Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Degree Programme in Software Engineering

Bachelor’s Thesis

Saku Suikkanen

USING RECURRENT NEURAL NETWORK MODELS AND FINANCIAL NEWS FOR PREDICTING STOCK MARKET MOVEMENTS

Supervisor: Post-doctoral researcher Annika Wolff

TIIVISTELMÄ

Lappeenrannan-Lahden teknillinen yliopisto LUT
School of Engineering Science
Tietotekniikan koulutusohjelma
Saku Suikkanen

Rekursiivisten neuroverkkojen ja talousuutisten käyttö osakemarkkinoiden ennustamisessa

Kandidaatintyö 2020

44 sivua, 13 kuvaa
Työn tarkastajat: Tutkijatohtori Annika Wolff

Hakusanat: neuroverkko, osakemarkkinat, luonnollisen kielen käsittely
Keywords: neural network, stock markets, natural language processing

Tässä työssä tarkastellaan talousuutisten hyödyntämistä yhdessä koneoppimisen kanssa osakemarkkinoiden ennustamiseen. Uutisia käsitellään tietyillä luonnollisen kielen käsittelyn menetelmillä ja niistä pyritään löytämään korrelaatiota osakkeen liikkeiden kanssa. Työn uutuusarvona ovat BNS- ja LDA-menetelmien, sekä 2 sanan kombinaatioiden käyttö LSTM-neuroverkon yhteydessä. Työn pääasiallisena tavoitteena on tarkastella edellä mainittujen menetelmien ja neuroverkkojen yhdessä tuottamien tulosten hyödyllisyyttä ja niiden vertautumista markkinoiden tehokkuuteen. Tutkimuksessa selvisi, että mallit, jotka sisältävät BNS-menetelmällä johdettuja 2 sanan kombinaatioita, tuottavat poikkeavia tuloksia niihin malleihin verrattuna, joissa niitä ei käytetä. Tulokset kuitenkin noudattivat kaiken kaikkiaan satunnaista vaihtelua ja luotettavia tuloksia ei täten saatu. Tulosten parantamiseksi, parempi lähtökohta voisi olla päivän sisäisten vaihtelujen ennustaminen, markkinoiden tehokkuuden hypoteesin mukaisesti. Käytetyt tietoaineistot olivat myös mahdollisesti liian suppeat kompleksisuudeltaan vaativalle ongelmalle.

ABSTRACT

Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Degree Programme in Software Engineering
Saku Suikkanen

Using recurrent neural network models and financial news for predicting stock market movements

Bachelor's Thesis 2020

44 pages, 13 figures
Examiners: Post-doctoral researcher Annika Wolff

Keywords: neural network, stock markets, natural language processing

In this work, the utilization of financial news alongside machine learning for predicting stock market movements is examined. The news are handled with various natural language processing methods for finding correlation between the derived attributes and stock market movements. The novelty of this work lies in the application of BNS and LDA methods as well as 2-word combinations alongside with LSTM neural network. The main point of the work is to examine the usefulness of the results achieved with the formerly mentioned methods and neural networks as well as comparing the results with market efficiency. In the research it was concluded that the models containing 2-word combinations derived with the BNS-method, produced differing results to those models, where the 2-word combinations were not used. However, the overall results followed random patterns and thus reliable results were not achieved. For achieving more reliable results, better approach could be predicting intraday stock market movements per Efficient Market hypothesis. Used datasets were possibly also too concise for the complexity of the problem.

TABLE OF CONTENTS

1 INTRODUCTION 3

2 RELATED WORK 5

3 PROPOSED APPROACH 10

3.1 DATASETS 10

3.2 DATA PREPARATION 10

3.3 TEXT DATA 12

3.3.1 Merging articles and market data 12
3.3.2 Data Cleaning 12
3.3.3 Features 13
3.3.4 Topic modeling 16

3.4 FINANCIAL DATA 18

3.4.1 Data cleaning 18
3.4.2 Features 18

3.5 PROPOSED MODEL 20

3.5.1 Recurrent Neural Network 20
3.5.2 Long short-term memory 21

3.6 ENVIRONMENT AND TOOLS 22

4 MODEL DEFINING 24

5 RESULTS 26

5.1 FIRST MODEL 27

5.2 SECOND MODEL 30

5.3 THIRD MODEL 32

5.4 FOURTH MODEL 33

5.5 DISCUSSION 34

6 SUMMARY 37

REFERENCES 38

ABBREVIATIONS AND SYMBOLS

BNS	Bi-Normal Separation
CPU	Central Processing Unit
EMH	Efficient Market Hypothesis
FNN	Feedforward Neural Network
GPU	Graphical Processing Unit
LDA	Latent Dirichlet Allocation
LSTM	Long Short-Term Memory
NaN	Not a Number
NLP	Natural Language Processing
RNN	Recurrent Neural Network
RWH	Random Walk Hypothesis
S&P 500	Standard & Poor’s 500
SVM	Support Vector Machine
TF IDF	Term Frequency – Inverse Document Frequency

1 INTRODUCTION

The information considering the future events around the globe have always been valuable for stock market traders because the value of traders' investments is directly proportional with the outlook of the invested company's profitability. Thus, traders have traditionally resorted to analyzing companies and their qualitative and quantitative aspects before the investment decision to ensure the increasing returns of their investments. The common methods for analyzing companies' future outlooks can be broken to two categories, technical analysis and fundamental analysis. Technical analysis focuses on the historical quantitative data such as historical prices, volume and volatility, while fundamental analysis aims to evaluate the company's future value by looking at, not only quantitative but also qualitative aspects such as investments, management and mergers.

One of the biggest sources for executing fundamental analysis comes from daily news. Due to the globalized world, improved transmitting of information and the market's complexity on social, political, economic and psychological levels, it is extremely hard for a single trader to read and analyze every publication and make accurate investment decisions. Researchers have applied various machine learning and *Natural Language Processing* (NLP) methods in search of a solution to create predictive models based on news articles. One of the most commonly used model seems to be *Support Vector Machine* (SVM) and while SVM has been proved to accomplish great results, the models are still far from reliable (Dang and Duong, 2016; Hagenau et al., 2013; Zhai et al., 2007). Lately by increased processing power of computing systems, more complex models have been able to practically implement and one high potential candidate is the relatively recent *Long Short-Term Memory* (LSTM) introduced by (Hochreiter and Schmidhuber, 1997). While LSTM has been widely used in predicting stock market movements based on historical price data i.e. technical analysis, its applications for news-based models remain minuscule. However, the capability of the model to hold long term memory has proven to be effective when dealing with time series data and creates intriguing opportunity research its capabilities further. (Fischer and Krauss, 2018; Huynh et al., 2017; Nelson et al., 2017; Persio and Honchar, 2016)

This thesis researches the methods and applications for the aim of creating a predictive machine learning model for stock market using LSTM model. Thesis will focus on US stock markets and more precisely on predicting *Microsoft* stock movements. Microsoft stock was selected as it is the largest company by market cap included in the *Standard & Poor's 500* index (S&P 500) on which Microsoft holds a weighting factor of 4% (Barron's, 2019). Since the S&P 500 is a good indicator on broader US markets and Microsoft stock is the single greatest factor that influences the S&P 500, it could be argued that Microsoft stock corresponds more on the fluctuations of broader economy than smaller companies.

The models features will consist of similar technical indicators used in technical analysis as well as news-based data. The target attribute will be the stock's movement (up, down) since the classification approach has been concluded as being more suitable for such a complex problem as explained in chapter 2. The created model will be finally inserted with a set of test data to evaluate the model's accuracy on predicting ability.

