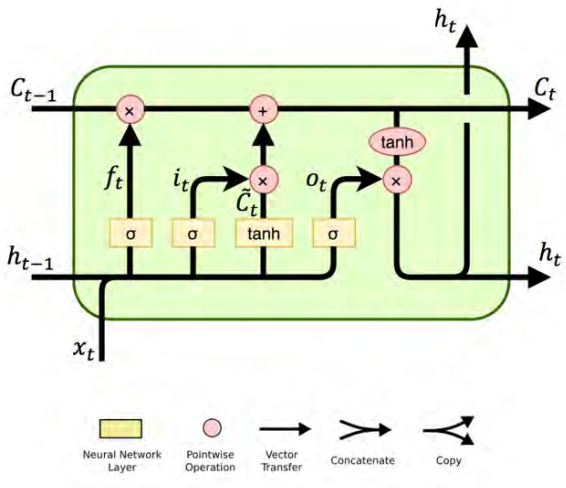


Figure 2.8: The structure of an LSTM cell consists of four different layers. [1]

be added to the cell state is a multiplication of i_t and $\tilde{C}_t$. Where i_t is defined by a sigmoid layer and $\tilde{C}_t$ by a tanh layer.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.10}$$
$$\tilde{C}_t = tanh(W_c[h_{t-1}, x_t] + b_c) \tag{2.11}$$

The new value of the cell state is now given by a combination of the first two steps. In the first step it is determined how much information will be lost, the remaining part is added by the new information as in Equation 2.12.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.12}$$

The output of the model is determined by both the cell state and the input. The value of the cell state is transformed by a tanh function and the input x_t is transformed by a sigmoid layer. These are multiplied:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.13}$$
$$h_t = o_t * tanh(C_t) \tag{2.14}$$

2.4.7 Bi-LSTM

When we read a text the context of the word is not only determined by the previous words, but also on the words occurring after the word. Regular RNNs are not able to read both backwards and forwards, *bidirectional LSTMs* (Bi-LSTM) were introduced to solve this problem. A bi-LSTM is combining two independent RNNs, one RNN will begin at the start and one RNN will begin at the end, therefore we can both read forward and backwards, and information from the past and from the future will be preserved.

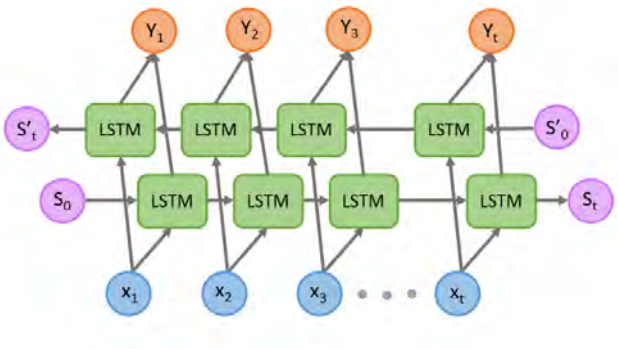


Figure 2.9: A Bi-LSTM network consists of two independent RNNs with LSTM layers.

2.5 Attention and transformer networks

Transformers are a type of Neural network that are becoming increasingly popular, since they are the building block and the reason for success in many transfer learning models, as will be explained in Section 2.6. These transformer networks have proven to be very effective on many NLP tasks and were first introduced by the paper "Attention is all you need" [44]. The transformer network is based on the attention mechanisms. Therefore, the attention mechanisms are introduced in Section 2.5.1 before the transformer network is explained in Section 2.5.2.

2.5.1 Attention mechanisms

When a human would read a sentence, one automatically scans trough the sentence and sees what parts of the sentence are important and remembers the important parts. An attention mechanism works similarly as a human mind, where it looks at the full input sequence and then determines what part of the sequence is important. The attention mechanisms (Bahdanau et al. [3]) was developed as a solution for the limitations of encoder-decoder architectures for Neural Machine Translation and was one of the most influential ideas in Deep Natural Language Processing.

Encoder-Decoder framework

In machine translation the goal is to translate a sentence x to a target sentence y. This is called a sequence to sequence *(Seq2Seq)* model. When a Recurrent Neural network is used for a sequence to sequence model, the sequences must have equal length. Therefore, Seq2Seq models normally consist of an Encoder-Decoder framework that was proposed by Cho et al. [9]. Here, the model consists of two components, the first component encodes the input sequence to a fixed-length context vector, that represents the input text. The second component decodes the context vector to the target sequence. In an *RNN Encoder-Decoder* both the encoder and the decoder are recurrent neural networks. The encoder encodes the vector $x = (x_1, \ldots, x_{T_x})$ to a context vector $\mathbf{c}$ by reading the sequence sequentially. Most commonly, an RNN is used such that the hidden state is

updated at each time step:

$$h_t = f(h_{(t-1)}, x_t) \tag{2.15}$$

and

$$\mathbf{c} = q(\{h_1, \ldots, h_{T_x}\}) \tag{2.16}$$

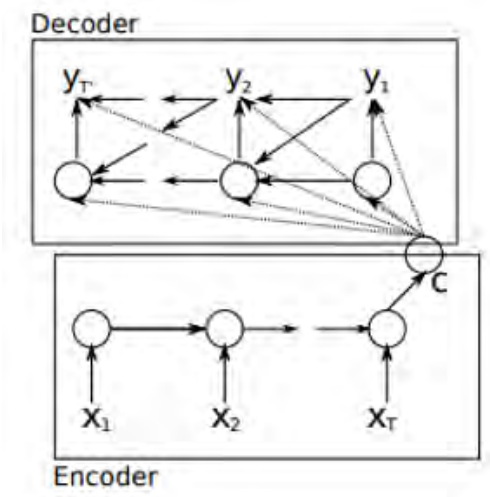
where $h_t \in \mathbb{R}^n$ is a hidden state at time t, and c the context vector that is a summary of the whole input sequence. f and q are nonlinear functions, these can be a simple element-wise logistic sigmoid or a more complex LSTM for instance. The decoder is again an RNN that decodes the context vector to the target sequence y, this is done by predicting the next word $y_{t'}$ given the hidden state h_t. The hidden state h_t also depends on y_{t-1} and the context vector $\mathbf{c}$:

$$h_t = f(h_{(t-1)}, y_{t-1}, \mathbf{c}) \tag{2.17}$$

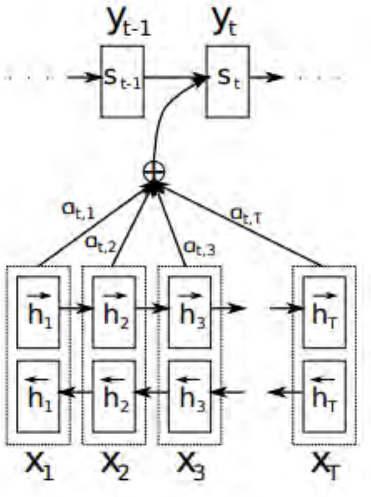
So, the conditional distribution of the next symbol is

$$P(y_t|y_{t-1}, y_{t-2}, \ldots, y_1, \mathbf{c}) = g(h_t, y_{t-1}, \mathbf{c}) \tag{2.18}$$

where f and g are activation functions. The graphical representation of the RNN Encoder-Decoder architecture is illustrated in Figure 2.10a. A limitation of the RNN encoder-decoder architecture is that the fixed-length context vector is not capable of remembering long term dependencies and when the input sequence gets larger the context vector is not able to store all the relevant information. The attention mechanism was introduced to resolve these issues.



(a) The architecture of a standard encoder-decoder model as proposed in [9]



(b) The architecture of an encoder-decoder model with attention as proposed in [3]

Figure 2.10: The transformation from RNN encoder-decoder models to encoder-decoder models with an attention mechanism.

Attention

In general encoder-decoder models the context vector is generally calculated by a RNN as shown in Figure 2.10a. Most information from the beginning of the sequence is already forgotten when arriving in this last cell. The attention mechanism does not build a single context vector from the last cell, but it connects the context vector to all different parts of the input sequence, this is visualised in Figure 2.10b. It reads the full input sequence at the same time and determines what parts are important by assigning attention weights. The decoder then takes both the context vector and the attention weights as input when decoding the sentence. In the model architecture with an attention mechanism the conditional distribution as in Eq. 2.18 is now defined as:

$$P(y_i|y_{i-1},\ldots,y_1,\mathbf{x}) = g(s_i, y_{i-1}, \mathbf{c}_i) \tag{2.19}$$

Where s_i is a hidden state of an RNN for time i and computed by:

$$s_i = f(s_{i-1}, y_{i-1}, \mathbf{c}_i)$$

The main difference with the RNN encoder decoder approach is for each target word y_i the probability is conditioned on a distinct context vector $\mathbf{c}_i$. This context vector $\mathbf{c}_i$ depends on a sequence of annotations $(h_1,\ldots,h_{T_x})$ to which the encoder maps the input sequence. Each annotation h_i consist of information of the whole input sequence, but specifically of the parts around the i-th word of the input sequence. The context vector $\mathbf{c}_i$ is computed by the weighted sum of the annotations h_i:

$$\mathbf{c}_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{2.20}$$

The weight α_{ij} for each annotation h_j is computed by:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}, \tag{2.21}$$

where

$$e_{ij} = a(s_{i-1}, h_j)$$

is an alignment model which scores how well the inputs around position j and the output at position i match. So in machine translation, if a word is important for the translation the attention weights increase. Therefore, the model can learn what parts of the input it should attend to when translating the sentence.

