



UPPSALA  
UNIVERSITET

UPTEC F 19044

Examensarbete 30 hp  
Juni 2019

# Sentiment Analysis of Equity Analyst Research Reports using Convolutional Neural Networks

---

Olof Löfving



UPPSALA  
UNIVERSITET

**Teknisk- naturvetenskaplig fakultet  
UTH-enheten**

Besöksadress:  
Ångströmlaboratoriet  
Lägerhyddsvägen 1  
Hus 4, Plan 0

Postadress:  
Box 536  
751 21 Uppsala

Telefon:  
018 – 471 30 03

Telefax:  
018 – 471 30 00

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

## Abstract

# **Sentiment Analysis of Equity Analyst Research Reports using Convolutional Neural Networks**

*Olof Löfving*

Natural language processing, a subfield of artificial intelligence and computer science, has recently been of great research interest due to the vast amount of information created on the internet in the modern era. One of the main natural language processing areas concerns sentiment analysis. This is a field that studies the polarity of human natural language and generally tries to categorize it as either positive, negative or neutral. In this thesis, sentiment analysis has been applied to research reports written by equity analysts. The objective has been to investigate if there exist a distinct distribution of the reports and if one is able to classify sentiment in these reports. The thesis consists of two parts; firstly, investigating possibilities on how to divide the reports into different sentiment labelling regimes and secondly categorizing the sentiment using machine learning techniques. Logistic regression as well as several convolutional neural network structures has been used to classify the sentiment. Working with textual data requires the mapping of text to real valued values called features. Several feature extraction methods have been investigated including Bag of Words, term frequency-inverse document frequency and Word2vec. Out of the tested labelling regimes, classifying the documents using upgrades and downgrades of report recommendation shows the most promising potential. For this regime, the convolutional neural network architectures outperform logistic regression by a significant margin. Out of the networks tested, a double input channel utilizing two different Word2vec representations performs the best. The two different representations originate from different sources; one from the set of equity research reports and the other trained by the Google Brain team on an extensive Google news data set. This suggests that using one representation that represent topic specific words and one that is better at representing more common words enhances classification performance.

Handledare: Andreas Johansson  
Ämnesgranskare: Prashant Singh  
Examinator: Tomas Nyberg  
ISSN: 1401-5757, UPTec F 19044

## Populärvetenskaplig sammanfattning

Språkteknologi (eng. Natural Language Processing), ett sidoområde inom artificiell intelligens och datorvetenskap, har på senare år blivit ett populärt forskningsområde på grund av de stora mängderna information som dagligen skapas på internet. Ett av de snabbast växande områdena inom språkteknologi är sentimentanalys. Inom sentimentanalys studeras polariteten och attityden inom tal och text men även annan subjektiv information. Detta betyder således att man försöker avgöra den underliggande tonen i denna information och den delas oftast in i positiv, negativ eller neutral ton. Historiskt har främst två olika angreppssätt använts för att analysera sentiment: maskininlärning samt att använda ett lexikon där ord har ett förutbestämt sentimentbetyg. Maskininlärning handlar om att skapa en modell som lär sig från tidigare data. Huvudsyftet med maskininlärningsmodeller är sedan att använda modellen för prediktion på data som inte använts under träning av modellen. På senare år har stora framsteg gjorts inom maskininlärning och således kommer maskininlärning att användas i detta examensarbete.

I detta examensarbete har sentimentanalys applicerats på analyser skrivna av aktieanalytiker. I dessa analyser presenterar analytikern sin syn på bolaget samt ett antal olika estimat på hur analytikern tror att företaget kommer prestera. Dessa estimat kan till exempel vara vinst per aktie, total vinst eller ett målpris för aktien. Analytikern presenterar även en rekommendation på hur den tror aktien kommer prestera givet det nuvarande aktiepriset. Forskning har visat att dessa rekommendationer har ett investeringsvärde och det är därför intressesant att undersöka om det går att förutspå förändringar i dessa rekommendationer. Ett möjligt angreppssätt är att analysera sentimentet i dessa rapporter för att sedan se om det går att sammankoppla förändringar i sentiment med förändringar i analytikerrekommendationer. Ett första steg i denna process är således att undersöka om det går att klassificera sentiment i dessa rapporter. Rapporterna har dock inte en explicit sentimentindelning, varpå detta behöver göras manuellt.

Detta examensarbete består av två delar. Då rapporterna inte har en explicit sentimentindelning undersöks i den första delen tre olika sätt att dela in rapporterna. Dessa är uppgraderingar och nedgraderingar av analytikerrekommendationer, förändringar i vinst per aktie-estimat samt förändringar i relationen mellan målpris och nuvarande pris för aktien. Den andra delen består av att klassificera sentiment i rapporterna genom maskininlärning. De maskininlärningsmodeller som använts är logistisk regression samt falt-ningsnätverk (eng. Convolutional Neural Network). Om modellerna visar på resultat att det går att klassificera sentiment i rapporterna vore nästa steg att undersöka om det går att förutspå analytikerrekommendationer. I detta examensarbete har dock fokus legat på att dela upp samt klassificera sentiment.

Input till maskininlärningsmodellerna kommer vara texten i analytikerrapporterna. Emellertid kan maskininlärningsmodellerna inte behandla texten rakt av, utan det krävs att texten görs om till en numerisk representation. Det finns många olika representationstekniker och i detta examensarbete har ett flertal undersökts. En av dessa representationstekniker kallas Word2vec och utvecklades av Google. Den skapar förtränade ordvektorer som maskininlärningsmodellerna kan använda sig av. Två uppsättningar av ordvektorer har använts. Den första uppsättningen ordvektorer har tränats på analytikerrapporterna, medan den andra uppsättningen är ordvektorer som Google själva har tränat på ett stort antal nyhetsartiklar. I detta examensarbete jämförs samt kombineras dessa två olika uppsättningar av ordrepresentationer.

Resultatet indikerar att uppgraderingar och nedgraderingar av analytikerrekommendationer är den uppdelning som presterar bäst av de tre undersökta uppdelningarna. För denna uppdelning visar sig det även att ett faltningsnätverk presterar bättre än logistisk regression. I det faltningsnätverk som presterar bäst används en kombination av de ordrepresentationer som tränats av Google samt de tränade från analytikerrapporterna. Ordrepresentationerna tränade av Google representerar vanligt förekommande ord effektivt medan de ordrepresentationer som tränats genom analytikerrapporterna representerar ämnesspecifika ord effektivt. Detta antyder då att en förbättring av klassificeringsresultat kan uppnås genom att kombinera ordrepresentationer som tränats på data från olika härkomst.

## **Acknowledgments**

First off, I would like to thank SEB for the opportunity to let me do my master thesis at SEB Solutions and to the entire Solutions team for their warm welcome and making my time enjoyable while working at my thesis. Secondly, I would like to thank Oscar Blomkvist for great collaboration and the valuable discussions we have had throughout the project. Lastly, I would like to thank my supervisor at SEB Andreas Johansson and my subject reader at Uppsala University Prashant Singh for their input and guidance.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Thesis objectives . . . . .	1
1.2	Thesis outline . . . . .	2
1.3	Collaboration with a student from KTH . . . . .	2
<b>2</b>	<b>Theory</b>	<b>3</b>
2.1	Machine Learning . . . . .	3
2.2	Logistic regression . . . . .	3
2.3	Artificial Neural Networks . . . . .	5
2.3.1	Activation functions . . . . .	6
2.3.2	Training a Neural Network . . . . .	7
2.3.3	Gradient-based optimization . . . . .	8
2.3.4	Regularization . . . . .	9
2.4	Convolutional Neural Networks . . . . .	10
2.4.1	Convolutional layer . . . . .	10
2.4.2	Pooling layer . . . . .	12
2.5	Working with textual data . . . . .	12
2.5.1	Bag of Words and term frequency-inverse document frequency . . .	12
2.5.2	Word embeddings . . . . .	14
<b>3</b>	<b>Method</b>	<b>16</b>
3.1	Data set and preparation of labels . . . . .	16
3.1.1	Upgrades and Downgrades of recommendation . . . . .	18
3.1.2	Changes in EPS estimates . . . . .	19
3.1.3	Change in relation between target price and current share price . . .	20
3.1.4	Cleaning the text data . . . . .	21
3.2	Feature extraction . . . . .	21
3.2.1	Bag of Words and term frequency-inverse document frequency . . .	22
3.2.2	Word2vec . . . . .	22
3.3	Machine learning development . . . . .	23
3.3.1	Logistic regression model . . . . .	23
3.3.2	Convolutional Neural Network architectures . . . . .	23
3.4	Training and testing the models . . . . .	26
3.5	Summary of thesis methodology . . . . .	27
<b>4</b>	<b>Results</b>	<b>29</b>
4.1	Word2vec . . . . .	29
4.2	Logistic regression . . . . .	31
4.3	Convolutional Neural Network architectures . . . . .	33
4.3.1	Upgrades and Downgrades of recommendation . . . . .	33
4.3.2	Changes in EPS estimates . . . . .	35
4.3.3	Changes in relation between target price and current share price . .	36
<b>5</b>	<b>Discussion</b>	<b>39</b>
5.1	Evaluation of the results . . . . .	39
5.2	Future work . . . . .	41
5.3	Predicting recommendation changes . . . . .	41
<b>6</b>	<b>Conclusions</b>	<b>43</b>

# 1 Introduction

Approximately two and a half exabytes ( $10^{18}$ ) of unstructured data is created every day on the internet [1]. For a human it would be impractical to analyze this data manually. Thus, research to process this type of data has recently been of great interest. The field of natural language processing (NLP), a subfield of computer science and artificial intelligence, attempts to analyze unstructured data and make sense of it. One of the fastest growing NLP tasks concerns the problem of determining sentiment in human natural language. Sentiment analysis includes the study of people's opinions and emotions towards entities such as organizations, individuals, products and topics. Sentiment measures the polarity and can also be explained as the underlying tone, which usually is categorized as positive, negative or neutral [2].

Sentiment analysis can broadly be grouped into two set of approaches, namely lexicon based-approaches and machine learning based-approaches [3]. Lexicon based sentiment analysis relies on using a sentiment lexicon, where each term in the lexicon has a predefined sentiment score. In machine learning approaches, a model is created that categorises sentiment based on past observations. Machine learning and especially deep learning approaches has shown impressive results across multiple data sets [4] and in recent time, state of the art results when it comes to sentiment analysis [5]. Sentiment analysis has mainly been applied to product reviews due to business owners being able to base decisions on user opinions on their products. However, sentiment analysis has also been applied to news articles and political debates, as well as finance [3].

In the field of finance, sentiment analysis has been applied to annual reports of companies [6] as well as how news articles impact the stock price [7], and how financial social media can be used to predict sentiment [8]. However, it has not been as widely studied when it comes to equity analyst research reports. In a research report, the analyst provides a written analysis on a company combined with several estimates on how the company is likely to perform in the upcoming few years. Such estimates can be earnings per share (EPS), revenue and target price, where target price is a share price the analyst believes is justified for the company. The analysts also deliver a recommendation which is a rating on how the analyst believes the stock of the company will perform given its current share price. These ratings can be such as *Buy/Hold/Sell* which is a three grade scale. *Buy* means that the analyst believes in a substantial improvement in share price. *Hold* means that the analyst thinks the share will trade close to share price and *Sell* means that the analyst believes in a decrease in share price. Research has shown that the recommendations have an investment value [9, 10], and thus it is of interest to investigate if it is possible to predict these changes before they occur. One approach to predict the recommendation changes would be to investigate the sentiment in these research reports and study if change in sentiment relates to change in recommendation.

## 1.1 Thesis objectives

This thesis aims to analyze the sentiment in research reports written by equity analysts. The study will be done at Skandinaviska Enskilda Banken (SEB) which has an internal record of tens of thousands of research reports written by equity analysts at the equity research department. The thesis would serve as a study to investigate if one can determine sentiment in these research reports. If one is able to successfully determine the sentiment, the next step would be to investigate if there exists a relationship between the changes in

recommendation and changes in sentiment. However, in this thesis, the focus will lie on determining and classifying sentiment in research reports.

As discussed previously, the research reports contain a recommendation, as well as other estimates. However, the reports do not explicitly contain a label of sentiment. This means that how one chooses to label the research reports, i.e. deciding on what is positive or negative, is somewhat a task on its own. A part of the project would thus be to investigate several different approaches on how to label the research reports.

The main two approaches to sentiment analysis are lexicon based-approaches and machine learning based-approaches. Recently substantially more promising results have been achieved using machine learning approaches. Another positive aspect with machine learning models is that they do not require a predefined sentiment lexicon. Thus, this thesis will investigate how to determine sentiment using machine learning models. Throughout the years, multiple different machine learning models has been used in the field of sentiment analysis. However recently, deep learning models and especially convolutional neural networks have delivered state of the art performance in sentiment analysis [5]. Thus, this thesis aims to use a convolutional neural network to classify the sentiment in research reports written by equity analysts. As a comparison to the deep learning model, a simple logistic regression model will be used as a baseline model to see if a more complex model performs better, or if a simple model can perform equally well. This means that the project will consist of the following tasks:

- Investigate several different labelling regimes of the research reports written by financial equity analysts
- Develop a machine learning model and specifically a deep learning model to determine sentiment in research reports

## 1.2 Thesis outline

The thesis starts with a section concerning theory, where machine learning in general is explained as well as techniques that will be used, including logistic regression, artificial neural networks and convolutional neural networks. How to work with text as input to machine learning models will also be discussed. This is followed by a section that concerns the methodology, where the data set is explained, including pre-processing of the data and how the labelling of the data has been done. This section will also include a presentation on how the machine learning models are constructed and why certain design choices have been made. Subsequently, the results of the models will be presented and compared. This is followed by a discussion of the results, where future work as well as a discussion on how to predict sentiment changes could be done. Lastly, the work done in the thesis is summarized and concluded.

## 1.3 Collaboration with a student from KTH

This thesis will be done in collaboration with a student from KTH. We have worked on certain tasks together but have done most parts individually. We will use the same data set and thus we have collaborated on the parts that concern the pre-processing of the data set, described in section 3.1. He will also use a logistic regression classifier as a baseline but classify the sentiment using a different deep learning and neural network structure, namely a variant of a recurrent neural network called a Long short-term memory network (LSTM). I will in this report analyze, discuss and conclude my work individually.

## 2 Theory

In this section, what machine learning is and various forms will be explained. This is followed by a description of several machine learning models including logistic regression, artificial neural networks and convolutional neural networks. Lastly, a description on how to work with text input is given where several techniques are discussed.

### 2.1 Machine Learning

Machine learning can be characterized as learning to make predictions based on past observations, thus creating a model which is able to make predictions based on input data [11]. The patterns are learned in a data-driven manner without requiring them to be mathematically specified. The main point of machine learning is to use the trained model on previously unseen data, meaning data not used for training. If the model is able to do this well, one can say that the model generalizes well [12]. There exist different types of machine learning algorithms, mainly supervised and unsupervised. In supervised machine learning, one has input variables  $\mathbf{x}$ , and an output variable  $\mathbf{y}$  and a model is trained to model the output  $\mathbf{y}$ . Thus, we have data that is labelled, meaning that we know what the output is supposed to be, given an input. We can then use this model trained on known data to predict the output when we only know the input. Unsupervised learning, on the other hand, only has access to the input variables  $\mathbf{x}$ , but not output  $\mathbf{y}$ . An example of an unsupervised machine learning problem is clustering which denotes the tasks of organizing data into groups based on some sort of similarity. There also exists semi supervised learning, where one basically makes the use of both labelled and unlabelled data.