The novelty in this thesis will be the applications of feature selection and extraction methods that have not been used with LSTM model within literature on financial prediction. These methods are *Bi-Normal Separation* (BNS), *Latent Dirichlet Allocation* (LDA) and 2-word combinations, and the aim is to research the effect of these methods on the results.

The main research question for this thesis can be formulated as followed:
How the BNS, LDA and 2-word combinations perform alongside LSTM neural network on predicting stock market movements?

Also, the following sub question is answered:
How does the results compare with market efficiency?

2 RELATED WORK

The predictability of stock markets has been popular topic of study even before the application of machine learning methods and one of the most notorious studies is the *Efficient Market Hypothesis* (EMH), introduced by (Fama, 1970). EMH's semi-strong form suggests that the stock markets fully reflect all available public information at any time, and it is closely associated with the *Random Walk Hypothesis* (RWH). According to the RWH, stock market prices follow an unpredictable pattern and matching this with EMH would state that public information would also follow an unpredictable pattern. (Fama, 1995, 1970)

Many of the machine learning related work on predicting the stock market is based on the EMH and RWT, which is why many of those studies have concluded that predicting only the examined stocks movement and thus the most favorable option (buy, sell, hold) for the trader, rather than accurate closing price produces more accurate models.

The main differences between earlier studies on the topic can be split on news text data preprocessing, other features, used algorithm or neural network and the wanted output. Most of the related work contain similar approaches to one another on at least some of points mentioned above.

The news data handling primarily differs in three aspects, which are the datasets, news data sentiment classification, data preprocessing and feature qualification. Datasets for prior research have basically included the news data and stock market data. Datasets have been collected from various news and stock market data providers. The commonality between the stock data in different research is the predicted stocks are included in some significant stock market index or the prediction is based on the index such as the S&P 500, Korea composite stock price index (KOSPI) and VN30 (Dang and Duong, 2016; Fischer and Krauss, 2018; Kim, 2003). Stock market indexes are used because they represent better the overall sentiment of stock markets in contrast to individual stocks.

In most prior and recent research, the news articles were classified based on the sentiment derived from the context. Different methods for article sentiment classification are based on two approaches, manual and automatic classification. Manual classification can be executed by e.g. domain experts by reading the article and deciding on the sentiment of the article context. Automatic classification however is done by e.g. mapping the article publish date and the current stock price at that date. Thus, by looking at the current stock price movement i.e. up or down, the article can be classified as positive or negative. Manual classification is naturally more accurate since an expert of field can fairly accurately evaluate the outcome on which industries and stocks it will affect and the magnitude of the impact as well as consider the exogenous associations on the matter. Thus, automatically evaluating the sentiment of the article is inaccurate as such since it won't consider the matters outside of the context of the article. (Dang and Duong, 2016; Hagenau et al., 2013)

Manual classification is commonly dictionary-based, where the dictionary contains e.g. most used words in their relative sentiment that domain experts have manually declared. In (Tetlock et al., 2008) researchers used the Harvard-IV-4 psychological dictionary for classifying the words that included in the negative and positive sentiment. The Harvard-IV-4 dictionary is widely used in psychological research across different fields of studies. However, in (Loughran and Mcdonald, 2011) it is stated that nearly three fourths of the words that are classified as negative in the Harvard-IV-4 dictionary are essentially not negative in financial context.

Automatic classification is commonly executed by mapping the momentary stock price and the news article, where the stock price movement direction acts as the alleged sentiment of the article (Dang and Duong, 2016; Hagenau et al., 2013; Mittermayer, 2004). The disadvantage related to manual classification is reduced accuracy. Since the complex nature of stock markets, it is difficult to implement such a general method for accurately analyzing the market sentiment for set of news.

After classifying whether an article is negative or positive related to stock market sentiment it is necessary to execute a feature extraction process. The initial features from the news text can be single words, n-grams, noun-phrases or 2-word combinations as introduced in (Hagenau et al., 2013, p. 689). For extracting these features from textual data, the document must be preprocessed. The most commonly used methods for preprocessing are removing stop words such as “the”, “a”, “or”, etc., word tokenizing, word stemming and punctuation removal. In word tokenizing, the whole text is transformed into a vector that contains all the single words that were e.g. delimited by blank character. For the resulting vector word stemming is applied, which reduces the number of features by adding the words together that contain the same body when plurals and/or inflected forms are disregarded. (Kao and Poteet, 2007)

This process is implemented to every article and the resulting vectors for negative and positive categories contain e.g. frequencies of the appeared words. The proceeding feature vectors can still be millions of columns long as in (Hagenau et al., 2013, p. 691), which would make the learning extremely ineffective and would led to overfitting. Overfitting is a common issue when dealing with machine learning algorithms and neural networks. While overfitted model seems to work well with data it has been presented before, but its performance could be abysmal when dealing with data it has not seen before. (Sebastiani, 2002, p. 15) Thus, using only these simple measures is usually insufficient, especially when dealing with n-grams or word-combinations. Therefore, in most researches a feature selection process has been initiated by using metrics for selecting features as mentioned in (Forman, 2003; Yang and Pedersen, 1997). One method that has been proven to be supreme when dealing with high number of features in classification problems is Bi-Normal-Separation (Forman, 2003, pp. 1298–1299), which is also used in (Hagenau et al., 2013) for feature selection in predicting stock market movements. In the same research, the BNS-model achieved the highest returns in market simulation. The BNS-method can be used to rate the prevalence of features in both, negative and positive classes and thus it helps on evaluating the potential influence of every word.

After feature selection the final step for text processing is feature representation. Feature representation is essential when dealing with textual data because machine learning models cannot take textual data as an input. There are multiple variants for representing document terms as numerical data within the feature vector and one of the most popular method for weighting features is *Term Frequency – Inverse Document Frequency* (TF IDF). The idea behind TF IDF is to look at the product of TF and IDF where TF is used to rate higher the influence of words that occur the most in document and IDF to do the inverse, which is rating the explanatory power of least occurring terms higher. The product of these two will then give an estimation of which terms have the highest influence. (Mittermayer, 2004)

The models based on news usually contain other features besides the preprocessed text data. These features can be the actual values derived from stock market data such as high, low, close prices, volume and change of price between dates (Zhuge et al., 2017). In some research, more advanced features used in technical analysis were set using the stock market data and applying the data to different mathematical formulas such as stochastic oscillators (%D, %K), momentum, Williams' percent range (%R) and n-day disparities (Kim, 2003, p. 311; Zhai et al., 2007, p. 1089).

The most commonly occurred machine learning model for news-based approach seems to be SVM and it has been proven to produce the best results against other models on text classification and financial prediction (Hagenau et al., 2013, p. 690). Although, SVM based models have been proved to produce greater accuracy in technical indicator based as well as text-based approaches, the potential for LSTM models on the matter seems promising (Fischer and Krauss, 2018; Hagenau et al., 2013; Huynh et al., 2017; Kim, 2003). LSTM networks have been considered some of the best models for NLP but the research on their applications for news-based approaches on predicting the stock market are currently rather limited (Graves et al., 2009; Huang et al., 2015; Zhou et al., 2015). For example, in (Graves et al., 2009) researchers topped the previous state-of-the-art model in handwriting recognition by using a LSTM-based model. Altogether, gated recurrent models such as the LSTM have been proven to alleviate the vanishing/exploding gradients problem that occur in standard *Recurrent Neural Networks* (RNN) and *Feedforward Neural Networks* (FNN) for better learning capability. As in (Siah and Myers, 2015), gated recurrent model was used to provide significantly better results in financial prediction compared to non-gated model.