2.5.2 Transformer network

The transformer network was proposed in the paper "attention is all you need" [44], as the title suggests this network has something to do with the attention mechanism. The transformer is an encoder decoder architecture that uses self-attention and point-wise, fully connected layers for both the encoder and the decoder. The architecture of the Transformer is shown in Figure 2.5.2, with the encoder on the left and the decoder on the right, the encoder and decoder can be stacked upon each other multiple times. It is

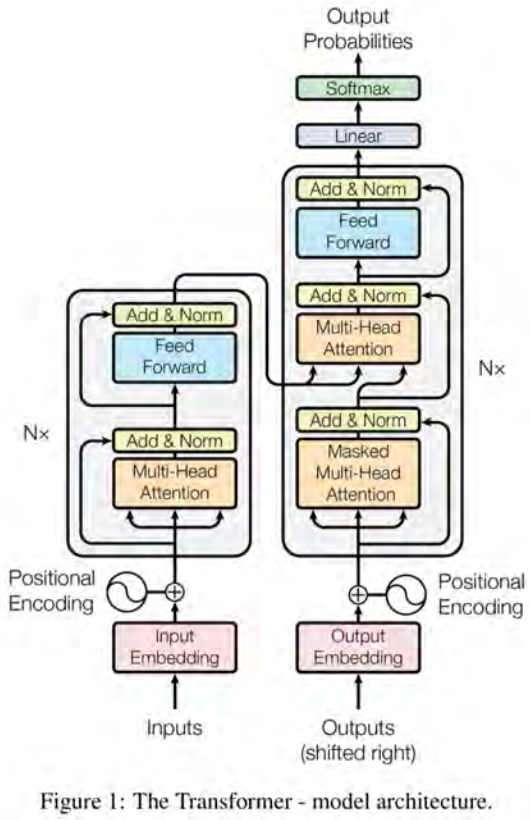


Figure 2.11: The model architecture of the transformer [44], the encoder is the left part and the decoder is the right part of the figure.

making use of attention mechanisms, however instead of first using an RNN on the input sequence, the attention is applied to the input sequence directly. Making the model more parallelizable and require less time to train.

The architecture mainly consists of multi-head attention and feed forward layers. The input and output are represented as embeddings before being fed into the model. Because the model does not use any RNN, positional encodings, that represents the relative position of the words in the sequence, are used to remember the order of the sequence. A closer look of the multi-head attention bricks is shown in Figure 2.12. The transformer is making use of attention functions where a query and a set of key-value pairs are mapped to an output. The query, keys, values and output are all vectors.

The attention mechanism is described by the following equation:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.22}$$

The matrices Q, K and V contain the vectors of the queries, keys and values. Eq. 2.22 describes the scaled dot product as visualised in Figure 2.12. First, the dot product of Q and K is scaled by a factor $\sqrt{d_k}$. A softmax function is applied to obtain the weights on the values V. Instead of performing a single attention function, Vaswani et al. [44]

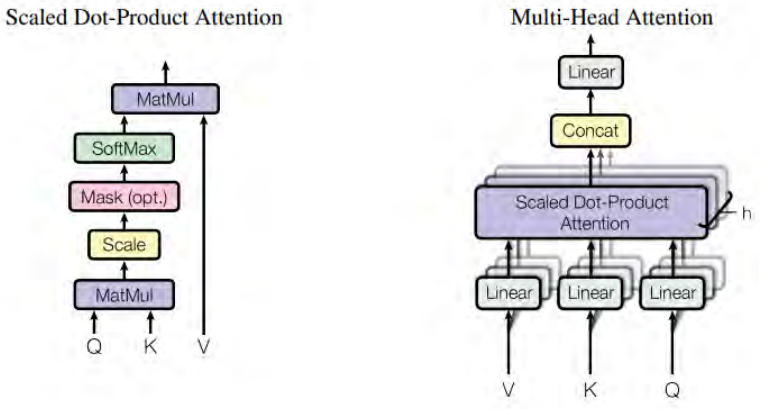


Figure 2.12: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel. [44]

found it beneficial to linearly project the queries, keys and values h times with different learned linear projections. These attention functions are performed in parallel and finally concatenated and once again projected. These result in the final values, as shown in Figure 2.12.

In a transformer network the sequence does not need to be processed in order, making it possible to parallelize the model during training. The model achieves state-of-the-art performance while improving the computation time. This led to the development of pre-trained systems like BERT as explained in Section 2.6.1.

2.6 Transfer learning

Deep learning methods try to learn features from large amounts of data, where features are automatically extracted from the data with virtually no need for manual feature engineering. This is a large advantage over traditional Machine Learning, where all of the features need to be designed manually. Deep Learning methods need large amounts of data to understand and learn patterns in the data, in many domains it is hard to find an annotated dataset that is large enough and of sufficient quality. Transfer learning is a concept in Machine Learning that can solve the problem of insufficient training data. In transfer learning, a model is trained on a large-scale dataset, then this pre-trained model is used to learn from new samples from a different downstream task. Patterns that are learned from the large-scale dataset are used as a starting point for the training. Low-level features are already learned from previous tasks, and tend to be very similar over the different tasks.

In the case of Natural Language Processing tasks, the model is trained on a large corpus of text files. The model is trained on an unsupervised learning task. There are several unsupervised learning tasks:

1. Language model - Predict next word
   (I thought I would arrive on time, but ended up 5 minutes **MASK**.)

2. Masked language model - Predict missing words
   (I thought I would **MASK** on time, but ended up 5 **MASK** late.)
3. Predict next sentence, where the model predicts the likelihood a sentence is the next sentence.

By training on one of the unsupervised learning tasks, the model does not need any labelled data and it learns how to read and understand the language. The model can use the knowledge about the language in NLP tasks like question answering, sentiment analysis or relation extraction. The pre-trained model is able to read and understand the language, however, it does not yet know how to detect sentiment or other new facts. It should first receive some examples of different labelled sentences that include sentiment. After receiving the task-specific data it is able to learn how to detect it.

The first major contribution due to transfer learning in Natural Language Processing was the use of pre-trained word embeddings in 2010 [43]. Different objectives are used to pre-train the word embeddings. There are two approaches to use pre-training techniques in Natural Language Processing tasks, feature-based and fine-tuning based approaches. An example of a feature based approach is ELMo [30], where the word representations models both the syntactic and semantic meaning of a word, besides the characteristics of the word use, it also models how these vary across linguistic contexts. In [30] it is shown that by adding the pre-trained model to existing models the state-of-the-art performance can be increased significantly for different NLP problems. In feature-based approaches the pre-trained representations are added as additional features to the existing models. In fine-tuning approaches, there are minimal task-specific parameters, and the pre-trained model is fine-tuned by learning from data from the downstream task. During the pre-training both models use the same objective function, where unidirectional language models are used to learn language representations. The advantage of this approach is that few parameters need to be trained from scratch.

2.6.1 BERT

Recently J. Devlin et al. from Google introduced BERT [12] (Bidirectional Encoder Representations from Transformers). BERT is a Natural Language Processing pre-training approach, it can learn deep bidirectional representations. In [12] they argue that current techniques like feature-based and fine-tuning approaches restrict the power of pre-trained representations, since the model architectures are not able to incorporate context from both directions. By introducing BERT they claim to improve the fine-tuning based approach by using a masked language model instead of a unidirectional language model. The masked language model is based on the Cloze tasks, described by Taylor in 1953 [42], in the Cloze task several words are removed from a sentence and a student is asked to fill in the missing content. During pre-training, randomly several tokens of the input sentence are masked, the objective of the model is to predict the original vocabulary id based on its context. By using this techniques the model does not require any human labelling. The masked language model can learn deep bidirectional representations of words by using a transformer encoder.

The BERT framework consists of two steps, pre-training the model and fine-tuning the model. In the pre-training the model is trained on unlabelled data to learn the low-level

features of the model. In the fine-tuning the model that is created during the pre-training is used as starting point, and the parameters are initialised with the weights resulting from the pre-training. Then all of the parameters are fine-tuned by using the labelled data from the specific task that needs to be solved. For every downstream task the fine-tuned model is different, but there are all initialised with the same weights.

BERT uses contextual representations, meaning there is not a single embedding for every word in the vocabulary, but the representation depends on the context of the word. The architecture of BERT is a multi-layer bidirectional Transformer encoder and uses self-attention, where the inputs interact with each other and find the parts that need more attention. BERT is outperforming all other pre-training approaches and achieves state-of-the-art performance on many NLP tasks.

2.6.2 BioBERT

Since biomedical text data contains different words than regular documents, when pre-trained models are directly applied to biomedical text mining, the results are often unsatisfactory. In [21] BioBERT is proposed and it is investigated how BERT can be adapted for biomedical corpora. BioBERT is a domain specific language model, the model is first initialised by the weights resulting of BERT and then BioBERT is pre-trained on a large corpora of biomedical documents.

Pre-training the model on biomedical corpus improves the performance in comparison to the original pre-trained model. The performance is tested for different biomedical text mining task and outperforms all of the previous state-of-the-art models.