For supervised machine learning one often talks about the tasks of classification and regression. Classification involves classifying the data into two or more predefined labels, whereas regression consists of predicting a quantity. This means that classification builds on qualitative outputs that take on values divided into  $K$  classes whereas regression builds on quantitative outputs that take numerical values [13]. In this thesis, focus will lie on classification and in the next sections several models that can be used for classification will be discussed.

### 2.2 Logistic regression

One of the most basic classification algorithms in machine learning is logistic regression. Even though the name contains regression, it is used for classification problems. However, Logistic regression is a generalization of linear regression, which is used for regression tasks. In linear regression, the output  $z$  is described as an affine combination of the input  $x_1, x_2, \dots, x_n$ .

$$z = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n \quad (1)$$

In equation 1 the  $\beta$ 's are the parameters of the model. Thus, one wants to learn the parameters that best model the output  $z$ . In classification the output is a range of classes  $\{0, 1, \dots, K\}$ , where  $K$  is the number of classes. For simplicity, this explanation will be limited to two classes. This gives us a binary classification task, where we are interested to determine the probabilities for the possible outputs given some input. If we let a binary output  $y$  be either 0 or 1, we are interested in the probabilities  $p(y = 0|\mathbf{x})$  and  $p(y = 1|\mathbf{x})$ . These probabilities are functions that sum up to 1 and thus only take values in the interval  $[0,1]$ . In order to facilitate this condition, logistic regression squeezes the

output from linear regression to the interval  $[0, 1]$  using the logistic function as given in equation 2.

$$h(z) = \frac{e^z}{1 + e^z} \quad (2)$$

The logistic function in equation 2 is bounded between  $[0, 1]$ , and we can use this to model the probabilities of  $p(y = 0|\mathbf{x})$  and  $p(y = 1|\mathbf{x})$ . Thus,  $p(y = 1|\mathbf{x})$  can be modelled as in equation 3.

$$p(y = 1|\mathbf{x}) = \frac{e^{\beta^T \mathbf{x}}}{1 + e^{\beta^T \mathbf{x}}} \quad (3)$$

By the law of probabilities, meaning that the probabilities sum up to 1,  $p(y = 0|\mathbf{x})$  immediately follows in equation 4.

$$p(y = 0|\mathbf{x}) = \frac{1}{1 + e^{\beta^T \mathbf{x}}} \quad (4)$$

The next step is to learn the  $\beta$ -parameters of the model from training which will be done using the maximum likelihood approach. Intuitively, we want to find the  $\beta$ -parameters such that inserting the found  $\beta$ -values into equation 3 yields a number as close to 1 as possible for inputs with label 1, and as close to 0 as possible for inputs with label 0. This can mathematically be formalized as in equation 5, where  $l$  is the likelihood function.

$$\begin{aligned} l(\beta) = p(y|\mathbf{x}; \beta) &= \prod_{i:y_i=1} p(y = 1|\mathbf{x}_i; \beta) \prod_{i:y_i=0} p(y = 0|\mathbf{x}_i; \beta) = \\ &= \prod_{i:y_i=1} \frac{e^{\beta^T \mathbf{x}_i}}{1 + e^{\beta^T \mathbf{x}_i}} \prod_{i:y_i=0} \frac{1}{1 + e^{\beta^T \mathbf{x}_i}} \end{aligned} \quad (5)$$

We want to optimize equation 5 with respect to  $\beta$ . However, it is often better to optimize the logarithm of the function due to numerical reasons. This can be done since the logarithm is a monotone function, which means that the argument we want to maximize behaves the same. Thus, the log likelihood function is given in equation 6.

$$\begin{aligned} \log(l(\beta)) &= \sum_{i:y_i=1} (\beta^T \mathbf{x}_i - \log(1 + e^{\beta^T \mathbf{x}_i})) - \sum_{i:y_i=0} \log(1 + e^{\beta^T \mathbf{x}_i}) = \\ &= \sum_{i=1}^n y_i \beta^T \mathbf{x}_i - \log(1 + e^{\beta^T \mathbf{x}_i}) \end{aligned} \quad (6)$$

In the second equality in equation 6, a simplification has been made utilizing the labels chosen earlier, where we have  $y_i = 0$  or 1. To maximize the log likelihood function, we want the gradient of the log likelihood function to be equal to zero as seen in equation 7.

$$\nabla_{\beta} \log(l(\beta)) = \sum_{i=1}^n \mathbf{x}_i (y_i - \frac{e^{\beta^T \mathbf{x}_i}}{1 + e^{\beta^T \mathbf{x}_i}}) = 0 \quad (7)$$

This results in a non-linear system of equations. Such a system lacks a closed form solution. Thus, one needs to use a numerical solver, i.e. an optimization method. Such methods will further be discussed in section 2.3.3. In order to use the model as a classifier, we want to calculate the probabilities of class 0 and 1 and choose the largest out of these which is the final step for a logistic regression classifier.

## 2.3 Artificial Neural Networks

Artificial neural networks or simply neural networks are inspired by the brain and its computation mechanisms consisting of units called neurons [11]. Neural networks can be used for regression and classification and they can somewhat be seen as an extension of logistic and linear regression. The neurons, or nodes as they are called in neural network terminology, multiplies each input and its corresponding weight, sums them, and lastly applies a nonlinear function to the sum, thus creating the output. The nonlinear functions are called activation functions and are explained more in depth in section 2.3.1. If the inputs are expressed as in section 2.2, with  $\mathbf{x} = [1 \ x_1 \ x_2 \ \dots \ x_n]^T$ , the process can then be expressed as in equation 8.

$$z = \sigma(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots \beta_n x_n) \quad (8)$$

In equation 8, the activation function is specified as  $\sigma$ . This process is illustrated in Figure 1 where we can see that the output is constructed using one set of weights multiplied by the input. A neural network is constructed by using several of these nodes, which creates layers called hidden layers where each node is known as a hidden unit.

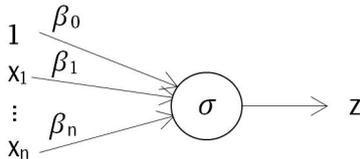


Figure 1: The node with activation function  $\sigma$ , its input  $\mathbf{x} = [1 \ x_1 \ \dots \ x_n]^T$ , its corresponding weights  $\beta_0 \ \beta_1 \ \dots \ \beta_n$  and the output  $z$ .

Each hidden unit has its own set of weights. The output of each hidden unit acts as the input to a similar process as in equation 8 and Figure 1. Using this strategy, we can specify how many hidden layers we want to use. If a neural network contains more than one hidden layer it is considered deep and thus called a deep neural network. Since each hidden unit has its own set of weights, it is possible to generalize equation 8 and specify it for a single hidden unit with index  $i$  and subscript  $l$ , where  $l$  indicates the layer.

$$h_i^{(l)} = \sigma(\beta_{0i}^{(l)} + \beta_{1i}^{(l)} x_1 + \beta_{2i}^{(l)} x_2 + \dots + \beta_{ni}^{(l)} x_n) \quad (9)$$

The output  $z$  will still be calculated as in equation 8. However, it will not use the original input  $\mathbf{x}$ , but instead use the hidden units of the last hidden layer as previously explained. Equation 9 describes the expression for a single hidden unit in a hidden layer. For a hidden layer  $l$  with  $M$  hidden units the structure can be written in matrix form given in equation 10. The weight matrix  $\mathbf{W}$  includes all the parameters and the intercept vector  $\mathbf{b}$  includes the intercept terms.

$$\mathbf{b}^{(l)} = [\beta_{01}^{(l)} \ \dots \ \beta_{0M}^{(l)}] \quad , \quad \mathbf{W}^{(l)} = \begin{bmatrix} \beta_{11}^{(l)} & \dots & \beta_{1M}^{(l)} \\ \vdots & \ddots & \vdots \\ \beta_{n1}^{(l)} & \dots & \beta_{nM}^{(l)} \end{bmatrix} \quad (10)$$

This can be utilized to describe a neural network mathematically. We let  $\mathbf{x}$  be the input to the first hidden layer. The input to the next hidden layer will then always be the previous hidden layer with  $z$  being defined as the output. This follows the same procedure

as discussed earlier. Thus, if we have a network of  $L$  hidden layers, a deep neural network with  $L$  layers can be described as in equation 11.

$$\begin{aligned}
 \mathbf{h}^{(1)} &= \sigma(\mathbf{W}^{(1)T} \mathbf{x} + \mathbf{b}^{(1)T}) \\
 \mathbf{h}^{(2)} &= \sigma(\mathbf{W}^{(2)T} \mathbf{h}^{(1)} + \mathbf{b}^{(2)T}) \\
 &\vdots \\
 \mathbf{h}^{(L-1)} &= \sigma(\mathbf{W}^{(L-1)T} \mathbf{h}^{(L-2)} + \mathbf{b}^{(L-1)T}) \\
 z &= \sigma(\mathbf{W}^{(L)T} \mathbf{h}^{(L-1)} + \mathbf{b}^{(L)T})
 \end{aligned} \tag{11}$$

### 2.3.1 Activation functions

An activation function is a nonlinear function with the objective to simulate the neuron like behaviour in the brain. The activation function is supposed to have a "switch-on" behaviour, which basically means that once the input is greater than a threshold, the output changes state. There are several activation functions that are being used in neural networks, and choosing which one to use has impact on the performance of the neural network. One important condition of the activation functions is that they need to be differentiable. We will come back to the reasoning behind this in section 2.3.2.

The two most common activation functions are the logistic function, also called sigmoid function, and the rectified linear unit (ReLU). The logistic function has previously been discussed in section 2.2. In the early days of development of neural networks, smoother non-linear activation functions were used such as the sigmoid function [14]. However, during recent years, ReLU has become the standard activation function to use in neural networks. This is mostly due to its simplicity and that ReLU typically learns significantly faster than the sigmoid function [15].

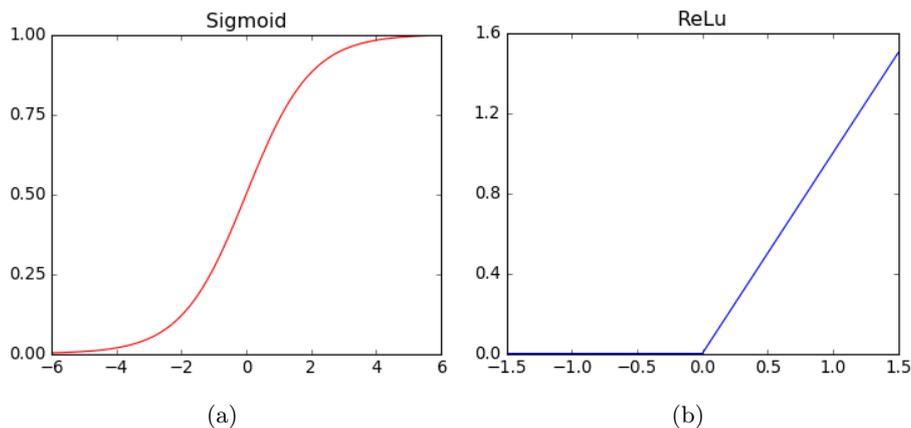


Figure 2: Sigmoid activation function in 2(a) and ReLU activation function in 2(b).

The shape of the sigmoid function and ReLU can be seen in Figure 2.3.1. The sigmoid function is an S-shaped function which transforms the input values into the range  $[0, 1]$ . The mathematical expression can be seen in equation 2. ReLU is a simple activation function that basically ignores values that are below 0 and outputs the value itself if it is

above 0. The mathematical expression is given in equation 12.

$$ReLU(x) = \max(0, x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{otherwise} \end{cases} \quad (12)$$

As previously seen in the output of the neural network, it is run through an activation function. For classification tasks, we are interested in the class probabilities and thus ReLU would not be suitable. Therefore, for binary classification, the sigmoid function can be used since calculating the probability for one class is enough due to that the probabilities sum up to 1. However, if there are more than two classes, a generalization of the sigmoid function is needed which is called the softmax function. If we have  $K$  classes, the softmax of the output  $\mathbf{z}$  is presented in equation 13, which gives us the class probabilities  $p(1|\mathbf{x})$ ,  $p(2|\mathbf{x})$ , ... ,  $p(K|\mathbf{x})$ .

$$softmax(\mathbf{z}) = \frac{1}{\sum_{i=1}^K e^{z_i}} [e^{z_1} \dots e^{z_K}]^T \quad (13)$$

### 2.3.2 Training a Neural Network

When training a neural network one wants to find the best set of parameters like in the case of linear regression. This is done by introducing a loss function,  $L(\hat{\mathbf{z}}, \mathbf{z})$ . The loss function can mathematically be seen as an error term and the network is trained by backpropagating it through the network. The loss function states the loss of predicting  $\hat{\mathbf{z}}$  when the output is  $\mathbf{z}$ . For a neural network, the use of negative log likelihood or cross entropy is commonly used. This assumes that a softmax activation function has been used on the final layer. If we let  $\mathbf{z} = z_1, z_2, \dots, z_K$  be a vector that represents the true output of a data point over  $K$  classes and  $\hat{\mathbf{z}} = \hat{z}_1, \hat{z}_2, \dots, \hat{z}_K$  represent the probabilities that the data point belongs to each class. The mathematical expression for the cross entropy loss can be seen in equation 14.

$$L(\hat{\mathbf{z}}, \mathbf{z}) = - \sum_i^K z_i \log(\hat{z}_i) \quad (14)$$

The training of a neural network consist of a forward pass and a backward pass of the network. The forward pass consists of the steps explained in section 2.3, where the data is led through the network, calculating the output from each hidden layer and hidden unit and ultimately finishing in a prediction after running it through the final activation function. After a prediction has been made, an error rate is calculated where the difference between the predicted output and the true output is calculated using the loss function. The goal of the training is to minimize the loss function with respect to the parameters of the model and this is done in the backward pass. The minimization is done using a combination of backpropagation and gradient based optimization. Backpropagation is used to methodically compute the gradient of the loss function with respect to every parameter in the network [16]. Once the gradients have been calculated, one can use a gradient based optimization technique; such techniques will be discussed in section 2.3.3.