Prior research on stock market prediction via machine learning methods could be generalized to predicting only the movement of prices, where the method would be to classify the movement of certain stock/index based on the features. Commonly this is done by creating three classes for up, hold and down or two classes for only up and down. The alternative way is to predict the actual price of the stock, which would require to predict also the magnitude of movement. In (Hagenau et al., 2013), researchers compared predicting only the movement and actual value on their SVM model. For their best model with 2-word combinations as a feature and BNS as feature selection base, they got 76,3% accuracy on predicting movement and only 20,2% accuracy on predicting the actual returns.

3 PROPOSED APPROACH

In this chapter, the used datasets, methods for preprocessing and the used machine learning models in this work are explained. All the used environments and tools used in e.g. preprocessing and feature extraction are explained in section 3.6.

3.1 Datasets

The datasets for this thesis consist of market dataset from *Quandl* and financial news dataset from (Nguyen, 2019). The market data is derived from the same time period as the articles have published. The news dataset contains some 110 000 news articles from international news organization *Reuters* and the articles publication dates range from October 2006 to November 2013. Therefore, the datasets represent time series.

The articles contain financial news, which contain relevant information about the economy and other financial anomalies at the time. The same news dataset was compiled and initially used in (Ding et al., 2014), where individual stock movements were predicted with nearly 70% accuracy. Considering these notable results, it is evident that the information contained by these articles is suitable for this problem.

3.2 Data preparation

Both datasets in this work are considered as time series, which is described in (Brockwell and Davis, 2016) as “a set of observations, each one recorded at a specific time.“ This applies to both datasets since each article is considered as an observation of information at specific time and same applies to stock market data. The difference between the information content of these datasets are obviously the form they are presented in, articles in textual data and stock market in numerical data. Thus, the ultimate methods for preparation for each of these datasets are different but there are general approaches on how time series modeling could be approached.

In (Brockwell and Davis, 2016), the general approach for time series modeling is described as followed:

1. Examine plotted graphs for
    a. a trend
    b. a seasonal component
    c. any apparent sharp changes in behavior
    d. any outlying observations
2. Remove trend and seasonal components to get stationary data
3. Choose a model to fit the stationary data
4. Forecasting and inverting the transformations for comparison

Data is converted to stationary form when the mean and variance does not correlate with time. In other words, the seasonal components have been eradicated and as a result, the data is fairly similar at every point in time. (Brockwell and Davis, 2016)

However, for ensuring the correctness and suitability of the data, other data preparation processes must be implemented before formerly mentioned time series modeling. These processes consist of data cleaning, outlier detection, feature extraction, feature selection and feature representation. (Berthold et al., 2010)

Data cleaning is the process of minimizing noise such as simple errors, inconsistencies and varying formats, also known as noise. Outlier detection could be considered as a part of data cleaning as it is used to detect and delete inconsistencies that could potentially create drastic errors in modeling. Feature extraction, selection and representation are implemented when the data can be classified as clean. These processes are used for extracting the features that will be used as an input for the model, selecting the best features from the corpus of extracted features and finally representing the selected features in a form that is applicable to be used for learning by the model. (Berthold et al., 2010)

3.3 Text data

In this chapter, the methods for processing textual news data, extracting and representing features are discussed.

3.3.1 Merging articles and market data

Firstly, the articles published during each day had to be converted to match the market times. Since Microsoft stock is traded in the *Nasdaq* stock exchange, the articles had to be assigned so, that if the article is published before the closing time of Nasdaq at 16:00 it will be assigned for the current day but if the publish time is over the closing hour, the article is assigned to the next day. Also, if the article is published during weekend, it is assigned to next Monday. Secondly, each article was grouped by its date assigned in the first step and merged with the corresponding date in the market data.

3.3.2 Data Cleaning

Data cleaning is crucial process especially for textual data as it is littered with noise which can be identified as punctuation, differing case sensitivity, numerical values, inflected forms, etc. (Berthold et al., 2010). Common noise reduction methods are such as stop word removal, tokenizing and stemming (Kao and Poteet, 2007).

After cleaning the text from punctuation, differing case sensitivity, numerical forms, etc., the next process is commonly word tokenizing which separates each individual word inside the document to its own token, i.e. converts the document into array of words. After tokenizing, common NLP practices involve stemming or lemmatizing each individual token. These methods produce the benefit of combining same words that may have e.g. inflected forms. The difference between stemming and lemmatizing is that stemming aims to find the common root of the word, whereas lemmatizing aims to find the actual dictionary presentation of the inflected word. (Kao and Poteet, 2007)

Final NLP method is stop word removal. Stop words are considered common English words such as pronouns, articles and prepositions. Removing excess stop words is crucial, as they have tendency for high prevalence in a set of articles and can marginalize words that have actual explanatory power which leads them to be excluded from the feature corpus if stop words are not to be removed. (Kao and Poteet, 2007)

In this work, the news data was cleaned by first removing numerical values from the articles, converting each word to lower case, removing punctuation symbols, tokenizing, lemmatizing and finally removing stop words.

Missing values also enhance the noise level of data and fixes for this problem can be ignorance/deletion, imputation and explicit variables. The used method for this work was ignorance/deletion and it only influenced the news data since the stock data from Quandl was completely intact. The ignored news data was mainly due to missing values, either date or the actual article was defective.

3.3.3 Features

After ensuring the data quality by cleaning the data, next step was extracting features from the cleaned datasets. The selected feature type for this work was 2-word combinations as initially introduced in (Forman, 2003) and firstly implemented as text classification method in (Hagenau et al., 2013). The reason for selecting this feature type was that it hadn't been used before with LSTM neural networks and it proved to give best results in the formerly mentioned research compared to single words, N-grams and noun phrases. A comprehensive explanation can also be found in the same work.

Feature selection is the process of finding the most optimal features to be used as an input for the model. For selecting the most prominent features, some evaluation function must be implemented. In this work, so called BNS method was used as the evaluation function which has been proven to be outstanding when dealing with high number of features in classification problems (Forman, 2003). Basically, the BNS measures the prevalence of single word, or 2-word combination in this work, by calculating the standard Normal distribution's inverse cumulative probability score, i.e. z-score, of the rate of certain positive words frequency in whole positive word corpus and vice versa for negative words. In this work the BNS is used to evaluate each feature with a value that corresponds to the prevalence of each feature in negative and positive class. Thus, the selected features are n-number of top words from both categories. The word combinations are evaluated in such way, that every combination of same words is added together, so the order of the words does not matter. For example, “negative forecast” and “forecast negative” are counted as same combination.

The formula can be defined as followed per (Forman, 2003):

$$BNS = F^{-1}(pr) - F^{-1}(nr)$$

Where *F* corresponds as the standard Normal distribution’s cumulative probability function, *pr* corresponds as the number of certain word occurrences in positive context divided by the total amount of occurrences in both, positive and negative classes. The variable *nr* acts as the opposite of *pr* by defining the rate of certain word occurrences in negative context. Zero values are replaced with 0.0005 to avoid *NaN* (Not a Number) values. (Forman, 2003)

Feature representation is the final process for the text data before it is fit into the model. For each day in the stock market data, a bag of words vector-based representation is executed for the words selected with the BNS method. This means that for each selected word, its prevalence for each day is determined and whether it includes in the current days articles it gets a value of 1 and if not, it gets a value of 0.

In the following figure, the 2-word combination bag of words representation is visualized.