Chapter 3

Data

In this research, we would like to generate a biomedical relation extraction method that can detect different kinds of biomedical relations in text. To generalise a model over the different relations, data from different sources is combined. Both annotated data and data created by distant supervision is used in this research. For some relations there are annotated datasets available that include a sentence that describes the relation. Creating these datasets is very time consuming and expensive, since in most cases this can only be done by a professional due of the complexity of the text. There is many knowledge available on biomedical relations between different type of entities. These relations can be for instance that a drug can treat a disease or that a disease is located in a part of the body. These relations are saved in structured formatting like databases. Sentences that describe these relations are much harder to find, since in most cases they are written down in biomedical documents. To look for these hidden sentences, the structured data is used to find the sentences that describe the relation in the database, this task is performed by distant supervision. With the dataset created by distant supervision new facts can be extracted from unseen documents.

3.1 Annotated data

There are some datasets available online that contain annotated sentences about a single relation. These sentences are combined with the data collected by distant supervision to create a large training set. In this thesis one annotated dataset is used:

The annotated dataset contains relations from the Drug-Drug interaction corpus [36], the corpus consists of unstructured textual information about drugs and their interactions. The data was manually annotated with the assistance of an expert pharmacist. In total, it contains 5,806 sentences and 30,779 candidate drug pairs. The data consists of both positive and negative samples. The positive samples are divided into a set of specific categories, however, these specific categories are not considered in this research.

3.2 Knowledge Base

To generate new facts by distant supervision, we should first collect a set of seed facts. These facts are relations that are already known. The first set of known relations are

the relations extracted from the annotated datasets. The knowledge base is a collection of different types of relations in the biomedical field. There are many official databases available, but in most cases they focus only on one part of the domain (for instance, treatments for diseases). Combining the different databases is a difficult task itself, because all the namings need to be aligned. A research of e-life [17] came up with a solution for this, they combined all data sources in one knowledge graph named *Hetionet*. Millions of biomedical studies were integrated in one network, 29 public resources were used to connect compounds, diseases, genes, anatomies and more. The main goal of the research was to find new relations by analysing the graph structure and predict whether there would be a relation between two target entities based on the existing relations. The goal of the research is similar to the goal of this research, however the method used is completely different. Their approach is based on network analysis, whilst in this research the methods used are based on NLP.

Entity type	**Description**	**Source database**
Compound	Approved small molecule compounds with documented chemical structures.	DrugBank [47].
Disease	Diseases	Disease ontology [35]
Side Effect	Adverse drug reaction	UMLS [5] & SIDER [8]
Symptom	Clinical abnormalities that can indicate a medical condition	MeSH [24]

Table 3.1: Description of the different entities used in this research

Hetionet consists of many different types of relations between different entities. Relations between genes and other entities are often very complex. the choice is made to exclude relations concerning genes in this research, since the process of detecting relations between genes is very different from detecting relations between diseases and drugs. The relations that were included in the research are described in Table 3.2, they refer to the entities described in Table 3.1. A visual representation of the existing knowledge graph is shown in Figure 3.1.

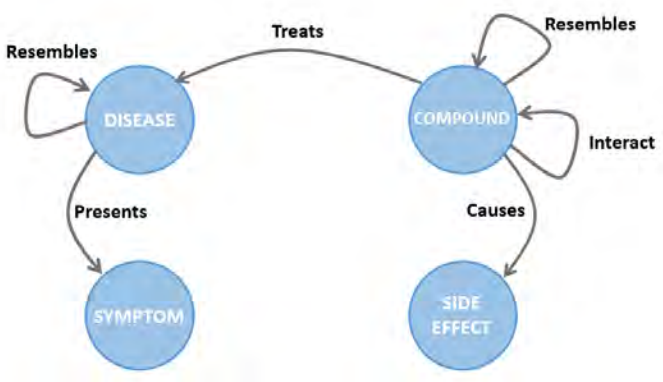


Figure 3.1: A visualisation of the knowledge graph as extracted from Hetionet.

Relation	Entity type 1	Entity type 2	Description
Resembles	Compound	Compound	Two drugs that resemble each other
Interact	Compound	Compound	Two drugs interact together
Causes	Compound	Side Effect	A drug causes a side effect
Treats	Compound	Disease	A drug treats a disease
Presents	Disease	Symptom	A disease has a symptom
Resembles	Disease	Disease	Two diseases that resemble each other

Table 3.2: Description of the different relations used in this research

3.3 Data by distant supervision

When data is extracted by distant supervision we should find sentences that describe the relation available in the knowledge base. Distant Supervision is based on the assumption that if two entities participate in a relation, all sentences that mention both of these entities express that relation.

The knowledge base used in this research is extracted from Hetionet, the new data by distant supervision is extracted from this knowledge base. We searched for new data in the PubMed archive [29], the PubMed archive consists of more than 30 million citations for biomedical literature from MEDLINE, life science journals, and online books. To extract the useful information from PubMed we query for entity pairs in our knowledge base and check in the query results whether there is a pair mentioned of two entities somewhere in the abstracts that occur in the knowledge base as well. Then this sentence is saved with the two targeted entities and the relation described in the knowledge base. The final collection of sentences is the dataset used in this research. The step by step implementation of the distant supervision approach is explained below.

1. Loop over the relations between entities in the knowledge base
2. For every relation in the knowledge base, query for the entity pair on the PubMed archive
3. Collect all the textual data where both entities are mentioned
4. Detect all the entities in the text by a Named Entity Recognition model by using SciSpacy [28] with the model trained on the BC5CDR corpus [16]. This model is able to detect diseases and chemicals in text data.
5. The entities are linked to the Unified Medical Language System [6] to link them to the entities in our knowledge base, this is done by the UMLS entity linker that is implemented in SciSpacy.

6. We check whether the entities exist in our knowledge base, if so, these entities are retained.

7. For every sentence with at least two entities in the sentence that exists in our knowledge base, it is checked whether one or more of the combinations of different entities exists in the knowledge base.

8. All combination of entities that occur in the knowledge base are saved into the dataset with the relation given in the knowledge base as label and the sentence and the targeted entities as data.

3.4 Negative Samples

An issue with the dataset that is obtained by distant supervision is that the dataset does not contain any sentences where there is no relation described. When the model would be applied to detect new relations, the main goal is to detect whether there is a relation in the sentence. Creating these negative samples is very complex, since one would need to manually check whether there is no relation between the targeted entities in the sentence. Manually labelling sentences is not feasible, because the sentences are very technical and long. The annotated dataset consists of a set of negative samples, but these negative samples ensure that there is no drug-drug interaction in the dataset, it can not guarantee that there is not another relation described. We do know that this dataset only contains drug-drug pairs, this would mean that there is no relation possible that requires a disease as one of the entities. Therefore, the negative samples can only be confused with the *resembles* relation between two compounds.

To be able to predict whether or not there is a relation between the targeted entities in the sentences, the dataset is slightly adjusted. The two adjustments made are explained and substantiated below:

***Resembles* relations:** In the distant supervision method we extracted relations of drugs and diseases that are similar to each other (*resembles*). The sentences that were found by distant supervision for these relations were in many cases very noisy, since these resembling diseases or drugs are often occur in the same sentence without the sentence mentioning the two are similar. Therefore the decision is made to move the resemble relations to the *negative* class. If two entities are similar they can not occur in any other relation covered in this research. By moving these relations to the *negative* class, the number of classes is reduced to 5 classes.

Change entity types: In Table 3.1 the different types of entities were described. There are four different types of entities: Compounds, diseases, side effects and symptoms. In the Named Entity Recognition model used in this research [16], side effects, diseases and symptoms are all recognised as diseases and all compounds are recognised as chemicals. To reduce the number of entity types, the entity types from the Named Entity Recognition model are used, the new relations and their entity types are shown in Table 3.3.

By changing the dataset we can now distinguish between a positive relation or no relation. Important to note is that the $Negative$ relation does not mean there is no relation

Relation	Entity type 1	Entity type 2
Interact	Chemical	Chemical
Causes	Chemical	Disease (side effect)
Treats	Chemical	Disease
Presents	Disease	Disease (symptom)
Negative	Disease/Chemical	Disease/Chemical

Table 3.3: Description of the different relations used in the final dataset

described between the targeted entities in the sentence, but if the *Negative* relation is predicted we can exclude the other 4 relations. After making these two adjustments the final dataset is created that consists of diseases and chemicals, the different relations can be visualised in a knowledge graph as shown in Figure 3.2. The dataset that is created is analysed in the next section.

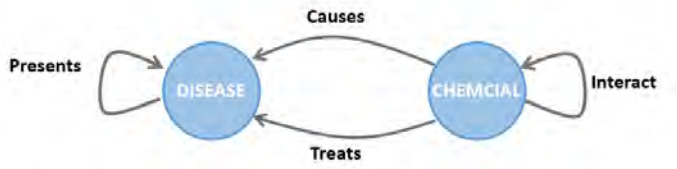


Figure 3.2: The possible relations visualised in a knowledge graph

3.5 Data Analysis

The dataset that is created by combining pre-annotated data and data by distant supervision, consists of 15,821 sentences. The number of sentences per relation is not equal, so the dataset is imbalanced. In Figure 3.3 the distribution of the dataset is shown.