As previously stated, backpropagation is used to compute the gradient of the loss function, thus computing the partial derivatives  $\frac{\partial L}{\partial w_{ik}^l}$  and  $\frac{\partial L}{\partial b_i^l}$  where  $i$  indicates the  $i^{\text{th}}$  neuron and  $l$  the layer,  $w$  any weight in the network and  $b$  any intercept. To compute  $\frac{\partial L}{\partial w_{ik}^l}$  we first

differentiate  $L$  with respect to  $\hat{\mathbf{z}}$  which yields,

$$\frac{\partial L}{\partial \mathbf{w}} = \sum_{i=1}^K \frac{\partial J}{\partial \hat{z}_i} \frac{\partial \hat{z}_i}{\partial \mathbf{w}} = \left( \frac{\partial L}{\partial \hat{\mathbf{z}}} \right)^T \frac{\partial \hat{\mathbf{z}}}{\partial \mathbf{w}} \quad (15)$$

where  $\hat{\mathbf{z}}$  is defined for  $K$  number of outputs. The last term in equation 15 can explicitly be computed by differentiating the last term in equation 11 where the result is seen in equation 16.

$$\frac{\partial \hat{\mathbf{z}}}{\partial \mathbf{w}} = \frac{\partial \sigma}{\partial \mathbf{y}^L} \odot \mathbf{W}^L \frac{\partial \mathbf{h}^{L-1}}{\partial \mathbf{w}} \quad (16)$$

In equation 16,  $\mathbf{y}$  denotes the input to the last activation function i.e.  $\mathbf{y} = \mathbf{W}^{(L)T} \mathbf{h}^{(L-1)} + \mathbf{b}^{(L)T}$  and  $\odot$  represents element-wise multiplication called the Hadamard product. By a similar differentiation, errors can be backpropagated all the way to the input giving the result seen in equation 17.

$$\frac{\partial \mathbf{h}^l}{\partial \mathbf{w}} = \frac{\partial \sigma}{\partial \mathbf{y}^l} \odot \mathbf{W}^l \frac{\partial \mathbf{h}^{l-1}}{\partial \mathbf{w}} \quad (17)$$

In the notation, no subscript has been added to the activation function but as discussed in section 2.3.1, different activation functions can be used. There may also be different activation functions for different layers. Furthermore, the equations can be summarized in a four step method inspired by the work of Michael A. Nielsen in [17] seen in equation 18, where  $\delta_i^l$  corresponds to  $\frac{\partial L}{\partial z_i^l}$ .

$$\begin{aligned} \delta^L &= \nabla \sigma(\mathbf{y}^L) \nabla L(\hat{\mathbf{z}}) \\ \delta^l &= \nabla \sigma(\mathbf{y}^l) ((\mathbf{W}^{l+1})^T \delta^{l+1}) \\ \frac{\partial L}{\partial w_{ik}^l} &= \delta^l h_k^{l-1} \\ \frac{\partial L}{\partial b_i^l} &= \delta^l \end{aligned} \quad (18)$$

### 2.3.3 Gradient-based optimization

As explained in section 2.3.2 the goal of training a neural network is to minimize the loss function which means finding the weights and intercepts that makes the loss as small as possible. This is done using optimization techniques and the most simple but still useful technique is called gradient descent. The idea behind gradient descent is to minimize the loss function by altering the weights with the opposite sign of the gradient [18]. The concept is given in equation 19, where  $\eta$  represents the learning rate,  $L$  the loss function and  $w$  any weight in the network. The learning rate describes the size of the step one wants to take and can be chosen in several ways, the most common being a small positive constant. This basic idea can also be used when optimizing for the coefficients in logistic regression discussed in section 2.2.

$$w_{\text{new}} = w_{\text{old}} - \eta \frac{\partial L}{\partial w_{\text{old}}} \quad (19)$$

The problem with gradient based optimization is the computation of the gradients. In the case for logistic regression this is rather easy and one can simply follow equation 7 where

one needs to solve a system of non-linear equations. However, in the case for neural networks this is a much more complicated process. Luckily, the gradients can be calculated using backpropagation as explained in section 2.3.2. The optimization method previously discussed relies on using all data points in the data set. In a neural network, this process becomes infeasible if one wants to calculate the gradient for all data points in every iteration. Due to this reasoning, the optimization is done in minibatches, where a minibatch is a sample drawn uniformly from the data set. This means that the loss function will be minimized as an average over the number of samples in the minibatch. This approach and modification is often used in deep learning and is called stochastic gradient descent [19].

In the research field of stochastic optimization several models have been developed that tries to increase the performance of gradient descent. Two such methods are AdaGrad [20] and RMSProp [21]. Both RMSProp and AdaGrad are adaptive methods, where adaptive means that during training, the methods automatically and individually tune the learning rate of each parameter. Recently, the method that has become the most popular to use when training neural networks is the method Adam. Adam is a combination of AdaGrad and RMSProp and has showed promising results compared to AdaGrad, RMSProp as well as stochastic gradient descent [22].

### 2.3.4 Regularization

When designing a neural network and especially a deep neural network, one might end up with more parameters than what the data consists off. This makes the neural networks prone to overfitting. This in turns means that we have a model that is too flexible in regards to the complexity of the data. To combat overfitting one can use regularization techniques to reduce the complexity of the model. Regularization can be implemented in several ways. One technique is to add a penalty term to the loss function that penalizes large parameter values. Another way of penalizing the parameters is to enforce an absolute upper bound of the weights vector in every node. This works by updating the weights as normal and then enforce the weight vector of every node to satisfy the condition  $\|w\|_2 < c$  where  $c$  denotes the upper bound [23].

A simple way of regularization is early stopping. The idea behind early stopping is to stop the training earlier than initially specified with the intention to stop training before the model starts to overfit. Early stopping criterion's can be ones such as when the validation accuracy or validation loss has not improved for a few epochs, however the training accuracy might still improve. An epoch is referred to as when one complete pass of the training data has been done [12].

Another way to regularize neural networks that is being widely used is called dropout. The main idea with dropout is to randomly drop out units and their corresponding weights from the network, with the purpose of preventing co-adaption of features [24, 25]. This basically means that we are limiting the nodes to be able to create complicated interactions between each other and also making the network less dependent on individual nodes. Dropout also serves another purpose where it simulates the process of training many different networks and averaging them by disregarding nodes with a certain probability [25]. Dropout is only applied to the network while training and thus, when the network is tested, the full network with all nodes and corresponding weights is used.

## 2.4 Convolutional Neural Networks

Convolutional neural networks are a type of neural network being used for situations when the input has a grid-like topology. The grid can for example be of 1-D nature and consist of time series or text and of 2-D nature consisting of images with a grid of pixels. The convolutional neural network builds on the mathematical operation convolution, which is a special linear operator. Convolutional neural networks consist of at least one layer utilizing convolution combined with a concept called pooling. Convolutional neural networks also uses the concept of several channels. These topics will be discussed further in the following sections.

### 2.4.1 Convolutional layer

The mathematical operation convolution can be described as an operation on two functions which expresses how one of the functions is modified by the other. In the discrete context of a convolutional neural network, one of the functions is the input whereas the second function is called a kernel or filter and the process can mathematically be explained as in equation 20.

$$S(i) = (K * I)(i) = \sum_n I(i - n)K(n) \quad (20)$$

Where  $I$  denotes the input,  $K$  the filter or kernel and  $*$  the convolution operator [19]. The filter  $K$  is what can be seen as being convoluted with respect to the input. In the convolution operator, one of the functions is flipped and slid over the other function. However, in the sense of a convolutional neural network, it is more intuitive to think of placing the filter on top of the input. This similar process is technically known as cross-correlation and this is the process implemented in a convolutional neural network and is often referred to as convolution [19]. The mathematical expression for cross-correlation is given in equation 21.

$$S(i) = (K * I)(i) = \sum_n I(i + n)K(n) \quad (21)$$

The process of cross-correlation is similar to convolution but with one exception which is that the kernel has been flipped in the convolution. Cross-correlation basically implements a sliding dot product and thus, what basically happens is that the filter slides across the input and performs element wise matrix multiplication. The process can be seen illustrated in figure 3, where an input is convolved with a 2x2 filter. The bold bars indicate the values that have been convolved with the filter and what the resulting output for that specific convolution is.

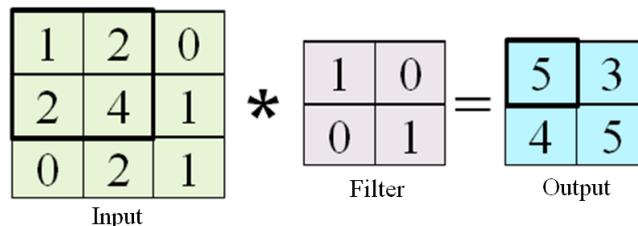


Figure 3: Convolution performed with a filter size of 2x2 and a stride of 1. The bold bar in the matrices indicate which values have been convolved and what the resulting value is.

In fully connected layers, which are explained in 2.3, all input units are connected to every hidden unit. However, when the filter slides across the inputs, connections are being made in smaller, more localized regions of the input. Only the individual input values that the filter covers in that instant will be used as inputs instead of using all inputs. Each input that is covered in the filter at that instant will learn a weight and it will also learn an overall intercept [26]. If we have a filter with size  $2 \times 2$  as in figure 3, and we consider the  $i^{\text{th}}, j^{\text{th}}$  hidden unit as input, the output runs through an activation function  $\sigma$  can mathematically be describes as in equation 22.

$$\sigma(b + \sum_{k=0}^1 \sum_{l=0}^1 w_{k,l} h_{i+l, j+k}) \quad (22)$$

Where a similar notation has been used as in equation 11, with intercept  $b$  but  $w$  denoting the weights in the filter. One important thing to note is that all hidden units will use the same filter weights. This means that parameters will be shared throughout the layer. In [19], this is described as one of the main advantages of using convolutional neural networks. These filter weights will be updated in a similar manner as the other weights in a neural network as described in section 2.3.2 using backpropagation.

When convolving over the input data, it will work fine when the convolution happens in the centre of the input data, but what happens when we apply a filter of size larger than 1 to the first or last word in for example a 1-D setting? There will not be enough inputs since the filter is of larger size than 1. To combat this one can make the use of padding. This basically means that we add zeros to the beginning and end of the input enabling the filter to be applied to the full input. Adding padding with zeros is called wide convolution, and not using padding is called narrow convolution [27]. The use of narrow convolution reduces the output size, which can be problematic when using large filter sizes. Adding zero-padding also help keeping the spatial dimensions intact.

In the discussion earlier about convolution, the filter has been shifted by one in each step. However, in a convolutional neural network, it is not always the case. The filter can be shifted by more than one in each step. The process of how much the filter is shifted in each step is known as the stride size. A stride size larger than one leads to a smaller output size due to fewer applications of the filter.

In this section, the process of having one filter sliding over the data has been discussed. However, it is common to deploy several different filters of the same size, but also filters of different sizes. This will extend the network and introduce channels. Each filter produces its own set of hidden units, where each filter will construct a separate channel. The convolutional network can also make use of channels in another way, namely the use of several input channels. In the case of 2-D grids, color images uses three input channels that represent red, green and blue colors. In 1-D grids, and especially text, one can make use of several channels where the inputs, which can be words, can be expressed in different ways, how to express words as inputs to a neural network will further be discussed in section 2.5

### 2.4.2 Pooling layer

The convolutional layer is usually followed by a pooling layer with the goal to reduce the spatial size of the input but keeping the most important information [11]. The pooling works in a similar way as the convolutional layer where instead of convolving with the purpose of performing matrix multiplication, a single value is extracted depending on some scheme. How to choose which value to keep can be done in different ways and two of the most common ones are max pooling and average pooling. In max pooling, the maximum value of the pooling size is chosen whereas in average pooling, an average across the values is computed. Pooling can also be implemented with different strides as discussed in 2.4.1. Max pooling can be implemented in a global setting where the maximum value for all hidden units in the channel is chosen which is called global max pooling. The process of max pooling and global max pooling can be seen in Figure 4. A max pooling with size 2x2 and stride 2 has been used, where the colors indicate from where the values has been pooled from. Since pooling reduces the spatial dimensions it also reduces the overall number of parameters of the model. This improves the computational efficiency of the network [19] and also helps to control overfitting [23].

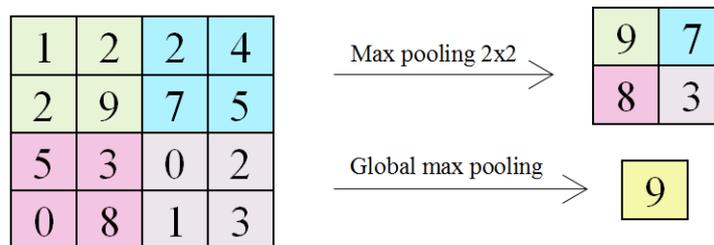


Figure 4: Max Pooling with size 2x2 and stride 2 and global max pooling on a matrix of size 4x4. Colors for max pooling indicate from what region the data is pooled from. Yellow indicates that it is pooled from the full matrix.

## 2.5 Working with textual data

When analyzing documents, the input will be the text in the documents. However, the machine learning techniques discussed earlier needs a numeric input and cannot use the raw text data right away. The process of mapping textual data to real valued interpretations is called feature extraction [11] and thus, the transformed text to real valued values is called features. In this section, how feature extraction can be done will be discussed which will include a discussion on Bag of Words-models, the use of term frequency-inverse document frequency (tf-idf) and word embeddings.

### 2.5.1 Bag of Words and term frequency-inverse document frequency

Bag of Words is probably the simplest feature representation technique that exists. It uses frequency of words in documents where the order of words is neglected. Given a set of documents, the model creates a corpus which will have the same dimensions as the number of individual words across all the documents. A corpus is a list of words or documents depending on the application. Every word will have a mapping to a specific place in the corpus which means that a word will be a one-hot encoded vector. The vectors of words will then be combined into document vectors. The model is best explained with

an example containing two small documents which basically will be two separate sentences illustrated in equation 23.

1. This is a positive analysis
  2. This analyst was negative in his analysis
- (23)

The two sentences will create a corpus containing,

['This', 'is', 'a', 'positive', 'analysis', 'analyst', 'was', 'negative', 'in', 'his']

where each word will have its own one-hot encoded representation. 'positive' will for example be given by the vector  $\mathbf{v}_{positive} = [0, 0, 0, 1, 0, 0, 0, 0, 0]$ . The sentences in equation 23 can numerically be expressed as vectors consisting of element wise summation of the one-hot encoded word vectors yielding the vectors seen in equation 24.

1. [1, 1, 1, 1, 1, 0, 0, 0, 0]
  2. [1, 0, 0, 0, 1, 1, 1, 1, 1]
- (24)

We now have a numeric representation where the frequency of each word is recorded, but as one can observe, the ordering is lost. In the sentences in equation 23 we have considered each word individually, but what if we consider grouping of the words? These groups are called word-ngrams or ngrams where n stands for how many words are grouped together. The ngrams are supposed to be more informative since it captures structures and context. If we consider bigrams where we have grouped the words in pairs of two, the Bag of Words corpus for the sentences in equation 23 can be seen in equation 25.

['This is', 'is a', 'a positive', 'positive analysis', 'This analyst',  
'analyst was', 'was negative', 'negative in', 'in his', 'his analysis']

(25)

In the example given it does not seem to be more informative, however, consider the example 'New York'. By just considering individual words the context would be lost since it would be 'New' and 'York' with no possible way of keeping track of the ordering. However, if we consider 'New York' as a bigram the information is not lost. It is common to group words in documents by  $n = 2$  up to  $n = 4$  or  $n = 5$  but not further since it is not feasible due to sparsity issues [11]. As one increases the words in the ngrams, fewer of each ngram will be found and thus machine learning techniques will have a hard time finding a pattern among the ngrams.