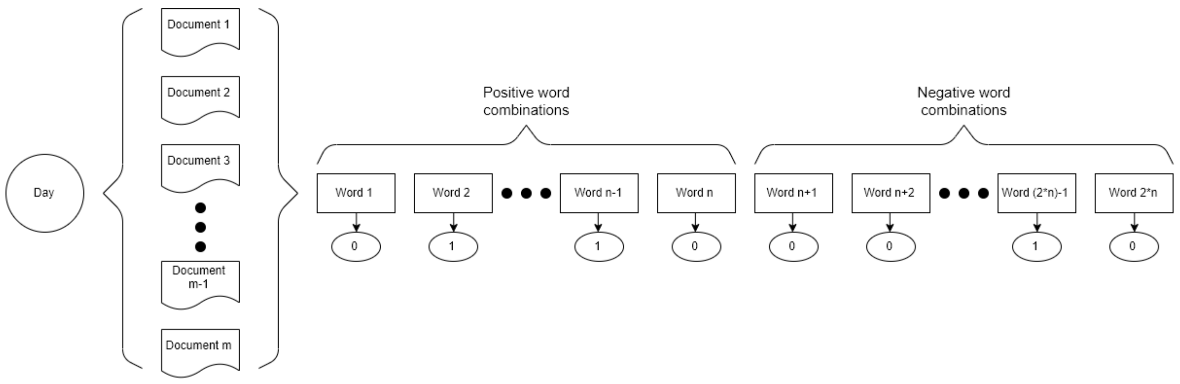

Figure 1 Text based feature representation.

In the preceding figure, every day contains m number of documents from which, the prevalence of certain 2-word combinations are determined as 0 or 1. There are n number of 2-word combinations in both, negative and positive, classes.

3.3.4 Topic modeling

Some of the features are extracted by so called probabilistic topic modeling, which is a method for defining probabilities of different topics that can best describe set of observed data. (Aggarwal and Zhai, 2012)

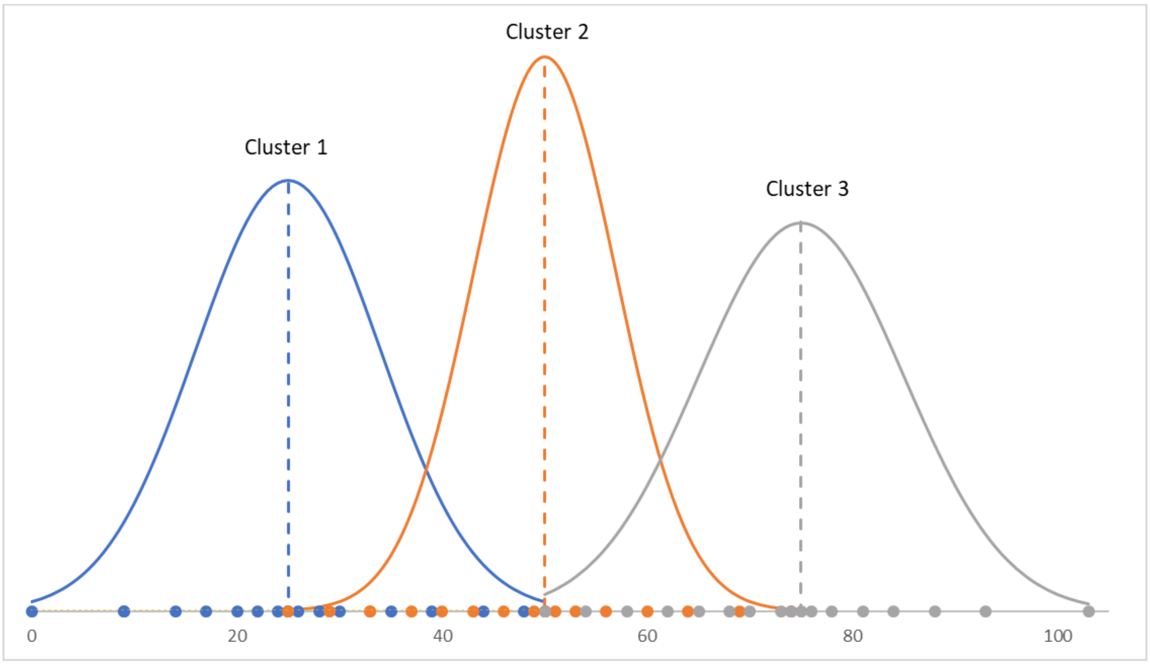

Figure 2 Mixture model working principle.

There are available a plethora of different probabilistic models and in this work the used model is so called LDA. LDA is one of the most frequently used fundamental model and more closely it fits into the category of mixture models. Mixture models generally work by clustering data points where each component is a distribution for a certain cluster and each data point belongs to a cluster by some probability. (Aggarwal and Zhai, 2012) In figure 2, the mixture modeling method is visualized, where each line corresponds to the normal distribution of certain cluster and each point in the x-axis corresponds to the observed data point. As LDA is only one of the various feature extraction/selection methods used in this work, it is not explained in more detailed manner. Detailed explanation on how LDA operates can be found in the original work in (Blei et al., 2003).

The topic modeling was started by first implementing similar NLP and data cleaning techniques to the text data, such as stop word removal, punctuation removal and lemmatizing.

The cleaned data could be then fitted to the LDA model and in this work the used model was extracted from *scikit-learn*-library. The optimal number of topics was defined using *GridSearchCV* function from the same library to iterate from a range of topic sizes and other parameters.

After concluding the topic modeling, it was clear from data that it represented non-stationarity. Thus, transformation and differencing were applied to achieve stationary data. However, transformation on logarithmic scale was not possible as the values for each topic varied from 0 to 1 and transforming value of 0 to logarithmic scale concluded to *NaN*-value. Thus every 0 value was converted to 0.01, which allowed for a transformation on logarithmic scale.

3.4 Financial data

In this chapter, the methods for processing numerical financial data, extracting and representing features are discussed.

3.4.1 Data cleaning

Data cleaning for the stock data consist mostly of outlier detection and can be done by visually examining the plotted data. By looking at the plots of each feature it was concluded that the stock market data did not include any outliers that could create false results.

3.4.2 Features

The selected features from the stock data for each day were closing price, opening price, volume, high price and low price. Also, based on these features, new features were calculated such as percentage of difference between high and low prices, momentum, Stochastic %K and Stochastic %D oscillators.

Before the feature extraction could be implemented, the closing price had to be shifted 1 step into the future, so that the current day for which the prediction would be done, could not see the actual closing value of the day. That is, at the current day it could be only known what the previous closing price was.

Because the plotted stock data clearly indicated trends and seasonality, the data had to be converted to stationary form. In this work, the stock data was converted by first transforming the daily values into logarithmic scale, which reduces the overall variance since the difference between smaller and larger values are smaller. Next step was setting mean to zero over time, which was achieved by so called differencing method where the daily data was subtracted from previous day values. (Brockwell and Davis, 2016)

In figure 3 and figure 4, the effects of converting data to stationary form are visible.