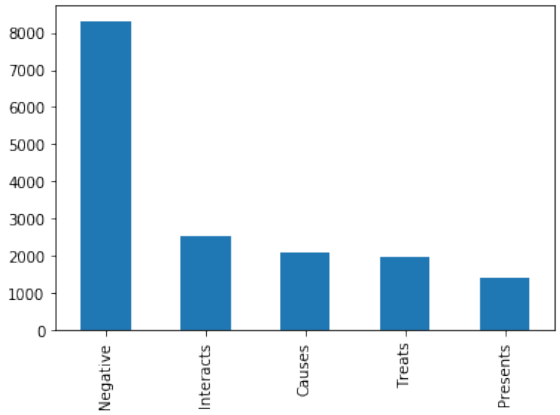
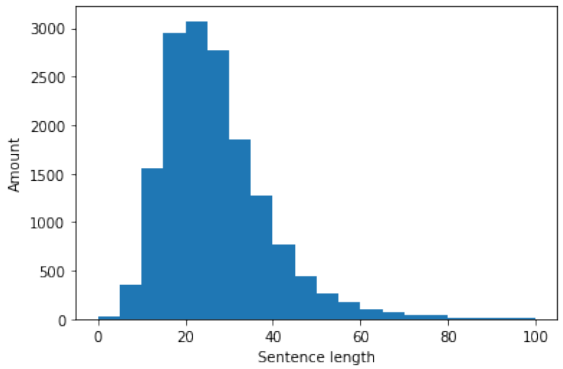


Figure 3.3: The number of sentences per relation type.

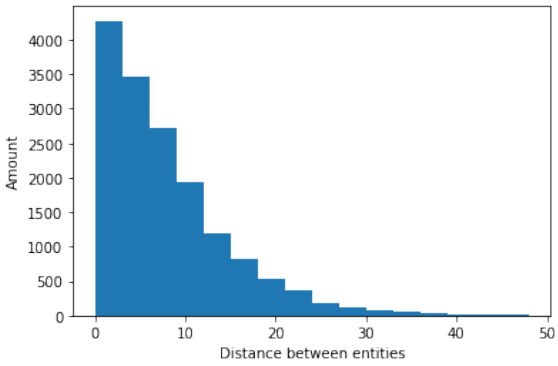
The complexity of the sentences strongly depends on the sentence length, therefore the distribution of the sentence length is shown in Figure 3.4a. From this figure it can be

concluded that there are relatively few short sentences in the dataset. As most sentences have at least 20 words.

When a relation between the targeted entities is extracted from the sentence, the number of words between the two entities can also say something about the complexity of the sentence, especially how difficult it will be to extract the relation. In Figure 3.4b it is shown how the distance between the targeted entities is distributed, most sentences have less than 5 words between the two entities, nevertheless there are also sentences with 30 words between the two entities.



(a) The distribution of the sentence length of the sentences in the dataset.



(b) The distribution of the distance between the targeted entities.

Figure 3.4: Charachteristics of the sentences

To see whether there is a correlation between the type of relation and the sentence length and the distance between the targeted entities we analyse the average values per relation, shown in Table 3.4. Interestingly, the distance between the entities does differ a lot per relation type, for instance for the *Causes* relation the average distance is 11.72, while for *Negative* class this is 5.65. This can tell us something about the complexity of the sentences.

Relation	Distance between entities	Sentence length
Negative	5.65	26.69
Causes	11.72	29.10
Treats	11.23	28.25
Presents	8.74	26.58
Interact	7.95	23.00

Table 3.4: The average distance between entities and sentence length per relation.

In Table 3.5 the 10 most important words for every relation are shown, the importance of the words is measured by the TF-IDF score (explained in Section 2.1.2). The table is interesting in several ways. First, there are some relations that are very similar like *Treats* and *Causes*, the top 10 words are nearly equal. It is surprising that the word *causing* is not included in the top 10 words for the relation *Causes*.

Negative	*Causes*	*Treats*	*Presents*	*Interact*
patients	patients	patients	patients	increase
mg	treatment	treatment	sleep	administration
drugs	study	study	symptoms	mg
effects	mg	efficacy	study	plasma
study	effects	used	disease	patients
effect	efficacy	safety	pd	concomitant
administration	associated	effects	associated	administered
drug	effect	mg	movement	drugs
treatment	therapy	therapy	disorder	increased
plasma	clinical	evaluate	common	effects

Table 3.5: The top 10 words per relation (based on TF-IDF Score)

Chapter 4

Methodology

In this Section a description of the research design is given, we will start with the experimental set-up of the deep learning methods and end with the experimental set-up of the transfer learning method where BERT is used.

4.1 Deep Learning

The dataset that is described in Section 3 contains sufficient samples to apply deep learning to this task. In recent literature on drug-drug interactions ([50], [51], [34] and [49]), many different features and architectures seem to be successful in the task of biomedical relation extraction.

4.1.1 Feature Selection

The advantage of deep learning techniques is that there is no need for complicated feature engineering. When deep learning models are applied to NLP-tasks the first layer is always an embedding layer. This layer will give the word embeddings of all the words in the sentence. We used pre-trained word embeddings [13] that are trained on the PubMed [29] and PMC texts by using the word2vec tool [15]. The sentence embedding is the most simple representation of the sentence. To test whether adding more features will increase the performance, the following different features are tested:

The **shortest dependency path** (*sdp-path*) contains the dependency syntax information of the sentence, the shortest dependency path is the shortest path between the two targeted entities as explained in Figure 4.1. Recent studies ([50], [25], [49] and [27]) have shown that the dependency path can boost the performance of the relation extraction. The shortest dependency path is also transformed by an embedding layer. In order to extract the shortest dependency path some pre-processing steps are required. The sentence is first tokenised to extract the tokens in the sentence, then a dependency parsing method can extract the shortest dependency path between two tokens in the sentence. The dependency parser from SciSpacy [28] is used in this research.

The **position Embedding** describes per word in the sentence the relative distance between that word and targeted entities [48].

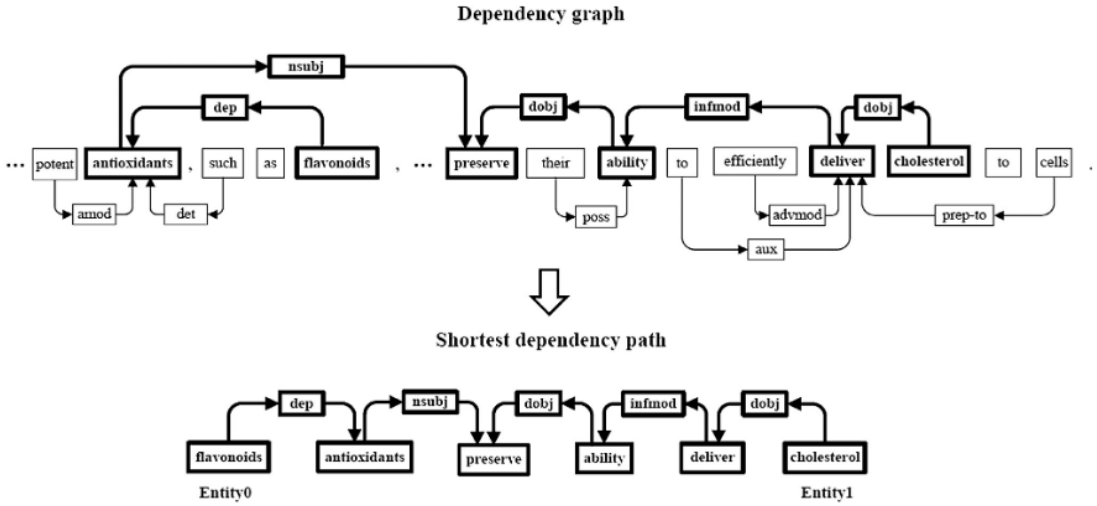


Figure 4.1: An example of the shortest dependency path between two entities. [49]

Let $\{w_1, w_2, \ldots, w_n\}$ the raw sequence of words and $\{d_1, d_2, \ldots, d_k\}$ the raw sequence of words in the shortest dependency path for every sentence S. For every word w_i in $\{w_1, w_2, \ldots, w_n\}$ the word embedding is given by w_i^{word} and is obtained from the full embedding matrix W_{word}. The relative distances $w_i^{dis_1}$ and $w_i^{dis_2}$ are obtained by calculating the relative distance between the location of the ith word and the location of the entities. The word w_i is represented by $z_{wi} = \{(w_i^{word})^T, (w_i^{dis_1})^T, (w_i^{dis_2})^T\}$. A word d_i in the dependency path $\{d_1, d_2, \ldots, d_k\}$ is represented by $z_{di} = \{(d_i^{word})^T, (d_i^{dis_1})^T, (d_i^{dis_2})^T\}$.

The feature set that will be tested consists of 4 different sets of features:

1. Word Embeddings
   $(z_{wi} = \{(w_i^{word})^T\})$
2. Word Embeddings + Position Embeddings
   $(z_{wi} = \{(w_i^{word})^T, (w_i^{dis_1})^T, (w_i^{dis_2})^T\})$
3. Word Embeddings + Shortest Dependence Path
   $(z_{wi} = \{(w_i^{word})^T\}, z_{di} = \{(d_i^{word})^T\})$
4. Word Embeddings + Shortest Dependence Path + Position Embedding
   $(z_{wi} = \{(w_i^{word})^T, (w_i^{dis_1})^T, (w_i^{dis_2})^T\}, z_{di} = \{(d_i^{word})^T, (d_i^{dis_1})^T, (d_i^{dis_2})^T\})$

4.1.2 Model architecture

There are different types of layers that have shown to be very effective in the task of relation extraction. As stated before the first layer is always an embedding layer to transform the words to the word embedding vectors. The subsequent layers are highly depending on the type of input. Zhang et al. [49] have shown that combining an CNN and a Bi-LSTM network is performing better than applying CNNs or Bi-LSTM seperately to the input sequence. While CNNs are better in learning local lexical and syntactic features, RNNs (Bi-lSTMs) are more powerfull in capturing dependency features over longtime spans. Therefore, Bi-LSTM are suitable for long and complicated sentences and CNNs are better

in capturing information from shorter sentences.