An extension of the simple Bag of Words-model is to make use of term frequency-inverse document frequency. This solves the problem with a Bag of Words-model that it assigns equal importance for each individual word in a text. This is solved by tf-idf which performs weighting of all words [28]. The weighting is done in two steps. Consider a document  $d$  in a corpus  $D$ . First, the frequency  $f_{w,d}$  of a word  $w$  in the document  $d$  is calculated by simply calculating the number of times the word occurs in the document. The next step is to calculate the inverse document frequency which is given by  $\log(|D|/f_{w,D})$  where  $|D|$  is the size of the corpus and  $f_{w,D}$  represents the number of documents that the word appears in. The tf-idf score of a word  $w$  in a document  $d$  in a corpus  $D$  can be seen in equation 26.

$$\text{tf-idf}(w, d, D) = f_{w,d} \cdot \log\left(\frac{|D|}{f_{w,D}}\right)$$
(26)

The idea behind tf-idf weighting is that a word that is rare in  $D$  but common in an individual document  $d$  is given a high score and deemed important whereas a common word that can be found in many documents is deemed unimportant. This method will help filter out words such as 'the' and 'a' which usually are common in an individual text but also across all texts. The tf-idf score would then be used by replacing the one in the one hot encoded vector with the tf-idf score of the word.

### 2.5.2 Word embeddings

In the previous section, the words have been represented with a single element either by their count or by a weighting scheme that awards higher score to words that are common in a specific text but not common across texts. However, these models have their limitations where a more advanced approach would be to represent each word as a dense vector instead of the sparse nature of the Bag of Words-model. This means that each word is embedded into a  $d$  dimensional space and is represented as a vector from that space. The size of the space and thus the vectors are independent of the size of the corpus and are usually much smaller than the corpus itself. These word vectors are known as word embeddings.

The most well-known model to use to construct word embeddings is called Word2vec and it is able to capture semantic relationships using only a few hundred dimensions. This model was developed in 2013 by the Google Brain team which was led by Thomas Mikolov [29]. Word2vec is in fact two separate methods that achieve similar result but using different approaches. Both methods are built on unsupervised neural network models that are being trained on a large corpus with the end goal of achieving vector representations of words, where similar words have similar representation. This means that similar words will have word vectors that are close to each other in the used vector space. This can be explained by an example that Mikolov used in their original paper. The result of the calculation  $\mathbf{v}_{Paris} - \mathbf{v}_{France} + \mathbf{v}_{Italy}$  is most similar to the vector  $\mathbf{v}_{Rome}$ , which means that the result is closest to the vector representation of  $\mathbf{v}_{Rome}$  even though the model was not trained using any labels what so ever [29]. The similarity between words can be calculated using cosine similarity, which measures the cosine of the angle of the word vectors [11]. If we consider vectors  $\mathbf{u}$  and  $\mathbf{v}$ , the cosine similarity can be expressed as the dot product divided by the magnitude of the vectors as seen in equation 27.

$$Similarity = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|} \quad (27)$$

The two models proposed in [29] are called Continuous Bag of Words (CBOW) and Continuous Skip-gram (Skip-gram). Both models are built on a neural network model with a structure containing one input layer, one hidden layer and one output layer. However, the non-linear activation function which usually is used on all layers in a neural network is removed from the hidden layer. In CBOW, the neural network tries to predict a word given some history and future words. Given some input words it uses a projection layer, similar to a hidden layer, but the weight matrix and the projection layer is shared for all words. The process then works as in a neural network where the goal is to minimize the cost function, but in CBOW the cost function is averaged over the history and future words considered. The network is finished using a softmax function with  $n$  classes, where  $n$  is the number of words in the corpus. After the network has finished its training, the output layer is removed and the weights trained in the hidden layer represent the vector

representation of the word. The Skip-gram model works in a similar fashion but tries to predict the opposite, i.e. given a word it tries to predict the history and future words. The main idea of the models can be seen in Figure 5, where two history and two future words have been used.

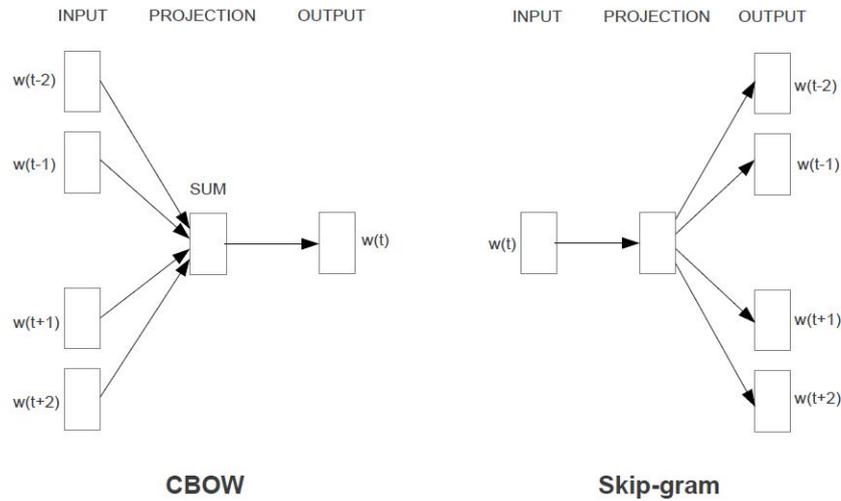


Figure 5: Illustration of the CBOW and Skip-gram models. Two history and two future words are being considered. Illustration taken from [29].

One drawback of Word2vec is that the model requires an enormous amount of data to be trained on. However, since the models are unsupervised and hence need no label, the models can be trained on any text that one can access. The models also concern languages, and as long as the text does not differ to much from normal language, i.e. for example no use of sarcasm, one can use vector representations that have been pre-trained by someone else. In this thesis, both pre-trained Word2vec representations as well as vector representations that have been trained from the input data will be used and tested. One of these pre-trained Word2vec representations contains over three million unique individual words and has been trained on about a hundred billion words from a Google news data set as explained in [30] which was done by Mikolov and the Google Brain team.

### 3 Method

In this section, the data set used for the machine learning models will be explained as well as how the text has been pre-processed and how the data has been divided into labelling regimes. This is followed by a section on how features are extracted from the text in the research reports. Subsequently, the methods developed and a motivation for choices made will be explained in detail. Furthermore, a discussion on how the methods are trained and evaluated will be presented. The section is finished with a summary of the workflow. Pre-processing and division of labels has been done in R, whilst the feature extraction and machine learning model development has been done in Python.

#### 3.1 Data set and preparation of labels

The data set used consists of more than 84 000 research reports in PDF format produced by the equity research department at SEB. The reports include documents such as analysts research reports on individual companies, macro analyses, sector analyses as well as documents concerning credit. The full list of documents can be seen in Table 1. The document ID represent a unique reference key and document type represent a brief explanation of the document. IBESCode represent what the document concerns, whether it is a report on a single company, an industry analysis, a continuous update or a strategy document. Document area is a key that divides the documents further into different sections including equity, credit, macro and corporate related documents.

Table 1: Document ID, document type, IBESCode and document area for the different documents in the data set.

Document ID	Document type	IBESCode	Document area
1	News Comment	COMPANY	1
2	Result Analysis	COMPANY	1
3	Company Update	COMPANY	1
4	Sector Analysis	INDUSTRY	1
5	Macro	STRAT	3
6	Weekly Comment	WEEKLY	1
7	Daily Comment Comment	DAILY	1
8	Results Preview	COMPANY	1
9	Presentation	COMPANY	1
10	Sector Comment	INDUSTRY	1
12	Special Situations & Pair Trades	INDUSTRY	1
13	IPO	COMPANY	1
14	Results Reaction	COMPANY	1
15	Instant Credit	COMPANY	2
16	Company Update - Credit	COMPANY	2
17	Credit Weekly	INDUSTRY	2
18	Equity Strategy	STRAT	1
19	Sector Update - Credit	INDUSTRY	2
20	Sector Comment - Credit	INDUSTRY	2
21	Credit Strategy	STRAT	2
25	Company Update - Corporate	COMPANY	4
26	News Comment - Corporate	COMPANY	4
27	Results Analysis - Corporate	COMPANY	4
28	Results Preview - Corporate	COMPANY	4
29	Results Reaction - Corporate	COMPANY	4
30	Company Comment	COMPANY	1
31	Company Comment - Credit	COMPANY	2
32	Company Comment - Corporate	COMPANY	4

In this thesis, the focus lies on determining the sentiment in research reports and thus we are only interested in equity reports that only consider one single company. This means that the full data set will not be used. To filter out the documents we first chose the documents that had an IBESCode equal to COMPANY. Hence we ignored document types such as daily and weekly updates as well as industry and strategy updates. Furthermore, only documents of "Document area" 1 was considered. This made us ignore documents that concern credit, macro and corporate documents. At first glance, one may think that documents with the corporate tag should be considered. However, these documents do not include an analyst recommendation and is thus not of interest for us. This means that the remaining documents ID's were 1, 2, 3, 8, 9, 13, 14, 30, which corresponds to *News Comment*, *Result Analysis*, *Company Update*, *Results Preview*, *Presentation*, *IPO*, *Results Reaction* and *Company Comment*. The types of documents are pretty self-explanatory expect for IPO. IPO stands for initial public offering and is when a company starts to get publicly traded on a stock exchange. Such a document thus relates to when companies are first introduced and if a company does an IPO, a document of type IPO will be the first analysis.

The next step after a subset of documents had been chosen was to extract the text from the documents. At first, the text was extracted directly from the PDF's. However, this method ran into several problems. We are interested in the raw text, but the analyses contain figures and tables that can make the extraction of the text problematic. The tables will be extracted as text which will result in table headings and text that is incoherent and does not represent full sentences. Similar issues arise when trying to extract text from figures. A common denominator for tables and figures was that they contained few words on each line. Thus, it was experimented with setting up a condition that only considered lines containing at least  $n$  words. However, when only considering each line on its own, the last line in some paragraphs was excluded if it contained less than  $n$  words. Each figure and table was contained in its own paragraph in the texts. Thus, an additional condition was added were a paragraph needed to have at least  $m$  words in it. The values of  $m$  and  $n$  were empirically tested and 5 was found to be a good value for  $n$  and 30 was found to be a good value for  $m$ . The parameters  $n$  and  $m$  were evaluated with the goal of keeping as much text and removing as much content from the tables and figures as possible. Using this method, a fairly good result of extracting the text was achieved. However, we could not guarantee the generality of the approach since testing it required us to read the document and compare if the text had been extracted well which was a time consuming process. Another problem encountered was the time it took to extract the text from an individual PDF. This was roughly 10 seconds per PDF and with 84000 documents this would have taken a substantial amount of time.

All documents had a similar front page that consisted of several bullet points that represents a summary of the report. The bullet points are plain text and contain no tables or figures. This means that no modification needs to be done in order to extract the text. Luckily, this text is also easily accessible via a separate data base. Thus, instead of extracting the text from the PDF's, one could possibly only use the bullet points on the front page of the document. This would simplify the extraction process significantly as well as avoiding the time needed to extract the text from the PDF's. This also made the text more similar in word length and shorter overall. This is advantageous when training the neural networks since the input needs to be of equal length and shorter texts means shorter training time, this will further be discussed in section 3.3.2. One could also argue

that the summary represents the main points made throughout the document and thus will be able to represent the sentiment in a similar way. Thus, it was chosen to only consider the bullet points on the first page and from here on, when referring to the documents and the text in the documents, only the bullet points on the front page are considered.

After the correct document types had been selected, there still existed reports that contained no recommendation. This could be due to a company not being covered by the analysts anymore and thus resulting in an unrated status. There also existed reports that had no text, this was due to storage errors in the data base system. These documents were removed which finally reduced the data set to 67485 research reports. These reports served as the basis for the input to the machine learning models and feature extraction methods.

The reports are from a time span of approximately 20 years. Due to the long time period, two different recommendation systems have been used by SEB. These contain both three graded scales but also four graded scales. The three graded scale is the most common one and several different names for the recommendation system has been used. However, what names being used has no actual meaning as long as the amount of steps is the same. Rating 1 is the best out of the three and the analyst see a substantial improvement in share price. Rating 3 indicates that the analyst see a decrease in share price, and thus rating 2 is between rating 1 and 3 and means that the analyst believes the share is fairly valued and it is trading close to target price. The four graded scale was quite rare and only occurred for some companies in the first few years of the data set. In the case that a 4 graded scale was employed, the last two ratings, 3 and 4, indicated a similar view on the share by the analyst. Only 50 were rated a 4 and to generalize, all reports rated 4 were merged with the reports being rated 3. This made it possible to only consider three ratings and treat all reports similarly.

Once the text had been extracted from the PDF's and the filtering and pre-processing of the documents had been done it was time to divide the text into different labelling regimes. As discussed earlier, the documents contain no explicit sentiment label and thus the regimes need to be defined by oneself. Three different labelling regimes were considered, namely;

- Upgrades and downgrades of recommendation, treating an upgrade as positive and a downgrade as negative sentiment
- Changes in earnings per share (EPS) estimates, treating an increase as positive and a decrease as negative sentiment
- Changes in relation between target price and current share price, treating an increase in relation as positive and a decrease as negative sentiment

The process of how these regimes were constructed will be explained more in detail in the upcoming sections as well as a discussion on why they indicate positive or negative sentiment.

### **3.1.1 Upgrades and Downgrades of recommendation**

The first labelling regime was to divide the research reports into labels of recommendation upgrades and downgrades. For an individual company, all reports were compared

to the previous report to see how the recommendation had changed. This was done by calculating the change in recommendation leading to a numerical rating between -2 and 2. The result was multiplied by -1 to more intuitively represent that a positive change is represented by a positive number. This means that an upgrade from 3 to 1 will be represented by 2 instead of -2 and similarly a downgrade from 1 to 3 will be represented as -2. However, for the majority of the reports, there was no change of recommendation and thus they resulted in 0. Most of the rating changes were either 1 or -1. Now the numerical ratings could be used to divide the reports into *Upgrades* and *Downgrades* where a positive rating indicated an upgrade and a negative rating indicated a downgrade. As previously discussed, some companies stopped being covered by analysts which resulted in no recommendation. However, analysts could start to cover companies again that had previously been uncovered. This would lead to periods where companies had no recommendation. If this period was too long, we might end up comparing recommendations where the fundamentals has changed too much. Thus, if it was more than 1 year between recommendations, that specific case was not treated for evaluation if being an upgrade or downgrade. 1 year was chosen because in that time, 4 quarters have passed and in that time the estimates made by the analyst might be outdated.

The reports being labelled as upgrades would be considered positive in their sentiment and the reports being labelled as downgrades would be considered negative in their sentiment. This is based on how the recommendation system works. If the recommendation is upgraded, the analyst is obviously more positive than earlier and thinks the share will perform better. If the recommendation is downgraded, the analyst is more negative than earlier and thinks the share will perform worse.

### 3.1.2 Changes in EPS estimates

The second labelling regime investigated was how the earnings per share (EPS) estimates has changed. These estimates are done on a year basis and estimates of several years forward will be made. This means that if we have a report from 2019, estimates of year-end 2019 will be made, as well as estimates of year-end 2020, 2021, ... . How many year-end estimates there will be depends on the company but throughout the data set, at least three year-end EPS estimates existed. To be able to compare EPS estimates between companies, we need to use some kind of normalization. However, calculating it as a percentage of change could be problematic since companies can report negative earnings. Thus, we instead decided to divide with the current share price. This results in a type of earnings yield [31]. However, a normal earnings yield considers the reported EPS divided by the current share price whereas we are using an estimate of future EPS but divide it with the current share price.