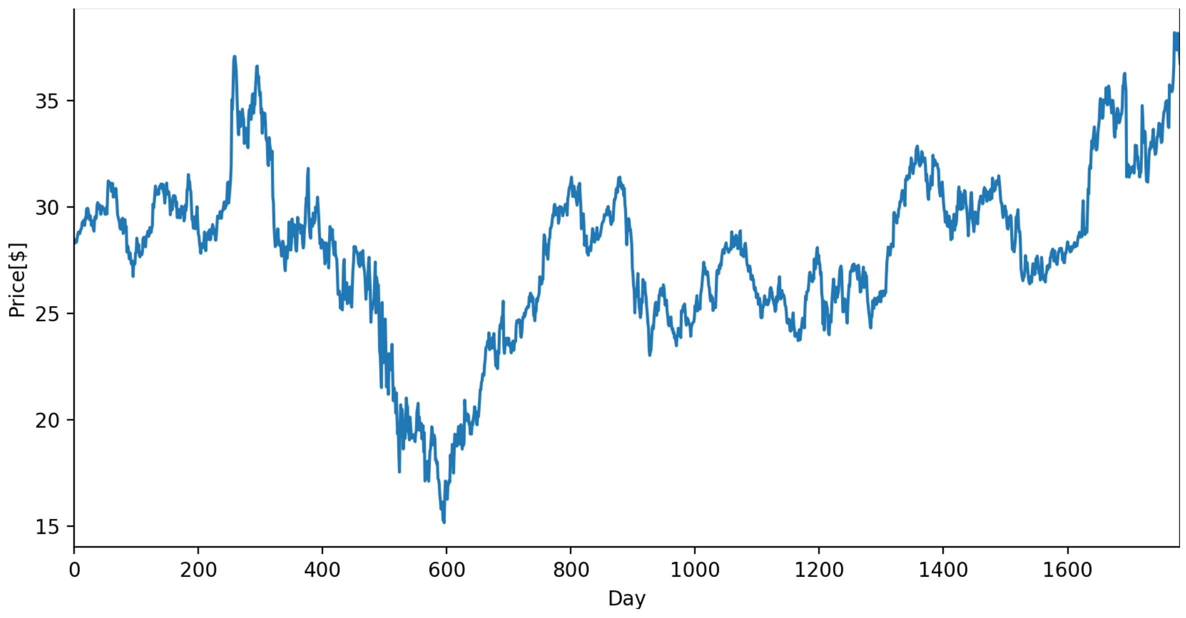

Figure 3 Microsoft stock price by day.

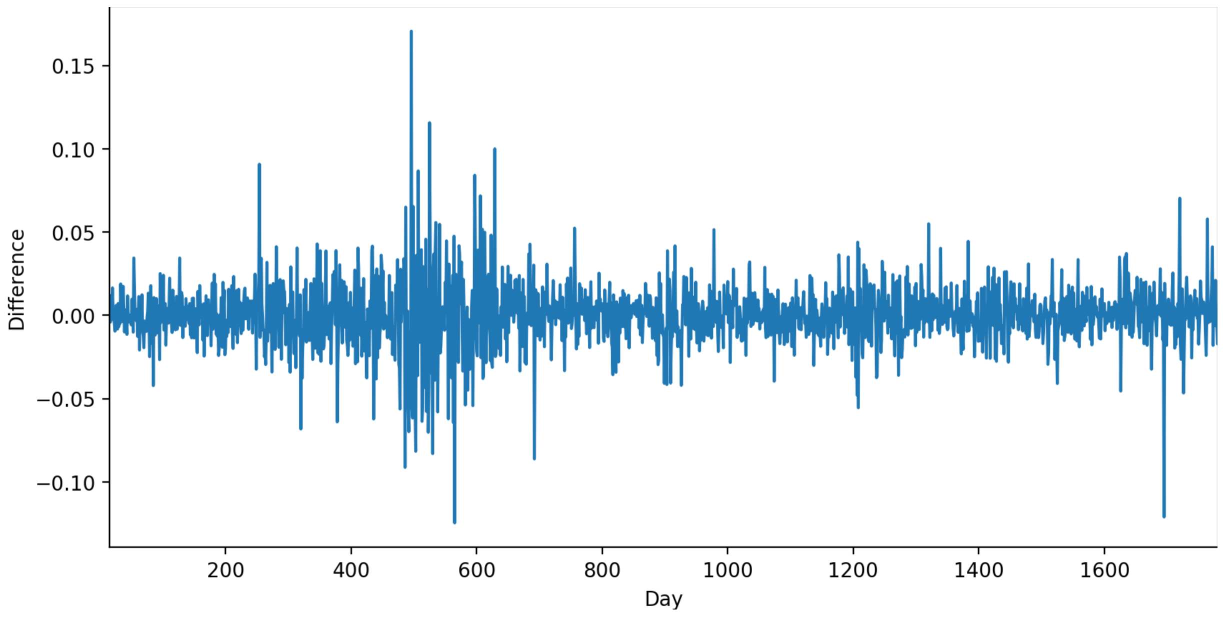

Figure 4 stationary Microsoft stock price by day

Comparing preceding figures**,** it is evident that variance is minimized, and mean is close to zero. As a result, seasonality is considerably reduced.

Also, all the features were scaled on interval of 0 to 1 using *MinMax* scaling function from scikit-learn-library. Scaling is generally recommended when training neural networks because if the different feature values fluctuate at different magnitudes, the model could emphasize the importance of the high values. (Berthold et al., 2010)

3.5 Proposed model

The selected model for the work was LSTM as it has been proven to be effective when dealing with time-series data. Also, some of the selected features have not been used in literature along with LSTM models which creates an opportunity to test their performance.

3.5.1 Recurrent Neural Network

LSTM fits in the group of Recurrent Neural Network architectures, which differ from Feedforward Neural Networks by the capability of using previous data recurrently i.e. remembering previous results. Thus, in similar case of predicting stock markets, FNN would use all of the inputted data, e.g. from time interval of 6 months and calculate the output based on the whole 6 months of data. RNNs however, can use the 6 months of data in sets of e.g. 1-month intervals, and utilize the interdependence of the data. (Goyal et al., 2018)

RNNs are especially useful for problems that contain sequential data such as time series. These kinds of datasets are called temporal data, which means that each observed data point is somehow dependent on previous data. Stock market data can be identified as temporal data as company's share prices are dependent on previous prices on a scale of days/months/years. (Goyal et al., 2018)

However, RNNs possess a certain problem called vanishing/exploding gradient that occurs when the weights are propagated through various timesteps and multiplied recursively in the previous functions. This means that when the weight is being multiplied various times, depending on whether the weights value is small or large, it starts to either vanish to zero or explode to infinity as it is passed through the timesteps. Thus, the RNNs are very sensitive to the number of timesteps and sequence lengths. (Goyal et al., 2018)

3.5.2 Long short-term memory

LSTM networks differ from the standard RNNs by having the capability to remember long-term results and relations which has been proven to alleviate the problem of vanishing/exploding gradients. This is possible through additional input and forget gates in the architecture that control on what information to preserve and what to forget and thus, keeping the gradient from vanishing/exploding. (Goyal et al., 2018)

The model is not accurately described in this work as the scope of the work is to merely evaluate the performance of certain features alongside with the LSTM network. However, a more general introduction to LSTM architecture is presented.

LSTM architecture consists of multiple cells that represents each time step. Each cell contains the gated structures used for manipulating the passing information. The gated structures are:

- Forget gate
- Input gate
- Output gate

Each cell is fed with the preceding cell state and hidden layer value which the cell uses to determine what information is kept and what is forgotten. In figure 5, the LSTM module is visualized.