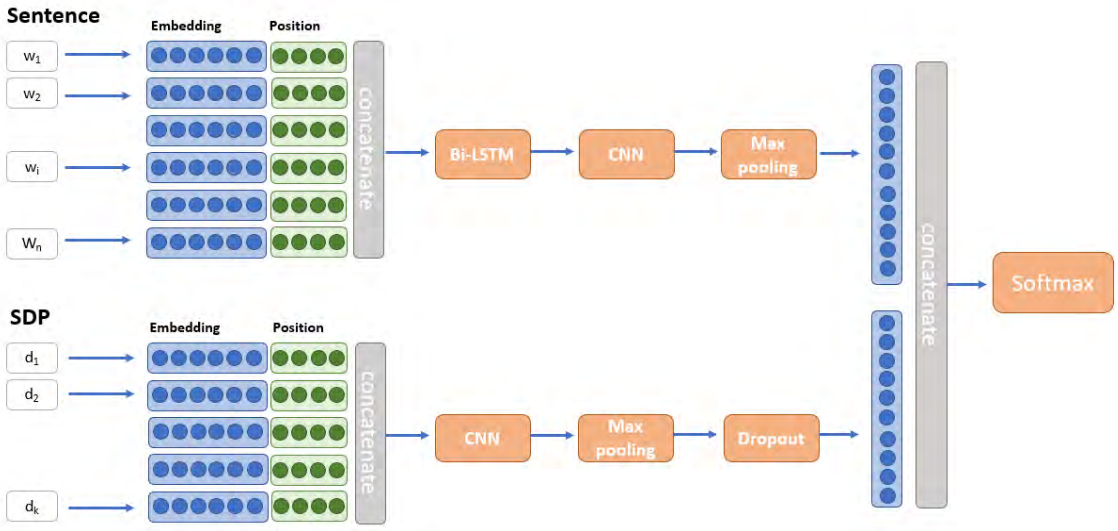


Figure 4.2: The model architecture of the deep learning network.

The full sentence and the shortest dependency path are two separate inputs in the network. Both the shortest dependency path and the sentence are first transformed by an embedding layer. If the position embedding feature is included in the feature set, the relative distance values are also transformed by an embedding layer. The position embedding is concatenated with the sentence embedding or the sdp-embedding.

The input sentence is usually a long and complicated sentence, therefore a Bi-LSTM layer is more suitable, since it can capture information over longtime spans. The Bi-LSTM layer consists of a drop-out rate to prevent the model from over-fitting. After the Bi-LSTM layer there is a convolutional layer that we expect to learn local features. By using a max-pooling the maximum value is selected to select the most important feature.

The shortest dependency path is much shorter and in most cases less than 5 words. Therefore, the shortest dependency path is first going through an embedding layer and then through a convolutional layer. After the convolutional layer a max-pooling layer is applied to select the most important feature. Finally, a Dropout layer is added to prevent the model from over-fitting.

After both the shortest dependency path and the full sentence travelled through all the layers of the network they are concatenated before the last fully connected layer. In the last layer a fully connected layer a Softmax activation function is used to return the final prediction.

The model architecture as outlined above describes the architecture when all features are used. The input sequence of all words and the input sequence of the words in the shortest dependency path are both passed into different branches of the network. When we would consider a different feature set, we can easily remove the branch of the network

that is not not used. In case there is no position embedding, the embedding vectors only consist of the word embeddings. When the shortest dependency path is not included in the feature set, the whole input branch of the shortest dependency path is deleted from the network.

4.1.3 Hyper-parameters

The hyper-parameters of this model are summarised in Table 4.1, these values are based on literature and initial testing.

Parameter	Value
Maximum sentence length	150
Maximum SDP length	15
Word embedding length	200
SDP embedding length	200
Position embedding dimensionality	15
Position embedding SDP dimensionality	5
LSTM hidden units	200
CNN hidden units	64
Drop-out rate	0.5

Table 4.1: The hyper-parameters of the model.

4.2 BERT

In recent years BERT is applied to several problems in the field of relation extraction. Bio-BERT is the BERT model that is pre-trained on a large corpus of biomedical data, therefore it performs very well on biomedical NLP tasks.

4.2.1 BERT base model

In this thesis we will apply BERT and test the performance against state-of-the-art deep learning methods. The BERT model will be initialised by using either BioBERT or BERT, then a final fully connected Softmax layer is added to classify the classes as shown in Figure 4.3. To test the difference between the original BERT base model and BioBERT, the performance of both models is tested. To use BERT almost no pre-processing is needed, the input of the model is a sentence and a label.

4.2.2 Architecture

The architecture of BERT is built on top of a transformer, the base model (used in this research) consists of 12 layers and 110 million parameters. Different from most other transfer learning models in NLP, BERT does not generate one word embedding for each word in the vocabulary, but the representation is depending on the left and right context of the word. To do this, BERT makes use of a transformer (Section 2.5), that uses an attention mechanism that can learn contextual relation between words. The transformer

reads the whole input sentence at once instead of sequentially, this is making the transformer bidirectional. In this research, the BERT base model is used as the base model, in the fine-tuning phase a Softmax layer is added to predict the probability of label c:

$$p(c|h) = softmax(Wh)$$

Where W are the task-specific weights that are learned by fine-tuning the model and h the values of the final hidden state.

At first the targeted entities are masked and the sentence is passed to the transformer. The model can now train on the set of training samples in the dataset and the model will be fine-tuned to solve this specific task. The architecture of the fine-tuning model is given in Figure 4.3, where the hidden layers are transformers.

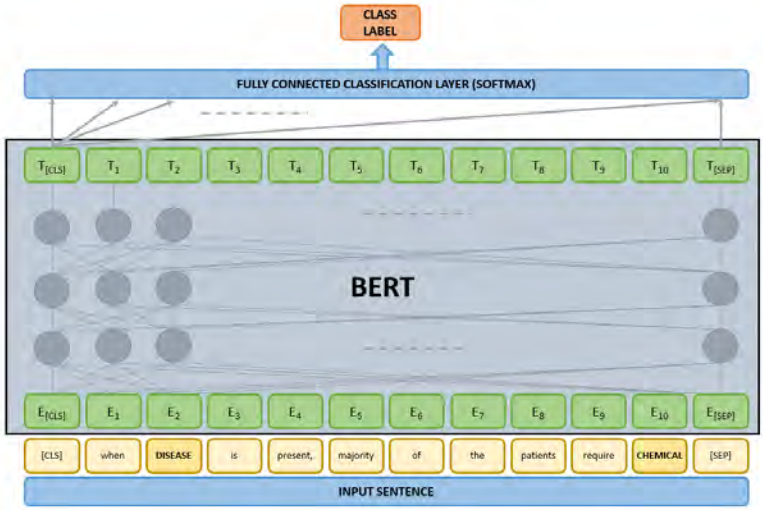


Figure 4.3: The BERT architecture applied to the Biomedical relation extraction task

4.3 Implementation

The implementation of the model is done in Python. The deep learning models are implemented by the package Keras [10] and the for the implementation of BERT the Tensorflow implementation from the BERT github [12] is used. The linguistic processing steps are performed by SciSpacy [28]. This is a NLP pipeline that consists of a biomedical tokeniser, sentence splitter, entity recognition model and a dependency parser.

4.4 Evaluation metrics

We evaluate the performance of the classification models by three performance measures, *precision*, *recall* and *F1-score*. These performance measures are used since the classes in the dataset are imbalanced. For every class all scores are calculated separately to measure the performance per class. The evaluation metrics are based on the values of the confusion matrix shown in Figure 4.4.

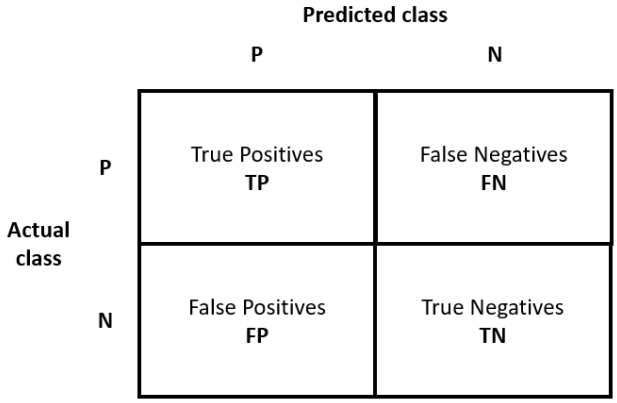


Figure 4.4: Confusion matrix

From this confusion matrix much information can be extracted. The two most important evaluation metrics that can be extracted from the confusion metric are precision and recall. Precision gives information about how many times the prediction was correct relative to the total number of times this class was predicted. Precision is defined by the following formula:

$$Precision = \frac{TP}{TP + FP}$$

The second important evaluation metric that can be extracted from the confusion matrix is the recall score. The recall measures how many of the actual class samples were missed by the model predictions. Recall is defined by:

$$Recall = \frac{TP}{TP + FN}$$

The recall and precison score combined is named the F1-score. The F1-score is the harmonic mean of the precision and recall and given by:

$$F1 - score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Since both precision and recall are important in this problem, the f1-score is evaluated for every model. To get a higher understanding of the performance of the models, we will also look into the confusion matrices. To show the overall performance over the classes instead of the scores calculated for every class individually, we also analyse the macro-average scores and the weighted average scores.