First, the calculation of change in EPS estimates was done in a similar way as for upgrades and downgrades, where the EPS estimate in a report was compared to the one in the previous report. A maximum delay between EPS estimates of 1 year was also introduced with similar argumentation as in the case for upgrades and downgrades. The next step was to divide with the price of the share for the report date. However, this required some modifications. The first one was that the EPS estimates and share price was sometimes in different currencies, and thus the EPS estimates and share prices was converted to American dollars to easily be comparable. The second one was that an unadjusted time series for share price had to be used in order to ignore the adjustments made for stock splits that could have happened throughout the time period. Once these modifications

had been done, the EPS estimates could be divided by the modified share price.

When studying the change in EPS estimates, the two most recent consecutive years were considered. To consider two years instead of only one was chosen since if we are late into a year, that estimate will almost be known and thus it is interesting to also investigate the consecutive year. If the change direction between the two years were different, the EPS estimate for the report was not considered and thus received no label. In the event that the two were of the same sign, an average was calculated. If both of the estimates were positive, a positive sentiment label was given and if they were negative, a negative sentiment label was given.

The motivation that an increase in EPS estimate is positive is based on a similar argument as in the case of upgrades and downgrades. If the analyst believes the company will increase their earnings, a sign that obviously is positive, the EPS estimate will be increased and thus can be seen as positive. If the analyst believes the company will decrease their earnings, the analyst will decrease the EPS estimate and thus can be seen as negative. However, small differences in EPS estimate might have to do with fluctuations in share price, which is not an indication on what the analyst thinks about the company, thus a threshold is needed. The threshold will filter out numerical differences that are too small and only differences above or below the threshold will be considered.

### **3.1.3 Change in relation between target price and current share price**

The third and final labelling regime investigated was the change in relation between target price and current share price, where current share price is the price at the date of the report. Target price is an estimate the analyst provides and is an estimation of what the analyst think the share is currently worth. Thus, if the target price is above share price, the analyst believes in an increase in price and if target price is below share price, the analyst believes in a decrease in price. This is similar to upgrades and downgrades but with two major differences. The first difference is that target price can be changed without the change of a recommendation which in theory will lead to a larger subset. Secondly, when looking at a relation, a threshold can be introduced, much like in the case for EPS estimates. This threshold can be varied depending on how big of a change one is interested in.

First off, the relation between target price and current share price on the date of the report was calculated. However, as in the case for changes in EPS estimates, share price and target price was converted to US dollars to combat differences in currencies. Splits could also be problematic and thus an unadjusted time series of share price was used. Large values of the relation were observed which were the result of data base errors. Such errors could be that the current price for the share and the target price varied heavily due to the fact that the split date had been reported differently. This lead to large values being observed in the relation and thus, only relations between 0.5 and 2 were considered. This was introduced on the intuition that it is very unlikely that the analysts believe in a doubling or halving in price. This means that we might filter out reports with no data base errors in their relation. However, we believe the cases this happens to be few. After the relation had been calculated, the relation in a report was compared to the relation of the previous report. This resulted in a metric that can be seen as a change in percentage points. As in the case for upgrades and downgrades as well as EPS estimates, only relations within 1 year were considered. Target price as an estimate was introduced late 2006, and thus only reports that had a target price estimate were considered.

A positive change in relation between target price and current share price was labelled as positive sentiment and a negative change was labelled as negative sentiment. However, as in the case for EPS estimates, a threshold was introduced. A positive relation can be seen as positive sentiment since the analyst see an upside in price and a negative relation can be seen as negative sentiment since the analyst see a decrease in price. One could argue that if this is the case, then only differences in target price could be used. However this could be problematic. Imagine a company presenting negative news and the price of the share drops. The analyst might reduce the target price, but he believes the market overreacted and he does not lower his target price as much as the price has dropped. This means that the analyst see a bigger upside and thus is more positive in his view to the company which should be labelled as positive sentiment. However, If one only studied the targeted price, this reaction would be interpreted as negative sentiment.

### 3.1.4 Cleaning the text data

An essential part of working with data in machine learning modelling concerns the cleaning of the data and especially when the data concerns text. The goal with cleaning the data is to attempt to maximize the information each word represent in a document. Thus, it is interesting to remove words or tokens that do not provide much valuable information. The cleaning of documents included removal of non-alphabetic characters such as symbols, numbers and punctuation as well as unnecessary white space. After the removal of symbols and numbers, individual characters were removed where a threshold of at least two characters were set to each word.

Another part of the data cleaning process consisted of removing words that directly could be tied to the labelling regimes. This means removing words such as 'upgrade' and 'downgrade' and the conjugations of such words. These words were removed since these single words could individually classify the whole text. While these words obviously contain semantic information, they also explicitly indicate what we are trying to classify. Thus, the algorithm might only pick out words instead of interpreting the sentiment of the document. The names used in the recommendation systems were also removed since they could indicate similar information as words directly tied to the labelling regimes. The words that were removed can be seen in equation 28, however, all conjugations are not explicitly shown.

$$\begin{aligned}
 & ['buy', 'hold', 'sell', 'reduce', 'outperform', 'underperform', \\
 & 'accumulate', 'upgrade', 'downgrade', 'restate', 'reiterate', \\
 & 'reaffirm', 'maintain', 'derate']
 \end{aligned} \tag{28}$$

## 3.2 Feature extraction

When the text had been cleaned and different labelling regimes had been set up, the last step before being able to use the text as input to a machine learning model is feature extraction. As discussed in section 2.5, feature extraction maps textual data into real valued interpretations. The feature extraction methods used were Bag of Words and tf-idf for the logistic regression classifier which will be used as a benchmark. The more sophisticated Word2vec algorithm that creates word embeddings for individual words will be used for the convolutional neural network structures. In the next two sections, the process of how these techniques were used will be presented.

### 3.2.1 Bag of Words and term frequency-inverse document frequency

The Bag of Words-model was implemented using the Python library Gensim [32]. A dictionary was created where each individual word across the documents used were given a unique index. The length of the dictionary is thus the number of unique words across the documents. Each individual word is then represented as a one hot encoded vector with all elements being zero except its unique index. The one hot encoded word vectors are then combined for each document and which creates document vectors. These vectors are very sparse since the length of the vectors is the size of the dictionary but only contain nonzero entries on the indexes of the words in the text.

As discussed in section 2.5.1, an extension to the Bag of Words-model is term frequency-inverse document frequency (tf-idf). The process was also implemented using the library Gensim and is similar to the Bag of Words-model but contain an extra step. When each word is represented by a one hot encoded vector, the 1 in the one hot encoded vector is replaced with a tf-idf score. The score is calculated using equation 26 presented in section 2.5.1. After this step, the document vectors are created as in the Bag of Words-model. Where the documents are represented by the counts of the tf-idf score of the words.

For both the Bag of Words-model as well as the tf-idf extension, word ngrams were also considered. This was done in a similar way except the dictionary was created with the grouping of the words given the size of the ngrams. This led to a larger size of the dictionary and thus size of the word and document vectors.

### 3.2.2 Word2vec

The Word2vec algorithm was implemented using Gensim and for this process, all cleaned documents were used and thus, disregarding the labelling of the documents. This can be done since Word2vec builds on a unsupervised neural network and we want to utilize as many words as possible to train the algorithm. When training the Word2vec algorithm several parameter and design choices need to be considered. First, the specific model needs to be specified whether to use the Continuous Bag of Words (CBOW) or Skip-gram approach. In the original paper by Mikolov and the Google Brain team, they showed that Skip-gram performed better on semantic tasks [29] and thus the Skip-gram model has been chosen. The Skip-gram model is also the model used for the pre-trained Word2vec implementation that the team made available with their work in [30].

The next step is to specify the dimensionality of the word vectors. In the original paper, Mikolov introduced a dimension size of 300, which also is the dimension used in their distributed model. Thus a dimension size of 300 has been used. For the model, a window size for how many history and feature words that is considered needs to be specified. In their paper, they used a context size of 5 and thus, this has been used in this thesis as well. One is also able to specify a minimum count for how many times a word needs to be included across all documents. In their original paper, this was specified to 5. However, since the amount of words for our data set is much fewer, the minimum count was set to 1 and thus all words were considered when training the Skip-gram model.

After the parameters and design choices had been made, the model was trained on the full data set of 67485 research reports with a total count of 14640419 words and 54322 unique words. Each word now has a 300 dimension representations. These representations

will be similar between words that occur in the same context and thus similar words will have similar representation. The similarity can be measured using cosine similarity and similarity results will be presented in section 4. After the word embeddings had been trained, the representations could be used in the input layer to a machine learning model. This will further be discussed in section 3.3.2.

### 3.3 Machine learning development

With the text converted to numerical representations using the feature extraction methods, the next step was to classify the sentiment in the documents using machine learning models. Logistic regression was implemented with the goal of being a simple method working as a benchmark that could be compared to a more sophisticated neural network model. A convolutional neural network was implemented where different architectures were tested. The steps taken to create the models will be presented in the next two sections.

#### 3.3.1 Logistic regression model

The logistic regression model was implemented using the Python library Scikit-learn [33]. Models were trained for the three separate labelling regimes defined using the document vectors from the Bag of Words-model as well as the tf-idf extension. For each different labelling scheme, the size of the input was equal to the size of the amount of unique words in each scheme. This was 18744 for upgrades and downgrades, 20421 for changes in EPS estimates and 28807 for changes in the relation between target price and current share price. For each set, an amount of weights equal to the amount of words were trained.

Ngrams of size 1, 2, 3, 4 and 5 were tested. The default optimization method for logistic regression in the Scikit-learn library was chosen.

#### 3.3.2 Convolutional Neural Network architectures

The convolutional neural network models were implemented using the Python library Keras [34] utilizing Tensorflow [35] as backend. Two separate models were implemented inspired by the work of Kim Yoon in [5]. The differences between the models are that one model uses a single input channel whereas the other model uses a double input channel concept. Besides this, the models are similar in structure, utilizing several parallel filters of different sizes.

The first step of a convolutional neural network model is the input layer. To be able to use the Word2vec model, a dictionary was created where each individual word in the data set was given a unique index. The unique index corresponds to a row number, where the row represents a word vector with 300 elements. This means that an embedding matrix will be created with the dimensions *Number of words*  $\times$  300. Words that were not represented in the Word2vec model was randomly initialized by uniformly sampling a 300 dimensional vector,  $U[-a, a]$ , where  $a$  was chosen to be 0.25 to achieve similar variance as the pre-trained vectors. The next step was to specify the input size, since the input size need to be the same for all documents. Thus, the lengths of the documents were inspected and a threshold of 400 words was set. This was chosen since most documents were around 350 words long and an upper threshold of 400 only filtered a few documents while still being computationally stable. One could have chosen the input size to be equal to the

maximum length of the documents. However, decreasing the training time was deemed more important than keeping all documents, thus the documents above 400 words were removed.

The documents were converted into sequences of the unique indexes corresponding to the words in the dictionary. These vectors were used as the input to the convolutional neural network. Documents that were shorter than 400 words were zero padded at the end, where index 0 represents a 300 dimensional zero vector, indicating no word.

The inputs will thus be unique index integers, that map the words to word embeddings. For the model that utilized a single input channel, the pre-trained Word2vec model by Google as well as the Word2vec model trained on the research reports were tested. The word embeddings can be seen as any other weight in the network and can thus be trained. Keeping a static embedding matrix as well as a dynamic matrix, with trainable weights, were considered. For the model that utilized a double input channel concept, the text input was duplicated, and the copies were considered for the two channels. One channel used the pre-trained Word2vec embeddings from Google and the other the Word2vec model trained on the research reports. Each word can now be thought of as having a 600 dimension embedding combined from two 300 dimensional embeddings, where each embedding originates from a different source. However, the embeddings are separate and divided into two channels. This means that each input document will be a 400x300 matrix. Static and dynamic embedding matrices were tested as well as keeping one static and one dynamic. The reasoning behind testing a double input channel is to utilize both of the Word2vec models at the same time, in the hopes that they will complement each other in areas where the word embeddings differ.

After the input layer had been constructed, the next step was to establish the convolutional layers. In the convolutional neural networks, the filters will slide over the words. The words are represented in 300 dimensional embeddings and thus the filters will slide over full rows of matrices. What also needs to be specified for the filters is the region size, which specifies how many words the filter will slide over at the same time. The region specification can be seen as an ngram detector, was a region size of 2 will consider two words simultaneously. This is similar to an ngram of size two. The convolutional neural networks will consist of several parallel convolutional layers with different region specifications. Different setups of parallel convolutional layers were tested, but ultimately 3 parallel convolutional layers were chosen where different region sizes were evaluated. For the convolutional layers, ReLU activation function was chosen and a stride of 1 was used. Padding to the convolutions was applied when necessary and was used to consider the beginning and ending words equally many times as the other words. As in [5], regularization was tested by adding penalization of large parameter values, where the  $l_2$ -norm constraint was set to 3 which led to an improvement in performance. Using a  $l_2$ -norm constraint was shown in [36] to have relatively little effect on performance, and could for some data sets even decrease performance. However, imposing a  $l_2$ -norm constraint increased performance for the tested models.

To each filter, global max pooling was applied, with the goal of extracting the feature with the highest value which corresponds to the most important feature. This also deals with different document lengths, since only the most important feature is considered. The amount of filters for each height specification was tested and the number of filters which yielded the best validation accuracy was chosen.

These features were concatenated and subsequently dropout was applied where several dropout rates were tested. It was experimented by adding an additional fully connected hidden layer with 256 units using ReLU as activation function. However, this structure did not increase performance in terms of validation accuracy. It also made the model more complex resulting in more parameters and a longer training time. Thus, an extra fully connected layer was not considered. The network was finished with a fully connected layer with 2 hidden units using a Softmax activation function to classify the data into positive and negative sentiment. The optimization method chosen was the Adam optimizer, where the learning rate was specified differently depending on the labelling regime. Finally, the loss function used was the cross entropy loss discussed in section 2.3.2.

An illustration and a simplification on how the single channel convolutional neural network structure works can be seen in figure 6. In the example, word representations with a dimension of 5 has been used. An input document consisting of "This analyst was negative in his analysis" has been used, thus creating a input matrix of the size 7x5. However, as discussed earlier, the input for the networks used in the thesis will be of the size 400x300. 3 region sizes of (2,3,4) with 2 filters for each region are shown that will be slid across the input matrix. Dropout has not been visualized in the schematic. The double input channel structure will be different in the way that the input matrix will consist of two matrices in parallel where the word representations originate from different sources.