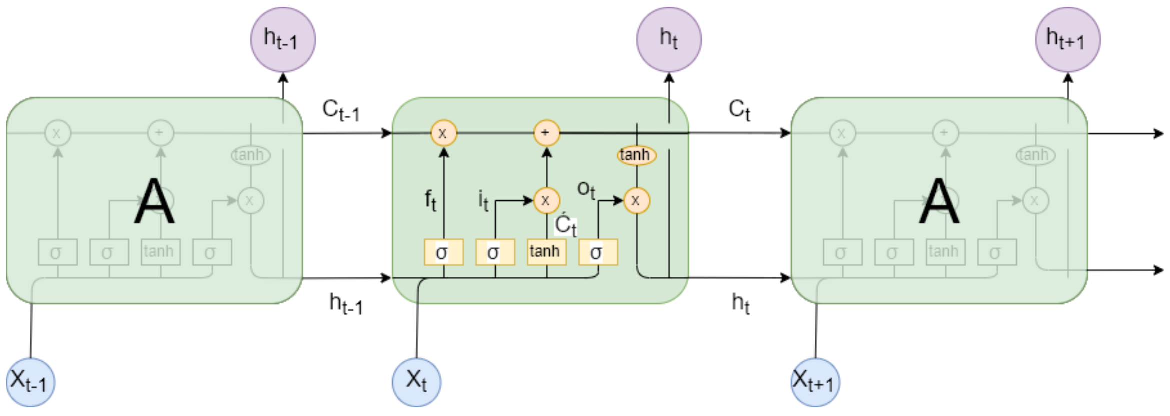

Figure 5 LSTM architecture after Goyal et al. (2018)

In the figure, C_t corresponds to the cell state at time t which is passed through every cell and changed accordingly to retain the information through each timestep. X_t is the input at time t and h_t is the value i.e. output that is passed onto the hidden layer at time t. The input gate i_t controls the contribution of new input values to the cell state, forget gate f_t estimates whether a value is relevant at time t and controls its inclusion/exclusion. Finally. output gate o_t controls the use of cell state in the output activation. (Goyal et al., 2018)

3.6 Environment and tools

The main developing environment was *JupyterLab* web-based interface which suits perfectly for the repetitive nature of data analysis and statistics because of its notebook style developing capability, where the code can be run in small snippets rather than needing to run the whole code every time. As a programming language, *Python* was used because of its suitability for data analysis thanks to some of its popular libraries such as *pandas*, *matplotlib*, *nltk*, *scikit-learn* and *numpy*. Pandas was the mostly used library and it provides excellent methods alongside with numpy for modifying, preparing and cleaning data. Matplotlib was used for the visual presentations and nltk for some of the NLP methods such as lemmatizing and stop word removal. Lemmatizing was done via nltk library's *WordNetLemmatizer* and stop word removal via the nltk's standard English stop word list which was extended by a couple of additional words considered as stop words.

For the implementation of neural networks, the used framework was *TensorFlow* which is machine learning library developed by *Google*. Since TensorFlow can be difficult to use as it is, another library called *Keras* was used on top of TensorFlow. Keras is a high-level neural networks API, which is known for its ease of use, modularity and flexibility. So, in this work, TensorFlow was used as the backend for providing the framework for building the neural networks and Keras was used as basically the frontend by providing easier implementation of the neural networks. Both TensorFlow and Keras are written in Python, so they are completely compatible with the mentioned programming language. Also, since deep learning model training can be time consuming, there was a desire to use *Graphical Processing Unit* (GPU), rather than *Central Processing Unit* (CPU) for training the model. GPUs can provide 2-3 times faster training compared to CPUs because of their parallel computing capabilities.

For implementing TensorFlow to work on a GPU, it was needed to create a *Anaconda* (Anaconda, Inc, 2019) virtual environment for easy installation of *TensorFlow GPU*. (Baltazar, 2018)

4 MODEL DEFINING

Before defining the model's parameters, the dataset was split into two sets, training and test data. Training data is used as input for the model in the training phase for fitting the model and test data is used to evaluate the performance of the model. As a result, the fitted model has not seen the test data and thus, it can be used to evaluate how the model performs on unseen data. (Berthold et al., 2010)

According to (Goyal et al., 2018), some of the most important hyperparameters for defining RNN models are:

- Hidden layer size
- Learning rate of optimizer
- Dropout rate
- Number of Epochs

Hidden layer size corresponds to the number of neurons in the layers between input and output. Selecting the number of neurons is crucial for the model to be able to learn correctly. Too small hidden layer size can result to underfitting, where the model is not able to detect the complicated signals and thus, will not generalize well. On the contrary, too many neurons can result to overfitting as well as increased time on training the model. Selecting the optimal number of neurons can be tricky, if not impossible, but generally the hidden layer size should be between the size of input and output. (Heaton, 2017)

Optimizer is the function that aims to minimize the loss value by tweaking the weights of the model and *learning rate* is the rate of which the optimizer is changing the weights. Too large learning rate could result in suboptimal solution and too small could result in slow or nonexistent convergence of the model. (Brownlee, 2019)

Dropout rate is a form of regularization, which controls the number of nodes that are updated in the training process and vice versa, the number of nodes that are not updated i.e. dropped out. Essentially dropout is a technique for preventing over-fitting and preventing the model to lean on certain features. (Brownlee, 2018a; Goyal et al., 2018)

Epochs define the number of times the model will go through the whole training dataset. Selecting right number of epochs are necessary for giving most of the individual data samples a change to influence the outcome, without overfitting. Selecting too few epochs will lead to underfitting as the model cannot achieve its full learning capability. Popular technique for not having to worry about number of epochs is early stopping, where at the end of every epoch the model's performance is evaluated and saved if it beats previous best result. The training can also be stopped if the model does not improve in a specified number of epochs. (Deeplearning4j, 2019; Nicholson, 2019)

Also, an important hyperparameter for RNNs is *batch size*. Batch defines as a set of individual data samples that are fed into the training process and worked through before updating the weights. As such, batch size defines as the size of the set of data samples the model sees before updating the weights. Generally small batch sizes of e.g. 32 samples are selected for two main reasons: small noisy batches offer a regularizing effect which improves generalization and small batches fit better into memory. (Brownlee, 2018b)

For the model parameters, the following was initially selected:

- Hidden layer size: 100
- Learning rate of optimizer: 0.001
- Dropout rate: 0.5
- Number of Epochs (with early stopping): 300
- Batch size: 32

The output layer for the model is *Dense* with 1 output and activation function as *sigmoid.*

5 RESULTS

The dataset was initially split into training and testing data, where training data consisted of 80% of the original dataset and thus, the test data was 20% of the original data. The training was first commenced with formerly mentioned parameters. Also, a parameter called *return_sequences* was set to True, which indicates that the model will be trained using sequences of samples, rather than only one sample, i.e. the model produces output at each time step rather than only at the last one. This can be visualized via figure 5, where at each cell there is a hidden layer output h_t, but without *return_sequences* set as True, there would be only one output h_t at the last cell. Thus, a generator was implemented that produced random batches from the training data where each sequence of a batch consisted of 3 days of data. This means that each sample in a batch includes 3 days of data and the weights were updated after the model had seen 32 * 3 samples which calculates to 160 samples total.

The models were trained using TensorFlow via GPU (Nvidia GeForce GTX 1070). The results are mostly visualized by graphs, where the red color indicates prediction of “down” and green color indicates “up” prediction. The different models are compared by their loss values as well as accuracy. Accuracy indicates the amount of correctly predicted classes and in this problem, it is suitable metric because this problem is more closely related to short term option trading rather than long term value investing. In option trading, the cost of false negatives and false positives is equal as a call option becomes invaluable with false positive and vice versa for put option.

5.1 First model

For the first training the total amount of features was 420, which included open, high, low, close, volume, % change between high and low, % change between previous close, 200 positive word combinations, 200 negative word combinations, 10 topics with their modeled probabilities, stochastic %D, stochastic %K and momentum.