4.5 Overfitting

When dealing with text data most models are very sensitive to overfitting. When the model is overfitting, the model achieves high accuracy on the training dataset, but low accuracy on the test dataset (unseen data). Overfitting can be prevented by looking at the validation metrics. If the training loss is decreasing while the validation loss is increasing it is likely that the model is overfitting on the training data, this is visualised in Figure 4.5. It is important to stop the training before the model starts with overfitting. During the training, the training loss and the validation loss are monitored, after every epoch the model is saved if the validation loss is lower than the previous value of the validation loss.

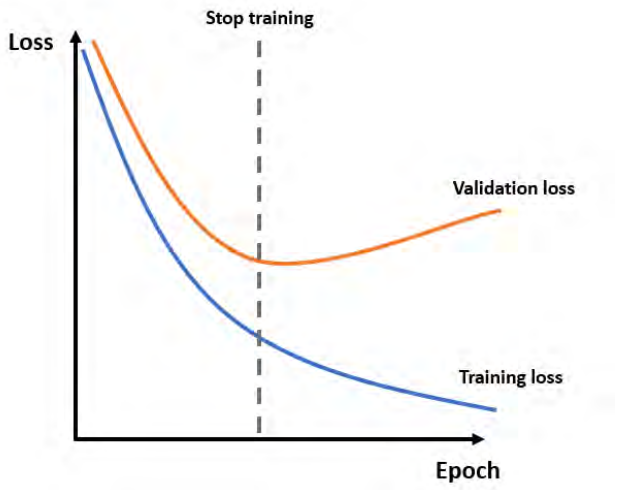


Figure 4.5: The model should stop the training when the validation loss starts increasing.

To reduce overfitting even more, a dropout layer is added to the model. The dropout layer randomly ignores a set of nodes and its incoming and outgoing edges. The training process will become noisy such that the model is less sensitive to overfitting when seeing the same sample multiple times.

4.6 Experiments

To compare the different models, we will perform a set of experiments. The dataset explained in Section 3 is used. The dataset is split into a training and validation set by randomly sampling 70% of the data as training data, the remaining 30% will represent the validation set. All models are trained with an identical training dataset.

When a sentence is passed to the model, it is important that the targeted entities that occur in the relation are masked. This has two main reasons: The first reason is that if the entity names are in the sentence, the model can learn to remember the entity names. Secondly, if the entities are masked by a recognisable term, the model can detect the location of the entity. Therefore, if two similar sentences are passed with with a different entity pair, the prediction does not have to be equal. Two different experiments will be performed for all models, in the first experiment the targeted entities will be replaced

by a general term ($ENT1$ for the first entity in the relation and $ENT2$ for the second one). In the second experiment, the targeted entities are replaced by its entity type ($DISEASE/CHEMICAL$). In both experiments, all other entities in the sentence are masked by $ENT0$. The performance of the models in the second experiment are expected to be better than the performance of the models in the first experiment. Since more information is available and some relations are not feasible, since certain entity types are required. The performance of the models is evaluated by using the performance measures explained in Section 4.4.

Chapter 5

Results

In this section we will outline the performance of the different models and the results of the experiments explained in Section 4.6 are evaluated. In the two experiments we will investigate how different models are able to detect relations in sentences. In the first experiment the model has no information on the specific entity type of the targeted entities, so the model does not know whether the entity is a disease or a drug. In the second experiment we test if and how the performance increases when the model does know what the entity types are.

5.1 Experiment 1

In the first experiment the targeted entities are replaced by general masks ($ENT1$ and $ENT2$), since the model does not know whether the entity is a drug or a disease they can only base the prediction on the words in the sentence. In this experiment we have tested four different Deep learning architectures and two different models of BERT as explained in Section 4. In Table 5.1 the performance of the different models is summarised. The performance over the different classes is varying a lot (between 0.4 and 0.9), this mainly has to do with the complexity of the class and the amount of samples in the dataset (the *Negative* class consists of much more labels than the other classes).

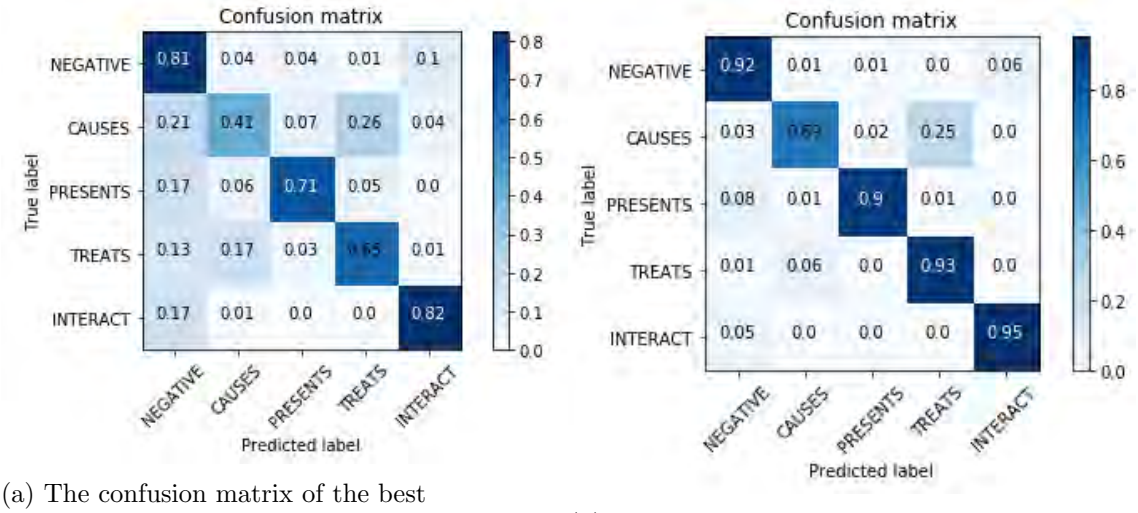
From the results shown, it is clear that the performance of the BERT models is not at all comparable with the performance of the Deep learning models. The best performing Deep learning model is the model with the shortest dependency path (Macro Average F1-score of 0.67), however, the difference in performance between the deep learning models is negligible. One surprising result is that the performance is decreasing 2% when adding the position embeddings to the model, suggesting the model does not benefit from knowing where the entities are located.

When looking at the performance of the two BERT models we immediately see that the performance is much higher than the best deep learning models, since the macro average is increasing with 13%. As expected, the BioBERT model is outperforming the BERT model, since the BioBERT model has a macro average F1-score of 0.87 and the BERT model has an macro average F1-score of 0.83. This 4% performance increase has to do with the model having more knowledge of the biomedical domain because it is pre-trained on the biomedical domain.

	DL Models				BERT Models	
Class	Base	Positions	SDP	Complete	BERT	BioBERT
Negative	0.82	0.8	0.82	0.82	0.92	0.94
Causes	0.42	0.42	0.46	0.45	0.7	0.77
Presents	0.66	0.64	0.68	0.66	0.87	0.91
Treats	0.62	0.61	0.65	0.64	0.79	0.84
Interact	0.72	0.7	0.75	0.72	0.85	0.89
Macro Average	0.65	0.63	**0.67**	0.66	0.83	**0.87**
Weighted Average	0.71	0.7	**0.73**	0.72	0.86	**0.9**

Table 5.1: Model evaluation for the first experiment (scores are all F1-measures).

The confusion matrices of the best Deep learning model (shortest dependency path) and the overall best model (BioBERT) are shown in Figure 5.1. The BioBERT model is better in distinguishing whether a sample is negative or not. In the DL model the model is predicting *Negative* very often. In both models the model is having trouble with the difference between the *Causes* relation and the *Treats* relation (25% of the times *Treats* was predicted the final label was *Causes*). This probably has to do with the similarity of the text in the sentences of both classes that was shown in Table 3.5.



(a) The confusion matrix of the best DL model (shortest dependency path)

(b) The confusion matrix of the BioBERT model

Figure 5.1: Comparison of the best DL model and the BioBERT model when looking at the confusion matrix per model

5.2 Experiment 2

In the second experiment the entities in the sentence are replaced by the entity type, for example, when a the first entity is a disease, the entity will be replaced by *DISEASE*. By doing this, there are fewer relations that can belong to this sentence. When both entities have entity type disease, the relation *Treats* is not possible for this example. In the second experiment we will compare the same models as in the first experiment.

Initially we would have expected the performance would increase significantly after using the entity types as masks. However, when analysing the results of the second experiment (Table 5.2), the performance of the Deep learning models was lower than the performance of the Deep learning models in the first experiment.