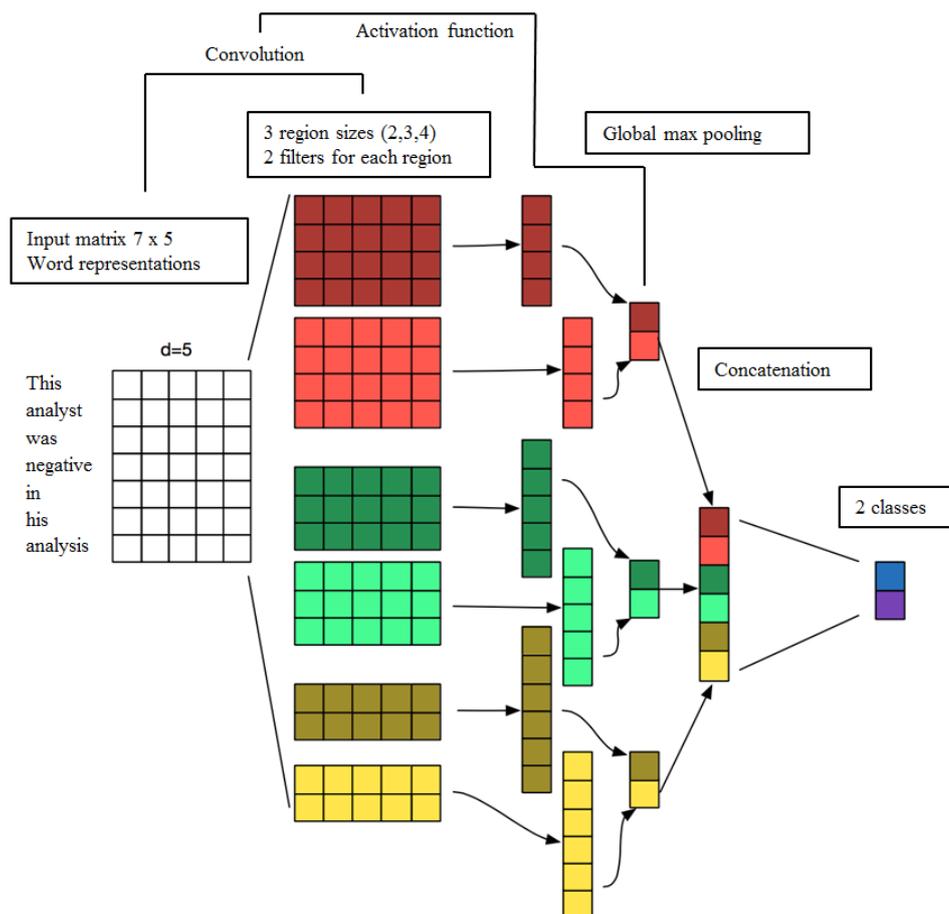


Figure 6: Schematic on how the single input channel structure works. Illustration taken from [36], however the illustration has been modified to fit the scope of the thesis.

The hyperparameters for the convolutional neural network structures that have been discussed in this section have been tested by a combination of trial and error and a grid search approach, where sets of hyperparameter values has been tested. The ranges have been chosen based on inspiration from previous work in [5] and the thorough testing of hyperparameters in [36]. The values ultimately chosen has been based on the best performing setup with respect to validation accuracy and will be different depending on the network structure and labelling regime. How the models have been trained and tested will further be presented in section 3.4.

### 3.4 Training and testing the models

The machine learning models were trained and tested on the three different labelling regimes specified. Equally many positive and negative documents were considered to achieve perfect class balance. This is important since the model may become biased towards the class containing more documents. This was accomplished by sampling a subset of documents from the class that had the most documents in each labelling regime. For upgrades and downgrades of recommendation, a total of 2776 negative and 2665 positive documents existed and thus 2665 documents from each class were chosen. For changes in EPS estimates, different threshold values were tested but it was ultimately set to 0.01 which yielded 3543 positive documents and 5290 negative documents and thus 3543 documents from each class were chosen. For the relation between target price and current share price, different threshold values were considered but ultimately, a value of 0.05 were chosen. This yielded 7973 positive documents and 7963 negative documents, and thus 7963 documents were considered for each class. The threshold values were chosen with regards to performance and keeping the set as large as possible but also based on what is reasonable from a financial perspective.

The next step was to divide the documents of the labelling regimes into training and test sets. For the convolutional neural networks, the documents were split into three sets, a training set and validation set used when training the model and a test set to test the model. The distributions were 60% for the training set, 20% for the validation set and 20% for the test set. The validation set was used to tune the parameters of the model to achieve as good accuracy as possible while preventing overfitting. The number of epochs needed for the network accuracies to stabilize varied and thus different number of training epochs was used for the different labelling regimes. After the training was done, the weights of the epoch that gained the best prediction accuracy on the validation set were saved. When testing the networks, the weights from the best epoch were used. A batch size of 50 was used for the convolutional neural networks.

For the logistic regression model the documents were also divided into training and test sets with 80 % for the training set and 20 % for the test set. The training was done using K-fold cross-validation. This means that the training set was divided into  $K$  folds. A model was trained  $K$  times, where each time a different fold was used as validation set and all other folds as training set. This results in  $K$  validation accuracy estimates and the cross-validation estimate is computed by averaging the  $K$  number of validation accuracies. The K-fold cross validation process was done using  $K = 10$ . K-fold cross validation was not used for the convolutional neural network architectures due to computational and time complexity. After cross validation had been done, the best parameters were chosen with respect to cross validation accuracy. Subsequently, a model was trained utilizing the full training set for training and then testing was performed on the test set.

### **3.5 Summary of thesis methodology**

In figure 7, a workflow of the methodology used in the thesis is presented. The figure explains the most important steps taken and highlights the main methods used in the necessary order. The top part of the figure explains the steps taken in 3.1 which includes the pre-processing of data, the cleaning of the text as well as the division of sentiment labels. The next step taken is the feature extraction process that was explained in section 3.2. This leads to the development of the machine learning models that was explained in section 3.3. The models developed are ultimately trained and tested, this was explained in section 3.4.

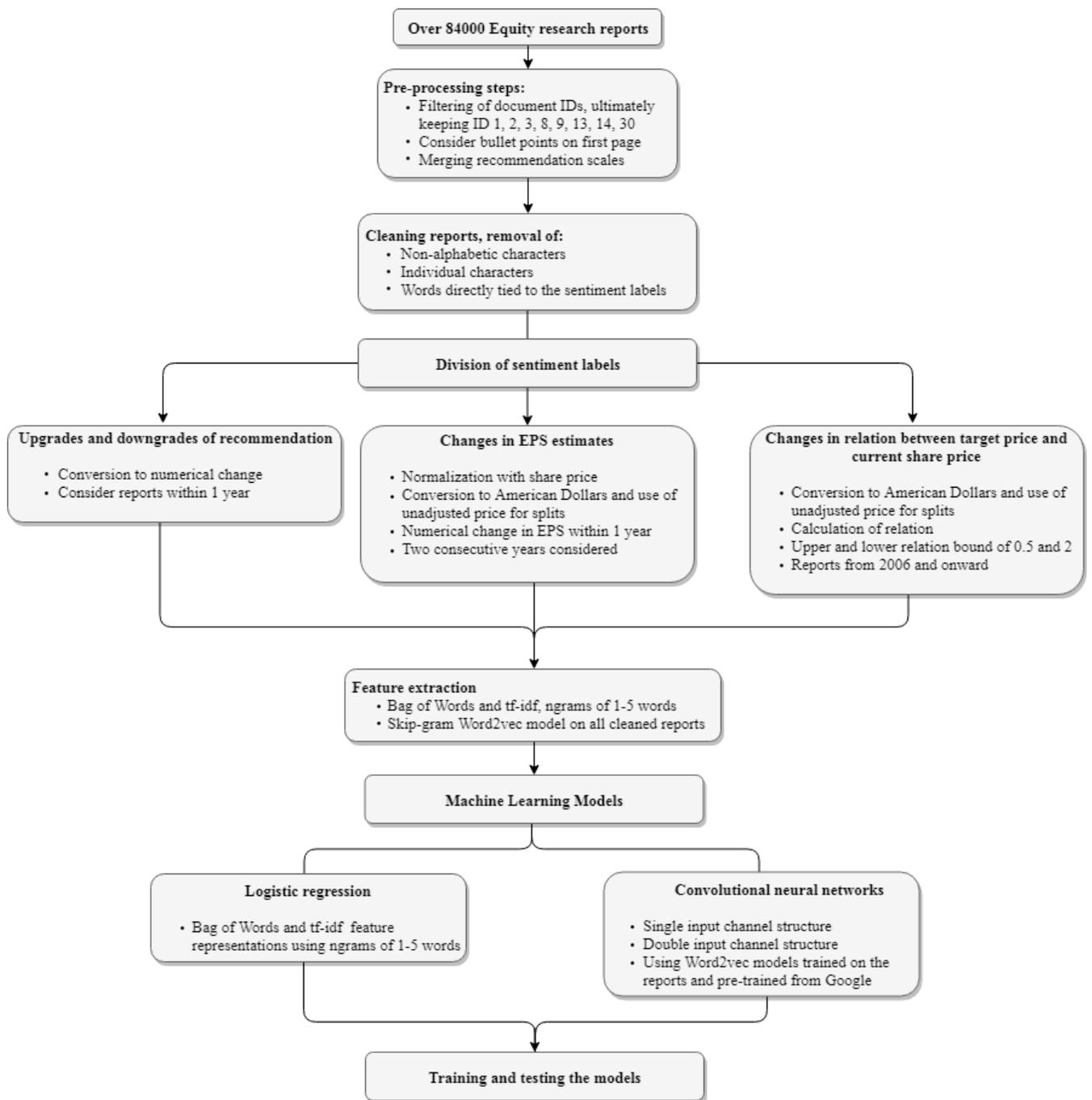


Figure 7: Workflow of the methodology used in the thesis.

## 4 Results

In this section, word embedding results from the trained Skip-gram model will be presented as well as compared to a model trained by the Google Brain team. Furthermore, accuracy results from the logistic regression model will be presented for the three labelling regimes. Lastly, accuracy results from the two proposed convolutional neural network architectures will be presented for the tested labelling regimes.

### 4.1 Word2vec

The result of the top five most similar words and their similarity score for six different words can be seen in Table 2. Similarity results for the Word2vec model trained by the Google Brain team can be seen in Table 3. The underscore present for some of the words in the Word2vec model trained by the Google Brain team indicate a space and thus separate words. The similarity is measured using cosine similarity. The first four words are chosen to highlight words that are tied to sentiment and the results are fairly similar. The last two words are two companies that are present in the research reports. In the context of this thesis, Avanza is a Swedish brokerage firm and Investor is a Swedish investment company. For these words, we see a big difference in similarity results. For the Word2vec model trained on the research reports, the most similar word to Avanza is Nordnet, which is Avanzas biggest competitor. The other similar words are Nordic Banks. However, the most similar words for the Word2vec model trained by the Google Brain team is the car Toyota Avanza. Similar behaviours can be seen for the word Investor. In the model trained on the research reports, the top two most similar words are other Swedish investment companies. Two of the other words, EQT and Mölnlycke, are companies owned by Investor. However, in the model trained by the Google Brain team, the most similar words to Investor are words related to an investor. Thus, these words have different similarity words between the Word2vec models. This is due to our data set being topic specific and thus it is interesting to compare the two models as word embedding inputs to the machine learning models.

As can be observed in Table 2 and Table 3, the word embeddings trained by the Google Brain team have an overall higher similarity score for the first 4 words than the word embeddings trained on the research reports. This is mainly due to that the fact that the data set that the Google Brain team used is far superior in terms of unique as well as total words. Some unintuitive examples in similarity between words can also be found in both of the two Word2vec models. This is because the models are based on words that occur in similar contexts and do not strictly build on similarity. This can be observed in Table 2 and Table 3, where *increase* is similar to *decrease* even though the words are antonyms.

Table 2: Most similar words for the Word2vec model trained on the research reports.

<b>Word</b>	<b>Most Similar Words</b>	<b>Similarity</b>
good	1. excellent	0.6589
	2. solid	0.6578
	3. decent	0.6136
	4. strong	0.6106
	5. healthy	0.5927
bad	1. Bad	0.4913
	2. winters	0.4847
	3. alarming	0.4777
	4. catastrophic	0.4707
	5. overstocking	0.4566
increase	1. rise	0.6968
	2. decrease	0.6702
	3. increased	0.6383
	4. decline	0.6125
	5. fall	0.5953
decrease	1. decline	0.7047
	2. decreases	0.6702
	3. fall	0.6356
	4. rise	0.6320
	5. moderation	0.5905
Avanza	1. Nordnet	0.7090
	2. Collector	0.6261
	3. SHB	0.6166
	4. Swedbank	0.5948
	5. Jyske	0.5590
Investor	1. Industrivärden	0.6371
	2. Kinnevik	0.6111
	3. IGC	0.5948
	4. EQT	0.5772
	5. Mölnlycke	0.5681

Table 3: Most similar words for the pre-trained Word2vec model by the Google Brain team.

<b>Word</b>	<b>Most Similar Words</b>	<b>Similarity</b>
good	1. great	0.7292
	2. bad	0.7190
	3. terrific	0.6889
	4. decent	0.6837
	5. nice	0.6836
bad	1. good	0.7190
	2. terrible	0.6829
	3. horrible	0.6703
	4. Bad	0.6699
	5. lousy	0.6648
increase	1. decrease	0.8370
	2. increases	0.7709
	3. increased	0.7578
	4. reduction	0.6908
	5. increasing	0.6872
decrease	1. increase	0.8370
	2. decreases	0.8094
	3. decreased	0.7642
	4. reduction	0.7175
	5. increased	0.7083
Avanza	1. Toyota_Avanza	0.5013
	2. Supermercados	0.4761
	3. Atoz	0.4674
	4. Grand_Livina	0.4617
	5. Toyota_Fortuner	0.4557
Investor	1. investor	0.6306
	2. Shareholder	0.5811
	3. Investors	0.5658
	4. investors	0.5505
	5. Small_Cap	0.5454

## 4.2 Logistic regression

The results from running the logistic regression model on the three labelling regimes can be seen in Table 4, 5 and 6. The Tables show 10-fold cross validation accuracies for the Bag of Words and the term frequency-inverse document frequency (tf-idf) feature specifications using 1,2,3,4 and 5-grams. The best accuracies for the corresponding ngram and feature representation method can be seen highlighted in bold. The testing accuracies for the best performing ngram and feature representation method for each specific labelling regime can be seen in Table 7.

Using upgrades and downgrades as labelling regime, the best accuracy was achieved using 2-grams for both the Bag of Words and tf-idf method which was 77.46 % for Bag of Words and 77.84 % for tf-idf. Tf-idf generally performed better than the Bag of Words approach expect for 1-grams. However, the simple Bag of Words was still performing well with

only being a few percentages units worse than the tf-idf approach for the ngrams where it showed worse performance.

Changes in EPS estimates and changes in the relation between target price and current share price yields a considerably lower accuracy than upgrades and downgrades. The best accuracy for changes in EPS estimates was achieved using 4-grams yielding an accuracy of 58.34 % for the Bag of Words approach and 57.73 % for tf-idf. The best accuracy for changes in the relation between target price and current share price for Bag of Words was achieved using 4-grams and yielded an accuracy of 57.14 % and 3-grams for tf-idf which yielded an accuracy of 56.87 %.

In Table 7, the results on the test set for the best ngram and feature representation method can be seen. One can observe that the model has generalized well with respect to validation and testing accuracies, where the accuracies deviate less than 1 percent across all the labelling regimes.