The final feature space for each day can be visualized as followed:

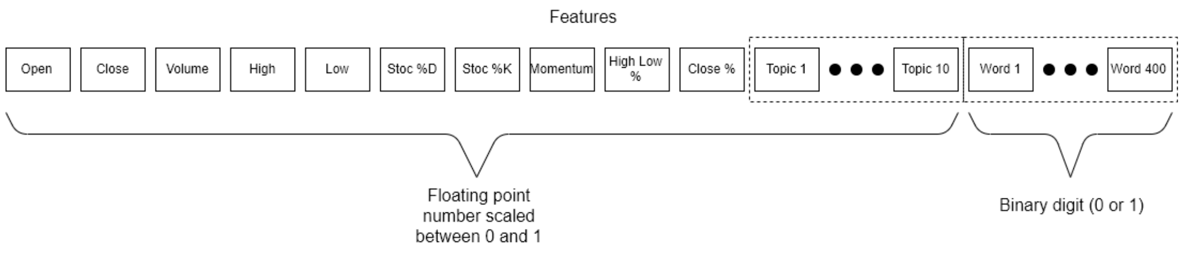

Figure 6 Final feature space for first model.

For the first model, the validation loss value did not start to converge until the training was stopped at 33 epochs due to early stopping. Since the training results were so underwhelming and it was obvious that the model is no good, the actual prediction was not performed. The results for training are visible in figure 7**.**

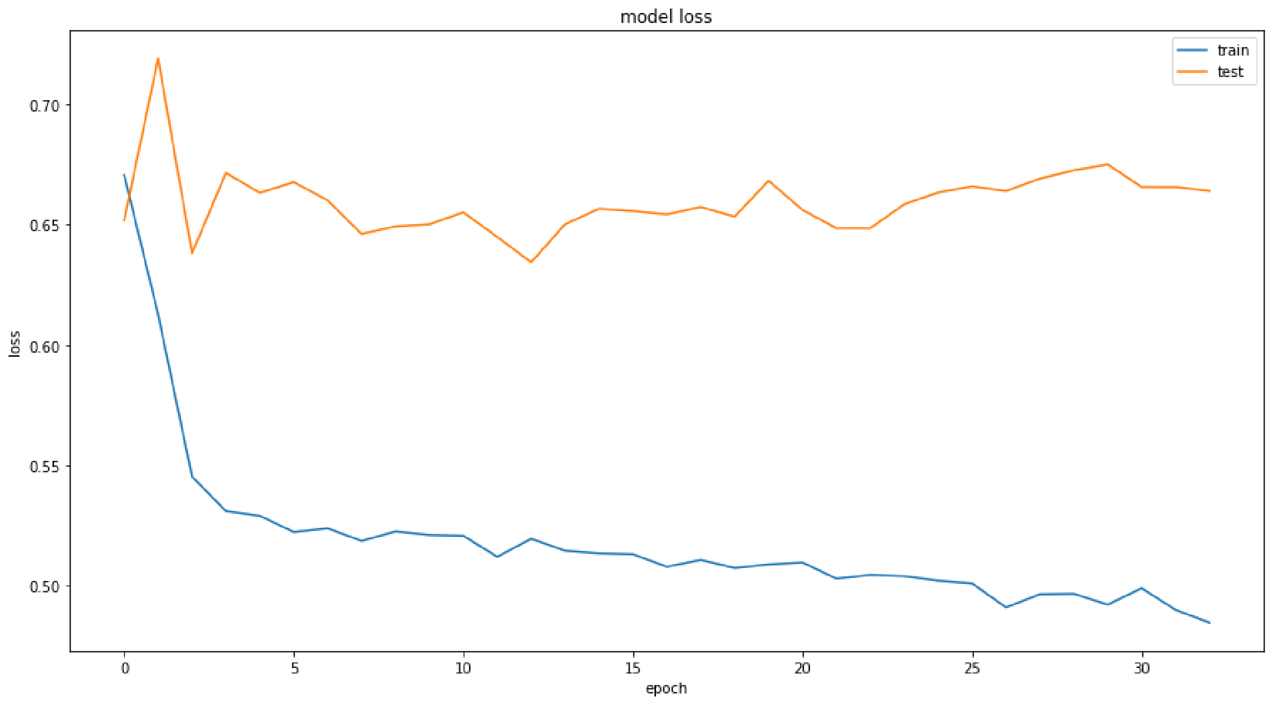

Figure 7 loss of the first model as a function of epochs with 0.001 learning rate.

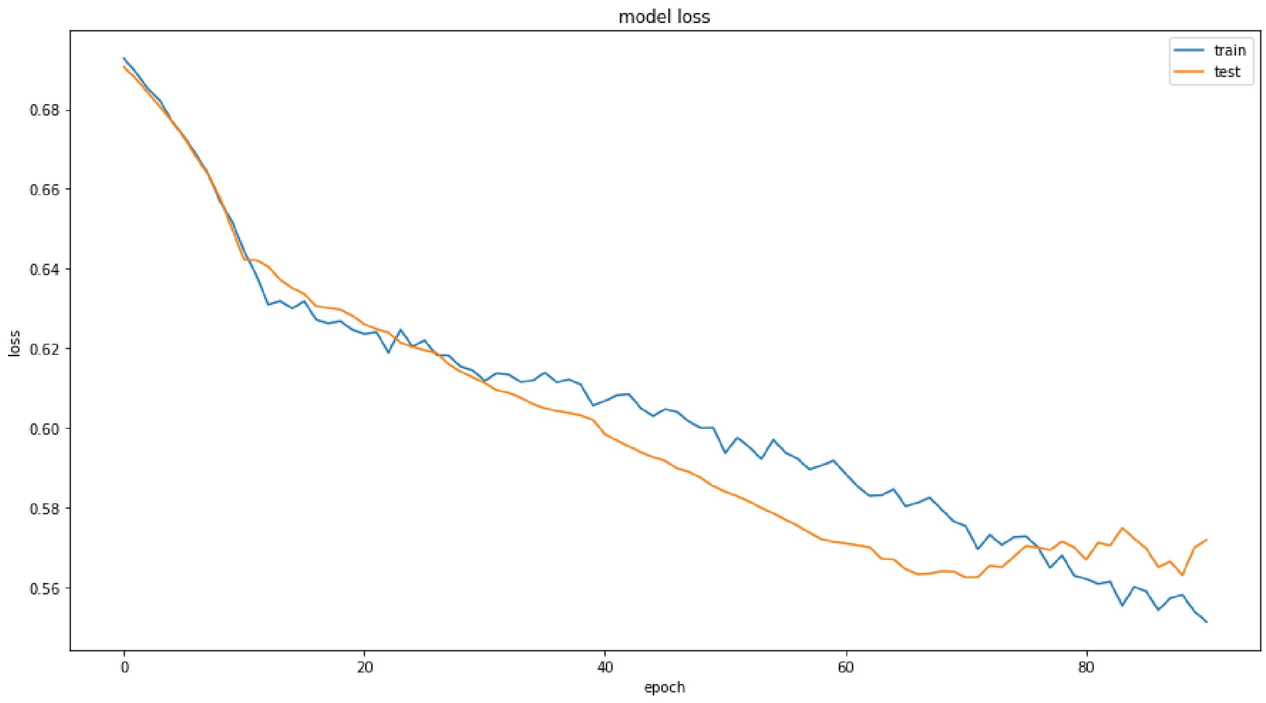

Figure 8 loss of the first model as a function of epochs with 0.0001 learning rate.

For next training, learning rate was dropped from 0.001 to 0.0001, which showed some improvement and the model started to converge. As shown in figure 8, the loss started to eradicate from the train data plot at 90 epochs and the training was stopped. The loss of the model was still quite high but showed some noticeable difference compared to previous results. Because the model did converge, the predictive results were also examined by loading the weights of the best model at epoch 73 and feeding the model with the test data, which the model had not seen before.