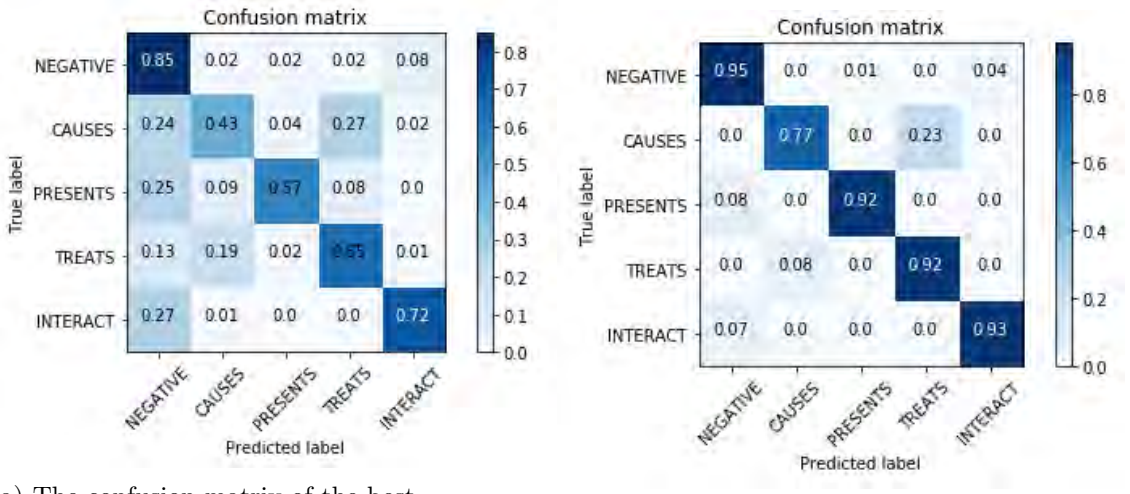
In contrast to the Deep learning models, the BERT models do have an increase in performance in the second experiment. The macro average F1-score of the BioBERT model is 3% higher than in the first experiment.

	DL Models				BERT Models	
Class	Base	Positions	SDP	Complete	BERT	BioBERT
Negative	0.76	0.77	0.82	0.82	0.95	0.96
Causes	0.31	0.41	0.41	0.47	0.76	0.83
Presents	0.64	0.64	0.69	0.64	0.91	0.94
Treats	0.58	0.58	0.65	0.63	0.81	0.86
Interact	0.62	0.62	0.73	0.72	0.89	0.9
Macro Average	0.59	0.6	0.66	0.66	0.86	**0.9**
Weighted Average	0.65	0.66	0.72	0.72	0.89	**0.92**

Table 5.2: Model evaluation for the second experiment (scores are all F1-measures).

The expectation for the second experiment was that the model would learn the relation between the entity types and the type of relation (i.e., the *Treats* relation consists of a disease and a drug). If the model is able to learn this characteristic, this should be visible in the confusion matrix. The boxes that represent predictions that are not possible when looking at the entity types that are in the sentence should be zero. For example the model should never predict the relation *presents* when the actual label is *treat*. However, when we take a look at the confusion matrix of the best DL model and the BioBERT model in Figure 5.2, the DL model is not able to learn the relation between the entity types and the type of relation. The BioBERT model clearly succeeded to learn the relation between the entity types and the type of relation, since the boxes that we would expected to be zero are indeed zero.

The reason that the Deep learning models are not able to learn the correlation between the entity types and the type of relation has probably to do with the difference in architecture between the Deep Learning models and the pre-trained models. The Deep learning models have different architectures and are making use of convolutional layers and bidirectional LSTM layers. These layers have proven to perform well on textual data and sequences. However, they are not always able to remember all of the important information that occurred during the sequence. The BERT models are making use of a attention mechanism that is looking at the full sequence at once and detecting what information is important. Therefore, the BERT models can quite easily learn the relation between the entity types and the type of relation.



(a) The confusion matrix of the best DL model (complete model)

(b) The confusion matrix of the BioBERT model

Figure 5.2: Comparison of the best DL model and the BioBERT model when looking at the confusion matrix per model

5.3 Final Model Comparison

To summarise the results given in the previous sections the results are summarised in Table 5.3. It can be seen directly that the performance of both BERT models is much higher than the performance of the different Deep Learning models. As expected the performance of the second experiment is higher than the first experiment, however this only holds for the BERT models. The Deep Learning models have equal or a poorer performance. From both the scores and the confusion matrix in Figure 5.2a can be seen that the Deep Learning models are not able to detect the relation between the entity types and the relations as given in Table 3.3.

	Experiment 1		Experiment 2	
Models	Macro F1	Weighted F1	Macro F1	Weighted F1
DL Base Model	0.65	0.71	0.59	0.65
+ position embeddings	0.63	0.7	0.6	0.66
+ shortest dependency path	0.67	0.73	0.66	0.72
DL Complete model	0.66	0.72	0.66	0.72
BERT	0.83	0.86	0.86	0.89
BioBERT	0.87	0.9	**0.9**	**0.92**

Table 5.3: The comparison of the different models in both experiments.

Chapter 6

Conclusion and Discussion

In this section we will answer the research question that is stated in Section 1. This is done by drawing conclusions from the results that were obtained in this research and explained in Section 5. Secondly, topics for future research are discussed while outlining some limitations and improvements to the research.

6.1 Conclusion

Biomedical relation extraction is a complex task within Text Mining. Besides biomedical literature being very complex and therefore hard to understand by both researchers and machines, it is also hard to train models since there is a limited amount of annotated data available. This thesis is focused on this difficult task by testing different methods to answer the research question: *Is it possible to extract relations between entities in biomedical literature by using deep learning or transfer learning?*

To answer this question, it is important that a sufficient number of annotated data samples is obtained. Since, this is not simply publicly available, and manually creating annotated data can only be done by a medical expert, data is extracted from Pub-Med by using Distant Supervision. In Distant Supervision one searches for sentences that describe relations that are already known (seed-facts). By combining a small annotated dataset and a dataset obtained by distant supervision, a dataset containing $15,821$ sentences and 5 different relations was extracted.

In recent developments in Natural Language Processing, mainly Deep Learning and Transfer Learning methods are used. Many Deep learning and transfer learning approaches achieve state-of-the-art performance in most NLP tasks. Therefore, in this thesis we will investigate how deep learning and transfer learning perform in the problem addressed in this thesis. The Deep learning models consist of different features sets that contain word-embeddings, position embeddings and the shortest dependency path between the entities in the text. Different types of layers like a Bidirectional LSTM and convolutional layers are used in these models. For the transfer learning model, the new state-of-the-art language model BERT is used. Two experiments were performed to find the best performing model. In the first experiment the names of the targeted entities were replaced by general mask terms (ENT1 and ENT2), the model can not base the prediction of the name of the entity. In the second experiment the names of the entities were replaced by the name

of the entity type (Chemical or Disease). By doing this fewer predictions are valid (a *treat* relation between two diseases is not valid) and therefore we predict that the model performance would increase.

In Table 5.3 the final model comparison is shown. It is clear that the best performing model was the BERT model where BioBERT was used as a base model and the targeted entities were masked by the entity type names. With a macro F1-score of 0.9 and a weighted F1-score of 0.92 it outperforms all other models by at least 3%. The performance of all BERT models is much higher than the Deep Learning models (a maximum macro F1-score of 0.67), this makes sense since BERT is already trained on huge amounts of text data. If more annotated data was available the Deep Learning models might have a much higher performance.

An interesting result is that the performance of the deep learning models decreased when using the entity types as masks, while the BERT models have increasing performance. These results suggests that the deep learning models are not able to detect the patterns between the entity types and the relations, this also was concluded by looking at the confusion matrix shown in Figure 5.2a. This may have to do with the difference in model architectures between the Deep Learning models and the BERT Models. BERT models use an attention mechanism to remember important information over the full sequence and are therefore able to remember the entity types that occurred in the sentence. The Deep learning methods use RNN and CNN and can therefore easily forget an important word in the beginning of the sentence.

As expected, the experiments show that the BioBERT model outperforms the BERT model. This is not very surprising since the BioBERT model has seen large amounts of biomedical literature and is therefore already able to understand the biomedical language.

To sum up, this thesis focused on finding a model that could detect relations between entities in biomedical literature. Our work has led us to conclude that the BioBERT model is very promising for the biomedical relation extraction task. With a weighted F1-score of 0.92, it achieves high performance on this task as well as a huge difference in performance with the best performing Deep Learning model. Based on the results of the research, we can state that it is definitely possible to extract relations from biomedical literature, especially when using a transfer learning model like BERT. Results so far have been very promising and these results constitute an excellent initial step towards automating biomedical relation extraction.

6.2 Discussion

In this research we proposed a model that can be used to extract biomedical relations. Despite the good performance, there are also multiple possible improvements.

Remove noisy data samples

The first point of improvement would be the dataset, the dataset is extracted by distant supervision and however this is a very effective method is does not come at no cost. The data samples extracted with distant supervision can consist of a noise. Due to the complexity of the sentences it is hard to detect these noisy samples. A possible solution for this would be to make a model that is able to predict the most plausible instances, similar to what has been done by Qin et al. [31]. They explored a Reinforcement Learning strategy to generate the false-positive indicator. This model would be applied on the dataset to remove the noise data samples, this step is performed before the proposed model would be applied on the dataset.

Run-time of BERT model

Another subject that is important to touch upon, is the size and run-time of the model. The BERT model is huge, and needs a GPU to properly run, making predictions of new instances can take around 30 seconds. When the model needs to predict many new instances, this may lead to problems. Reducing the size and decreasing the run-time was outside the scope of this project, but might be necessary when continuing the research.