Table 4: Classification accuracies using logistic regression for the Bag of Words and tf-idf approach for different ngrams using upgrades and downgrades as labelling regime.

<b>Upgrades and Downgrades of recommendation</b>		
	Bag of Words	Tf-idf
1-grams	72.33 %	70.33 %
2-grams	<b>77.46 %</b>	<b>77.84 %</b>
3-grams	76.50 %	77.48 %
4-grams	74.04 %	75.35 %
5-grams	69.79 %	71.06 %

Table 5: Classification accuracies using logistic regression for the Bag of Words and tf-idf approach for different ngrams using change in EPS estimates as labelling regime.

<b>Changes in EPS estimates</b>		
	Bag of Words	Tf-idf
1-grams	53.58 %	52.56 %
2-grams	55.31 %	55.95 %
3-grams	56.78 %	56.74 %
4-grams	<b>58.34 %</b>	<b>57.73 %</b>
5-grams	56.74 %	56.65 %

Table 6: Classification accuracies using logistic regression for the Bag of Words and tf-idf approach for different ngrams using the relation between target price and current share price as labelling regime.

<b>Changes in relation between target price and current share price</b>		
	Bag of Words	Tf-idf
1-grams	52.68 %	52.28 %
2-grams	55.45 %	55.17 %
3-grams	57.03 %	<b>56.87 %</b>
4-grams	<b>57.14 %</b>	56.60 %
5-grams	56.56 %	56.15 %

Table 7: Testing accuracies using logistic regression for the best performing ngram and feature representation method for each tested labelling regime.

<b>Labelling regime</b>	<b>Feature representation method</b>	<b>Testing accuracy</b>
Upgrades and Downgrades	Tf-idf 2-grams	77.84 %
EPS estimates	BoW 4-grams	58.96 %
Relation between target price and current share price	BoW 4-grams	56.41 %

### 4.3 Convolutional Neural Network architectures

In the next three sections the results for the single and double channel convolutional neural network architectures will be presented for the three tested labelling regimes. The tuned hyperparameters will be specified and the training and validation accuracies throughout training will be presented. The testing accuracies for the weights yielding the highest validation accuracy will also be presented. For the single input channel structure, the Word2vec embeddings trained by the Google Brain team as well as embeddings trained on the research reports were tested. However, the Word2vec embeddings trained on the research reports showed superior results across all the labelling regimes. Investigations of whether to use trainable or non-trainable input layers were also made. A non-trainable input layer for the single input channel structure and trainable input layers for the double input channel structure showed superior results in terms of validation accuracy compared to other combinations.

#### 4.3.1 Upgrades and Downgrades of recommendation

For upgrades and downgrades of recommendation, the final version of the single input channel neural network structure used region sizes of (3,4,5) in the parallel convolutional layers and for each region, 175 filters gave the best results. A learning rate of 0.001 and a dropout probability of 0.6 were used. In Figure 8, the training and validation accuracy can be seen where the training accuracy starts to converge towards 1 after 9 epochs and the validation accuracy stabilizes around 80 % after 6 epochs. The highest percent in validation accuracy achieved was 81.24 % and these weights were thus saved to use for testing on the test set. Using these weights, a testing accuracy of 81.05 % was reported.

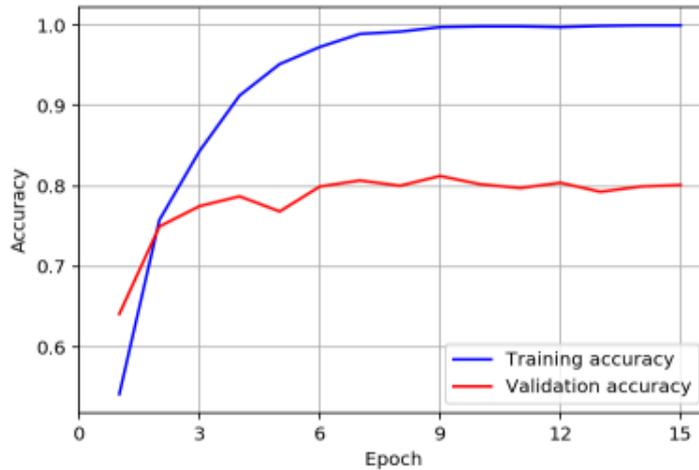


Figure 8: Training and validation accuracies for the single input channel structure using upgrades and downgrades as labelling regime.

The final version for the architecture using a double input channel structure used a similar set of hyperparameters as in the single input channel structure. Thus, a region size of (3,4,5), 175 filters and a learning rate of 0.001 was used. However, higher validation accuracy was achieved for the network using a dropout probability of 0.5 instead of 0.6. The training and validation accuracies can be seen in Figure 9. After roughly 7 epochs, the training accuracy stabilizes close to 1. The validation accuracy converges to an accuracy of roughly 82 % and the highest validation accuracy achieved was 82.83 %. This resulted in a testing accuracy of 83.40 % using the same weights as for the best reported validation accuracy.

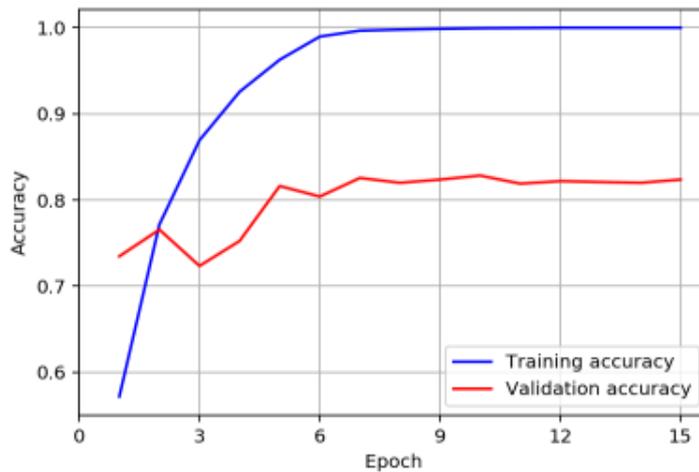


Figure 9: Training and validation accuracies for the double input channel structure using upgrades and downgrades as labelling regime.

The results from Figure 8 and Figure 9 are summarized in Table 8. The best training and validation accuracy respectively are highlighted in bold. The best validation and testing accuracy were achieved for the double input channel structure.

Table 8: The best validation accuracy and the testing accuracy for those network weights for the tested network structures.

Upgrades and Downgrades of recommendation		
Network structure	Validation accuracy	Testing accuracy
Single channel	81.24 %	81.05 %
Double Channel	<b>82.83 %</b>	<b>83.40 %</b>

### 4.3.2 Changes in EPS estimates

In the case when investigating the change in EPS estimates a similar set of hyperparameters were used as in the case for upgrades and downgrades of recommendation. The structure using a single input channel network structure used region sizes of (3,4,5) and 175 filters for each region. For the network, a dropout probability of 0.5 was used. However, a learning rate of 0.0001 was used. Using a non-trainable input layer decreased the rate as of how fast the training accuracy increased, but ultimately led to an increase in validation accuracy. Thus, the network was trained for 50 epochs which is considerably longer than the single input channel structure used for upgrades and downgrades. The results from training the network can be seen in Figure 10. After roughly 50 epochs the training accuracy stabilized around 1 and the validation accuracy converged to roughly 0.56-0.57. The maximum validation accuracy achieved was 57.59 % and using these weights the network achieved a testing accuracy of 55.92 % on the test set.

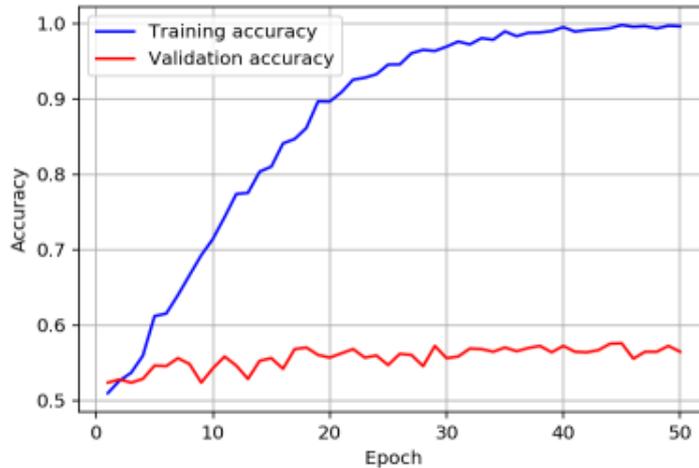


Figure 10: Training and validation accuracies for the single input channel structure using changes in EPS estimates as labelling regime.

The network utilizing a double channel input structure used a similar set of hyperparameters with region sizes of (3,4,5), dropout probability of 0.5 and a learning rate of 0.0001. However, using 225 filters for each region as opposed to 175, yielded a higher validation accuracy. The network required less epochs for the training accuracy to converge and thus only 20 were needed. The results for the training and validation accuracies can be seen in Figure 11. The training accuracy converged to 1 and as for the single input channel, the accuracy stabilized at roughly 0.56-0.57 with the highest validation accuracy reaching 57.66 %. Using these weights, a testing accuracy of 56.06 % was achieved on the test set.

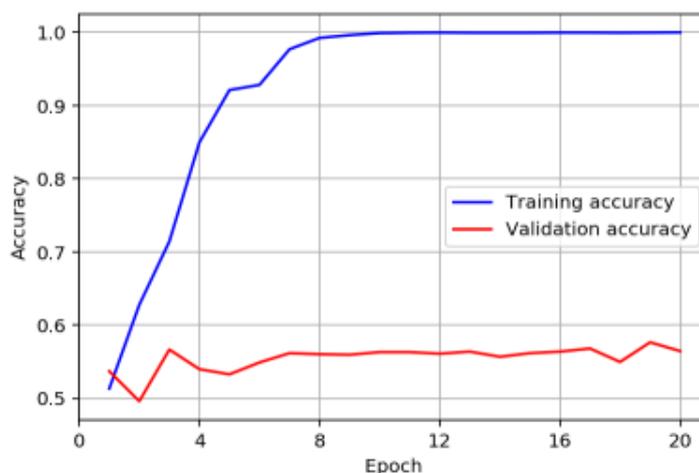


Figure 11: Training and validation accuracies for the double input channel structure using changes in EPS estimates as labelling regime.

The results from Figure 10 and Figure 11 are summarized in Table 9 where the best validation and testing accuracies are reported. The accuracies for the best performing structures are highlighted in bold. As in the case for upgrades and downgrades of recommendation, the double input channel yielded the best accuracies on the validation and test set. However for this labelling regime, the accuracy differences between the structures are minimal. As in the case for logistic regression, the validation and testing accuracies were considerably lower than using upgrades and downgrades of recommendation.

Table 9: The best validation accuracy and the testing accuracy for those network weights for the tested network structures.

Changes in EPS estimates		
Network structure	Validation accuracy	Testing accuracy
Single channel	57.59 %	55.92 %
Double Channel	<b>57.66 %</b>	<b>56.06 %</b>

### 4.3.3 Changes in relation between target price and current share price

When considering changes in relation between target price and current share price as labelling regime, similar hyperparameters as used for the other labelling regimes showed the best performance in terms of validation accuracy. The single input channel structure used region sizes of (3,4,5) with 175 filters, a dropout probability of 0.5 and as for changes in EPS estimates, a learning rate of 0.0001. Similar behaviour was found as in the case for changes in EPS estimates regarding the number of epochs and thus the network was trained for 50 epochs. Results from the training of the network can be seen in Figure 12, where the training accuracy stabilizes at around 1 after 50 epochs. The highest validation accuracy achieved was 56.80 % and using the same weights for the network when testing on the test set gave a testing accuracy of 54.77 %.

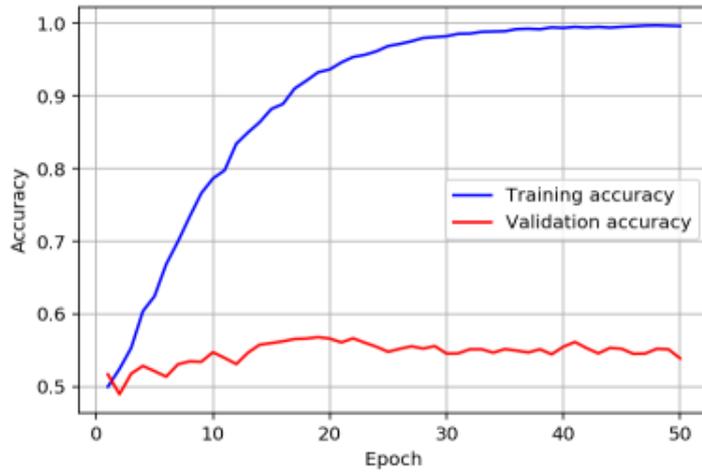


Figure 12: Training and validation accuracies for the single input channel structure using changes in relation between target price and current share price as labelling regime.

The network structure using a double input channel used a similar set of hyperparameters with region sizes of (3,4,5), a dropout probability of 0.5 and a learning rate of 0.0001. However, using 225 filters as opposed to 175 increased the performance in terms of validation accuracy. Similarly as in the case for changes in EPS estimates, an increase in filters for the double input channel structure increased the validation accuracy. The results from the training of the network can be seen in Figure 13, where the training accuracy stabilizes at around 1 and the validation accuracy converges to roughly 0.56-0.57. The maximum validation accuracy reached was 57.83 % and using the same weights on the testing set yielded a testing accuracy of 56.15 %.

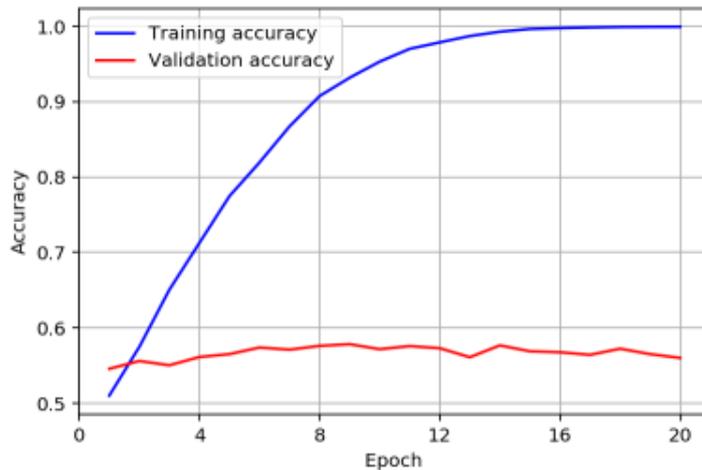


Figure 13: Training and validation accuracies for the double input channel structure using changes in relation between target price and current share price as labelling regime.

The results from Figure 12 and Figure 13 can be seen summarized in Table 10, where the highest validation accuracies and the corresponding testing accuracies are shown. The best performing metrics are highlighted in bold. As for both upgrade and downgrade of recommendation as well as changes in EPS estimates, the double input channel structure outperforms the single channel with respect to both validation and testing accuracy.

However, as logistic regression also showed, the convolutional neural network architectures show considerably lower accuracy compared to using upgrades and downgrades as labelling regime.

Table 10: The best validation accuracy and the testing accuracy for those network weights for the tested network structures.