Negative samples between chemicals and diseases

Currently, the dataset consist of many negative samples. These negative samples are collected by combining the negative samples from the annotated dataset and the *resembles* relation from the dataset by distant supervision. A problem with the negative data samples is that is does not contain any negative relations between a chemical and a disease. Therefore, when a disease and a chemical are the targeted entities in the sentence, the model will never predict a negative relation. These negative relations can be added by randomly taking a sentence with a disease and a chemical and assuming there is no relation between the two. However, this approach is not tested and it could generate many noisy samples.

Multiple relation extraction

It can be that two entities in a sentence are related, but there are also other entities in the sentence that are not involved in any relation or are involved in another relation. When the same sentence would be passed to the model, but different entities are highlighted, this should generate a different prediction. A possible improvement to the model to tackle this issue would be to implement the model as a multiple relation extraction method. When applying this method multiple relations can be extracted from one single sentence. In [45] a new solution is proposed that can extract multiple relations by only passing a single sentence to the model once. In the current model the relations would each go into the model individually. By using this technique the performance could increase since it would look at the locations of the targeted entities in the sentences and look for every entity pair

individually how they are related as shown in Figure 6.1.

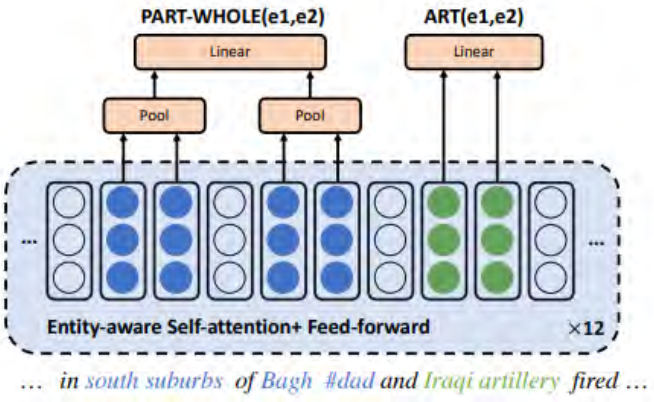


Figure 6.1: Model architecture of a multiple relation extraction model as introduced in [45].

To implement this, the problem needs to be transformed into a different type of problem and adjustments needs to be made to the data and the architecture. When obtaining the data by distant supervision, one must already find out where all of the entities are located and how all of the entity pairs are related. The architecture of the model is different, since there is not a single label, but there are multiple labels and position attention should be implemented. The implementation of the model was already completed, however since the data needed to be changed completely, the results could not be compared with the existing models. Therefore the decision was made to exclude this in this thesis.

Other applications

The problem studied in this thesis is a very general problem and therefore, the solution can easily be applied to other domains as well. Within Accenture there are already several projects and persons interested in applying these techniques in other domains. Furthermore, there are many biotech companies working on implementing knowledge extraction methods by using state-of-the-art techniques to deal with the huge amount of biomedical literature. The research was conducted in a way that it would be easy to use the same techniques in another domain as well, since only a knowledge base needs to be provided and the data is extracted by using this knowledge base.

Bibliography

[1] Recurrent neural network (rnn), long-short term memory (lstm) gated recurrent unit (gru). `http://dprogrammer.org/rnn-lstm-gru`. Accessed: 2010-10-14.

[2] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 85–94, 2000.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Bjorne, Kaewphan, and Salakoski. Uturku: Drug named entity recognition and drug-drug interaction extraction using svm classification and domain knowledge. 2013.

[5] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.

[6] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.

[7] Sergey Brin. Extracting patterns and relations from the world wide web. In *International Workshop on The World Wide Web and Databases*, pages 172–183. Springer, 1998.

[8] Monica Campillos, Michael Kuhn, Anne-Claude Gavin, Lars Juhl Jensen, and Peer Bork. Drug target identification using side-effect similarity. *Science*, 321(5886):263–266, 2008.

[9] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[10] François Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[11] Md Faisal Mahbub Chowdhury and Alberto Lavelli. Fbk-irst: A multi-phase kernel based approach for drug-drug interaction detection and classification that exploits linguistic information. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 351–355, 2013.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[13] Pyysalo et al. (2013). Biomedical natural language processing. `http://bio.nlplab.org`, 2019.

[14] Firth. A synopsis of linguistic theory 1930-1955. 1957.

[15] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[16] Javier Herrero, Matthieu Muffato, Kathryn Beal, Stephen Fitzgerald, Leo Gordon, Miguel Pignatelli, Albert J Vilella, SM Searle, Ridwan Amode, Simon Brent, et al. Database: The journal of biological databases and curation. *Database (Oxford)*, 2016:1–17, 2016.

[17] Daniel Scott Himmelstein, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina L Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, and Sergio E Baranzini. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *Elife*, 6:e26726, 2017.

[18] Lawrence Hunter and K Bretonnel Cohen. Biomedical language processing: what's beyond pubmed? *Molecular cell*, 21(5):589–594, 2006.

[19] Kim, Liu, Yeganova, and Wilbur. Extracting drug–drug interactions from literature using a rich feature-based linear kernel approach. 2013.

[20] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[21] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 09 2019.

[22] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6(2):167–195, 2015.

[23] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. 2015.

[24] Carolyn E Lipscomb. Medical subject headings (mesh). *Bulletin of the Medical Library Association*, 88(3):265, 2000.

[25] Shengyu Liu, Buzhou Tang, Qingcai Chen, and Xiaolong Wang. Drug-drug interaction extraction via convolutional neural networks. *Computational and mathematical methods in medicine*, 2016, 2016.

[26] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. 2009.

[27] Makoto Miwa and Mohit Bansal. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770*, 2016.

[28] Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. Scispacy: Fast and robust models for biomedical natural language processing. *arXiv preprint arXiv:1902.07669*, 2019.

[29] US National Library of Medicine. Pubmed. `https://www.ncbi.nlm.nih.gov/pubmed`, 2019.

[30] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. 2018.

[31] Pengda Qin, Weiran Xu, and William Yang Wang. Robust distant supervision relation extraction via deep reinforcement learning. *arXiv preprint arXiv:1805.09927*, 2018.

[32] Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. 2010.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. 1986.

[34] Sunil Kumar Sahu and Ashish Anand. Drug-drug interaction extraction from biomedical text using long short term memory network. 2015.

[35] Lynn M Schriml, Elvira Mitraka, James Munro, Becky Tauber, Mike Schor, Lance Nickle, Victor Felix, Linda Jeng, Cynthia Bearer, Richard Lichenstein, et al. Human disease ontology 2018 update: classification, content and workflow expansion. *Nucleic acids research*, 47(D1):D955–D962, 2018.

[36] I. Segura-Bedmar, P. Martinez, and C. de Pablo-Sanchez. Extracting drug-drug interactions from biomedical texts. 2010.

[37] Isabel Segura-Bedmar, Paloma Martinez, and Cesar de Pablo-Sánchez. Using a shallow linguistic kernel for drug–drug interaction extraction. *Journal of biomedical informatics*, 44(5):789–804, 2011.

[38] Isabel Segura Bedmar, Paloma Martínez, and María Herrero Zazo. Semeval-2013 task 9: Extraction of drug-drug interactions from biomedical texts (ddiextraction 2013). Association for Computational Linguistics, 2013.

[39] Hye-Jeong Song, Byeong-Cheol Jo, Chan-Young Park, Jong-Dae Kim, , and Yu-Seop Kim. Comparison of named entity recognition methodologies in biomedical documents. 2016.

[40] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM.

[41] Sun, Dong, Ma, Sutcliffe, He, Chen, and Feng. Drug-drug interaction extraction via recurrent hybrid convolutional neural networks with an improved focal loss. 2019.

[42] W. L. Taylor. Cloze procedure: a new tool for measuring readability. *Journalism Quarterly, 30, 415-453.*, 1953.

[43] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. *48th Annual Meeting of the Association for Computational Linguistics*, 2010.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[45] Haoyu Wang, Ming Tan, Mo Yu, Shiyu Chang, Dakuo Wang, Kun Xu, Xiaoxiao Guo, and Saloni Potdar. Extracting multiple-relations in one-pass with pre-trained transformers. *arXiv preprint arXiv:1902.01030*, 2019.

[46] Jonathan J Webster and Chunyu Kit. Tokenization as the initial phase in nlp. In *COLING 1992 Volume 4: The 15th International Conference on Computational Linguistics*, 1992.

[47] David S. Wishart, Craig Knox, Anchi Guo, Dean Cheng, Savita Shrivastava, Dan Tzur, Bijaya Gautam, and Murtaza Hassanali. Drugbank: a knowledgebase for drugs, drug actions and drug targets. *Nucleic Acids Research*, 36(Database-Issue):901–906, 2008.

[48] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. Relation classification via convolutional deep neural network. 2014.

[49] Yijia Zhang, Hongfei Lin, Zhihao Yang, Jian Wang, Shaowu Zhang, Yuanyuan Sun, and Liang Yang. A hybrid model based on neural networks for biomedical relation extraction. 2018.

[50] Yijia Zhang, Wei Zheng, Hongfei Lin, Jian Wang, Zhihao Yang, and Michel Dumontier. Drug–drug interaction extraction via hierarchical rnns on sequence and shortest dependency paths. 2017.

[51] Zhehuan Zhao, Zhihao Yang, Ling Luo, Hongfei Lin, and Jian Wang. Drug drug interaction extraction from biomedical literature using syntax convolutional neural network. 2015.