<b>Changes in relation between target price and current share price</b>		
Network structure	Validation accuracy	Testing accuracy
Single channel	56.80 %	54.77 %
Double Channel	<b>57.83 %</b>	<b>56.15 %</b>

## 5 Discussion

In this section, the results from logistic regression and the convolutional neural network architectures will be compared and discussed. This is followed by a segment that will discuss improvements that could be made to the models and what could be tested in the future. Finally an idea of how the results could be used when predicting changes in recommendations will be presented.

### 5.1 Evaluation of the results

The best results in terms of validation and test accuracy for each labelling regime as presented in section 4 can be observed in Table 11. For upgrades and downgrades of recommendation, the best performing structure was the double input channel convolutional neural network (CNN). For changes in EPS estimates, the Bag of Words (BoW) approach yielded the best results in terms of both validation and testing accuracy. When considering the relation between target price and current share price, the double input channel convolutional neural network performed the best in terms of validation accuracy but the Bag of Words approach yielded the best result in terms of testing accuracy.

Table 11: The best validation accuracy and the testing accuracy for the best performing model for each labelling regime.

Labelling regime	Type of model	Validation accuracy	Testing accuracy
Upgrades and Downgrades	Double channel CNN	<b>82.83</b> %	<b>83.40</b> %
EPS estimates	BoW 4-grams	<b>58.34</b> %	<b>58.96</b> %
Relation between target price and current share price	Double Channel CNN	<b>57.83</b> %	56.15 %
	BoW 4-grams	57.14 %	<b>56.41</b> %

One out of the two tasks of the project consisted of investigating different labelling regimes of research reports. Out of the three investigated regimes, upgrades and downgrades of recommendations clearly showed superior results as can be seen in Table 11. The other regimes showed substantially worse performance, which indicate that these labelling regimes cannot represent any pattern across the documents. However, all tested machine learning models for both of the labelling regimes showed an accuracy of over 50 %. This means that the models have learned some kind of relationship between the documents and performs better than a classifier predicting only one of the labels. Still, the fact remains that upgrades and downgrades of recommendation performed much better and thus was the regime that was best able to divide research reports into two separate groups.

Changes in EPS estimates and relation between target price and current share price show some ability of classifying research reports. However, most of the emphasis when evaluating how well the machine learning methods are performing should be put on upgrades and downgrades. This is due to the fact that the machine learning models are most likely not performing poorly because of their design. The reason why the regimes perform poorly is most likely due to how the division and the assumptions that goes with them has been made, i.e. the fact that the regimes represent close to no pattern between the classes. The threshold introduced have been optimised to keep as many documents as possible and to increase validation accuracy but the reasoning behind the choices is quite vague. What actually is financially reasonable might not be good enough of a motivation. Thus, a more

thorough investigation of the thresholds and how the models behave would be advised. Investigating more into what actually causes the data errors that have been found would also most likely increase the classification accuracies for those labelling regimes. However, the regimes should not directly be disregarded if one was to use the classifiers to predict recommendation changes.

The second task consisted of developing machine learning models to determine sentiment in research reports. Specifically a logistic regression model and convolutional neural network structures. Comparing the results between the two models, the results were mixed, where the convolutional neural network structure performed better for some labelling regimes whereas logistic regression performed better for other regimes. Thus, one cannot clearly say that a more complex model in the form of a convolutional neural network always will be better. However, as discussed earlier, more emphasis should be put on the labelling regime upgrades and downgrades of recommendations, since it is the only labelling regime that shows a promising prediction accuracy. When considering this regime, the double input channel convolutional neural network performs roughly 5 % better. This suggests that a more complex model can perform better than a simple one. However, since logistic regression still perform well, it should not completely be ruled out if one was to try and use the classifier for predicting recommendation changes.

Across all the labelling regimes for the convolutional neural network structures, the double input channel structure performs better in terms of validation and testing accuracy. Thus, the idea of having the words being represented by two embeddings trained on separate data sets seems to be successful. This might be explained by the similarity scores presented in Table 2 and Table 3 in section 4. The words presented that are tied to sentiment, which can also be seen as words that are more common and frequent across all types of documents, have a higher similarity score in the Word2vec model trained by the Google Brain team than the model trained on the research reports. Thus, the embeddings trained by the Google Brain team seems to better represents such words. However, the Word2vec model trained by the Google Brain team does a poor job at representing words that are topic specific to our data set. This can be seen in Table 2 and Table 3 as similar company names does not show high similarity. However, the Word2vec model trained on the research reports show a high similarity between such words. Thus, by combining two different Word2vec models, where one is better at representing common words, and one at representing topic specific words, we have in this thesis been able to achieve an improvement in validation accuracy. This suggests that if one has a small data set, one can use two Word2vec models where the one trained by the Google Brain team one will provide a good embedding overall for words that are common, and a Word2vec which is trained on a smaller topic specific data set will provide good embeddings for topic specific words. However, this would further need to be tested to be able to conclude it for a more general case.

When training the convolutional neural network structures, the training accuracy reaches 1 across all labelling regimes. This might indicate that the methods are overfitting. However, much emphasis has been put in trying to avoid overfitting by investigating different regularization techniques such as dropout and restricting network weight sizes. For example, increasing the dropout probability past 0.6 had no improvement in performance. However, one can also observe that for both logistic regression and the convolutional neural network structures, the models generalize well between the validation accuracy and test accuracy.

## 5.2 Future work

Tuning of the hyperparameters for the convolutional neural networks has been done. This investigation showed that for the double input channel structure, it was advisable to use more filters than for the single input channel structure. Several region sizes has also been tested, but ultimately region sizes of (3,4,5) performed the best for all structures and labelling regimes. However, the tuning of the hyperparameters has been done using only one training and validation set due to the time complexity. Further testing utilizing 10-fold cross validation would be interesting to perform to be more certain as to which hyperparameters perform the best. The tuning was done using a combination of trial and error and a grid search approach, where values were tested in certain ranges that showed promising performance. However, a more sophisticated approach called Bayesian optimization could be used for tuning the hyperparameters. Bayesian optimization trains the network using different hyperparameters, which generates a model function for each tested set of hyperparameters. These can be used to predict the optimal setup across the entire domain [37].

In this thesis, convolutional neural networks have been used to determine the sentiment in research reports. An extension to the already used structures would be to include a Long Short-Term Memory (LSTM) layer after the convolutions as done in [38] and [39]. LSTM is a type of recurrent neural network and in these types of networks, the output is fed back into the network. Such networks perform well when dealing with sequential data and it would thus be interesting to investigate if that would lead to an increase in performance.

A restriction that was made throughout the thesis was to only consider the first page of the research reports. This was done with the motivation that the first page represents a summary of the document and to reduce the document size and thus the time and computational complexity. This also avoided the problem with extracting the text from the PDF's. If one was to solve the extraction problem and time was not an issue, it would be interesting to use the full documents for training. It would also be interesting to run the already created classifiers on the full documents to see how they perform. An advantage of using the full research reports would also be that it would lead to more text and thus more words overall which would be beneficial when training the Word2vec algorithms.

## 5.3 Predicting recommendation changes

Previously in this section, the concept of using the classifiers to predict changes in recommendation has briefly been touched upon. But how would such a process actually work? Investigating such a model is out of scope of this thesis. However, it would be interesting to theorize on how such a concept could work.

If one were to predict changes in recommendations using sentiment analysis, one would test if there exist a relationship between changes in recommendation and changes in sentiment. This test could be done by running the classifier on new unseen documents and if they are classified as positive one could expect an upgrade and if they are classified negative, one could expect a downgrade. However, if the document already has a recommendation of Buy, no upgrade of recommendation can be made. Thus, a buy document would only be considered if it was negatively classified. This holds true but in the opposite direction for documents that has a Sell recommendation, these reports would only be considered if they were positively classified. However, if a report had a Hold recommendation, they could be

both upgraded and downgraded and thus, such reports would be the most interesting to investigate.

However, to expect that each report will either lead to an upgrade or downgrade would be an incorrect belief since that is not the case. Thus it would be more interesting to investigate the classification probabilities. In the usual case for a binomial classifier, a prediction is based on if the classification probability exceeds 50 %. However, one could propose a new threshold that if a document exceeds a classification probability of X %, the document might indicate that an upgrade or downgrade will happen. Investigations would need to be made in order to determine this threshold, but as a starting point one could test 95 %, and expect all documents above this threshold to indicate an upgrade or downgrade. This number is based on the fact that the data set consisted of roughly 5% upgrades and 5% downgrades respectively.

## 6 Conclusions

In this thesis, machine learning has been used to determine sentiment in research reports written by financial equity analysts from the equity research department at SEB. The reports contain no explicit sentiment label and thus three labelling regimes have been constructed. Logistic regression as well as several convolutional neural network structures has been tested on these regimes. The regime upgrades and downgrades of recommendation showed the most promising results at dividing the documents into positive and negative classes. Logistic regression yielded a final testing accuracy of 77.84 % and a double input channel convolutional neural network yielded a testing accuracy of 83.60 %. Changes in EPS estimates as well as changes in the relation between target price and current share price performed significantly worse and did not achieve accuracies over 60 %. For these regimes, logistic regression performed equally well or better. The results across the three labelling regimes show no significant improvement in using a convolutional neural network structure over a logistic regression model. However, if one were to only consider upgrades and downgrades, which is the only regime that shows promising results, a convolutional neural network structure performs noticeably better than a logistic regression model.

Across all the labelling regimes, the double input channel convolutional neural network performed better than the single input channel structure. The double input channel structure utilized two separate Word2vec models, one trained by the Google Brain team and the other one on the research reports. The results suggest that the two models complement each other and an increase in performance can be achieved by using a combination of word embeddings.

If one were to use the constructed classifiers for prediction of changes in recommendations, one could classify unseen reports but introduce a classification threshold. If the reports were classified as positive or negative with a certainty above the threshold one could expect a change in recommendation. If further tests were to be made, a focus should lie on using upgrades and downgrades of recommendation as labelling regime.

## References

- [1] Zhao F. Natural Language Processing – Part I: Primer. S&P Global Market intelligence, 2017.
- [2] Mäntylä MV, Graziotin D, Kuutila M. The Evolution of Sentiment Analysis - A Review of Research Topics, Venus, and Top Cited Papers. arXiv:1612.01556. 2018.
- [3] Medhat W, Hassan A, Korashy H. Sentiment analysis algorithms and applications: A survey. Ain Shams Engineering Journal. 2014;5:1093-1113
- [4] Collobert R, Weston J, Bottou L, Karlen M, Kavukcuogly K, Kuksa P. Natural language processing (almost) from scratch. The Journal of Machine Learning Research. 2011;12:2493-2537.
- [5] Kim Y. Convolutional Neural Networks for Sentence Classification. arXiv:1408.5882. 2014.
- [6] Jimmy SJR, Xiaoyu W, Wei W, Huizhong G, Guan W, Stephen SL. International Conference on Information Systems (ICIS 2013): Reshaping Society Through Information Systems Design. 2013;2:1367-1375
- [7] Xiaodong L, Haoran X, Li C, Jianping W, Xiaotie D. News impact on stock price return via sentiment analysis. Knowledge-based Systems. 2014:14-23
- [8] Sohangir S, Wang D, Pomeranets A, Khoshgoftaar TM, Big Data: Deep learning for financial sentiment analysis, Journal of Big Data 2018;5(3)
- [9] Womack KL. Do Brokerage Analysts' Recommendations Have Investment Value? Journal of Finance. 1996;51(1)
- [10] Yung-Ho C, Chai-Chung C. Financial analysts' stock recommendation revisions and stock price changes. Applied Financial Economics. 2008;18(4):309-325
- [11] Goldberg Y. Neural Network Methods for Natural Language Processing. Morgan & Claypool Publishers;2017.
- [12] Lindholm A, Wahlström N, Lindsten F, Schön TB. Supervised Machine Learning. Uppsala. Uppsala University; 2019. available at: [http://www.it.uu.se/edu/course/homepage/sml/literature/lecture\\_notes.pdf](http://www.it.uu.se/edu/course/homepage/sml/literature/lecture_notes.pdf) (Accessed 2019-03-06)
- [13] James G, Witten D, Hastie T, Tibshirani R. An Introduction to Statistical Learning with Applications in R. 7 ed. New York: Springer Science+Business Media;2013.
- [14] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436–444.
- [15] Glorot X, Bordes A, Bengio Y. Deep Sparse Rectifier Neural Networks. Journal of Machine Learning. 2011;15:315–323.
- [16] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature. 1986;323(6088):533–536
- [17] Nielsen MA. Neural Networks and Deep Learning. Determination Press. 2015. Chapter 2

- [18] Nielsen MA. Neural Networks and Deep Learning. Determination Press. 2015. Chapter 1
- [19] Goodfellow I, Bengio Y, Courville A. Deep Learning, Chapter 5. MIT Press; 2016, available at: <http://www.deeplearningbook.org> (Accessed 2019-03-12)
- [20] Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research. 2011;12:2121–2159.
- [21] Tieleman T, Hinton GE, Lecture 6.5 - RMSProp, COURSERA: Neural Networks for Machine Learning. Technical report. 2012.
- [22] Kingma DP, Ba JL. Adam: A method for stochastic optimization. arXiv:1412.6980. 2014.
- [23] CS231n Convolutional Neural Networks for Visual Recognition. Stanford University. available at: <http://cs231n.github.io/neural-networks-2/> (Accessed 2019-03-12)
- [24] Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research. 2014;15:1929–1958
- [25] Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov R. Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580. 2012.
- [26] Nielsen MA. Neural Networks and Deep Learning. Determination Press. 2015. Chapter 6
- [27] Kalchbrenner N, Grefenstette E, Blunsom P. A convolutional neural network for modelling sentences. arXiv:1404.2188. 2014.
- [28] Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. In Information Processing & Management, 1988;24(5):513-523.
- [29] Mikolov T, Chen K, Corrado GS, Jeffrey D. Efficient estimation of word representations in vector space. arXiv:1301.3781. 2013.
- [30] Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J, Distributed representations of words and phrases and their compositionality. arXiv:1310.4546. 2013
- [31] Will Kenton, Earnings Yield, available at: <https://www.investopedia.com/terms/e/earningsyield.asp> (Accessed 2019-04-01)
- [32] Gensim, available at <https://radimrehurek.com/gensim/> (Accessed 2019-04-03)
- [33] Scikit-learn, available at <https://scikit-learn.org/stable/index.html> (Accessed 2019-04-05)
- [34] Keras, available at <https://keras.io/> (Accessed 2019-04-08)
- [35] Tensorflow, available at <https://www.tensorflow.org/> (Accessed 2019-04-08)

- [36] Zhang Y, Wallace BC. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. arXiv:1510.03820. 2016.
- [37] Shahriari B, Swersky K, Wang Z, Adams RP, de Freitas N, Taking the Human Out of the Loop: A Review of Bayesian Optimization. Proceedings of the IEEE 2016;104(1):148-175
- [38] Wang J, Yu LC, Lai KR, Zhang x. Dimensional Sentiment Analysis Using a Regional CNN-LSTM Model. 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Short Papers. 2016:225-230
- [39] Yenter A, Verma A. Deep CNN-LSTM with combined kernels from multiple branches for IMDb Review Sentiment Analysis. 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference. 2017:540-546.