The results are visualized in figure 9. From the graph it is evident that the prediction leaned on the "up" value. By examining model's metrics, the accuracy of the model was defined as 0.513, which is also the percentage of positive classes in the actual data. Also, when examining the predicted probabilities of output, it was clear that the model did not have strong basis for its predictions as the predicted probabilities for positive class fluctuated between a range of 0.51-0.55.

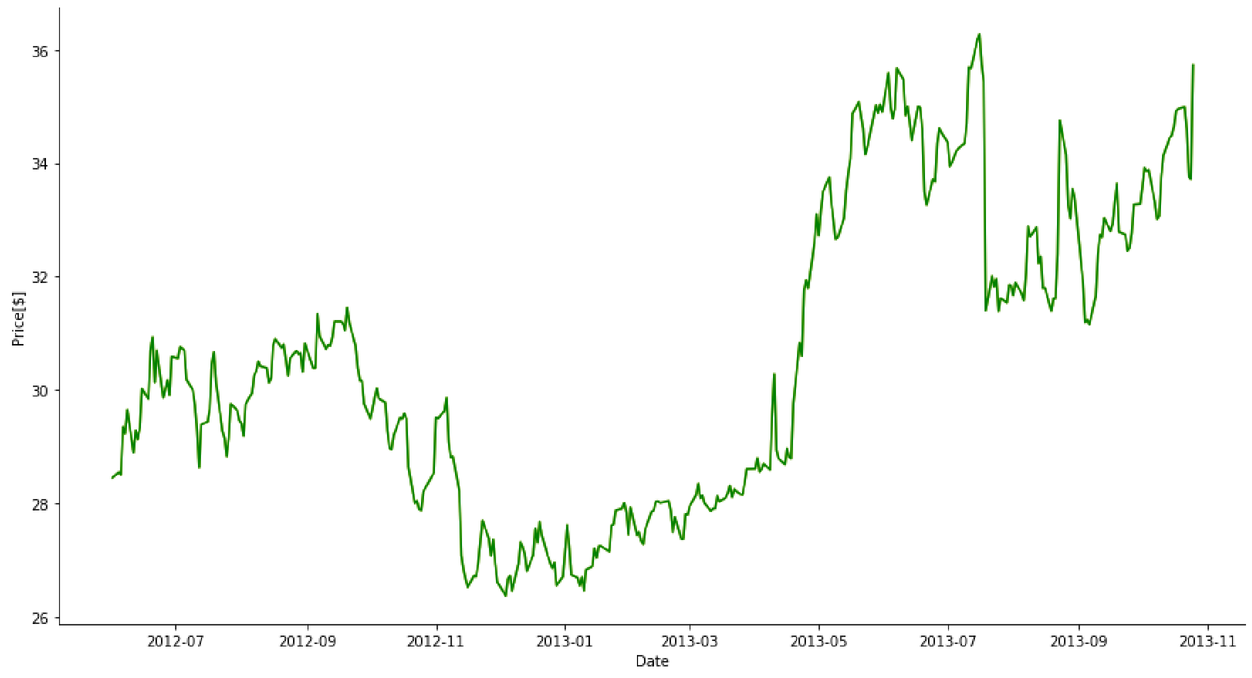

Figure 9 results of the first model.

At this point it could be argued that the results indicated that the model could not learn the problem with the current parameters and features. Thus, the options were to try different hidden layer sizes, add more layers, try different batch sizes and add more features, i.e. words for both sentiment classes.

5.2 Second model

By setting the layer size from 100 to 150 and further to 200, the loss values did not converge as much and sometimes resulted in exploding gradients as the loss function quickly increased by a multitude. The layer size was also reduced to 64 and further to 32, where the loss values did converge but the results remained the same and the model favored another class every time, resulting in a homogenous outcome. The model was also constructed using two LSTM-layers, with a hidden layer size of 100 nodes for each, which did produce differencing results. Although, the model did still favor one class over the other, the mean of the predictions was closer to 0.5. This does not however make the prediction any better and could indicate, that by making the model more complex, it is even more confused by the input. Changing the batch size and sequence length did not have any effect either.

By testing various parameters and their combinations, the last option was increasing the amount of words representing the negative and positive classes. Thus, the next step was to increase the word amount from 200 words in each class to 500 words in each class.

The new model was trained with the initial setting with one LSTM-layer with hidden layer size as 100. Learning rate was set to 0.0001, batch size as 32 and sequence length as 3. As the model converged very quickly and then exploded, the learning rate was dropped to 0.00001. This as itself did not produce different results so the hidden layer size was increased to 512. With layer size as 512, the results became more interesting as can be seen from figure 10. The accuracy of this model 0.448, was worse than others and from the graph it is seen that the predictions are highly opposite compared to the actual values. However, for binary classification problems this is better result than previous ones, since the result can be inversed. The resulting graph after inverting the results can be found in figure 11, and by looking at the graph, the results are fairly good in some segments. The accuracy for the inversed graph was calculated as 0.552. However, this does not indicate that the model is any good and there is a high change that the model won't predict new data with accuracy of 55,2%.

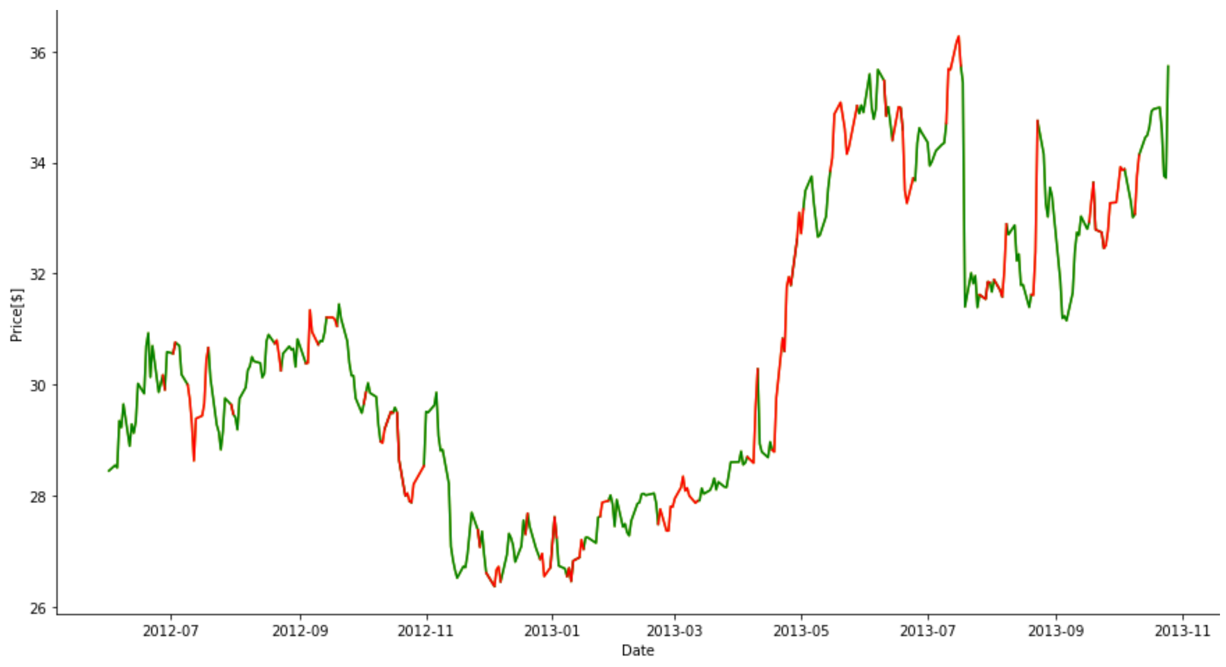

Figure 10 second model results.

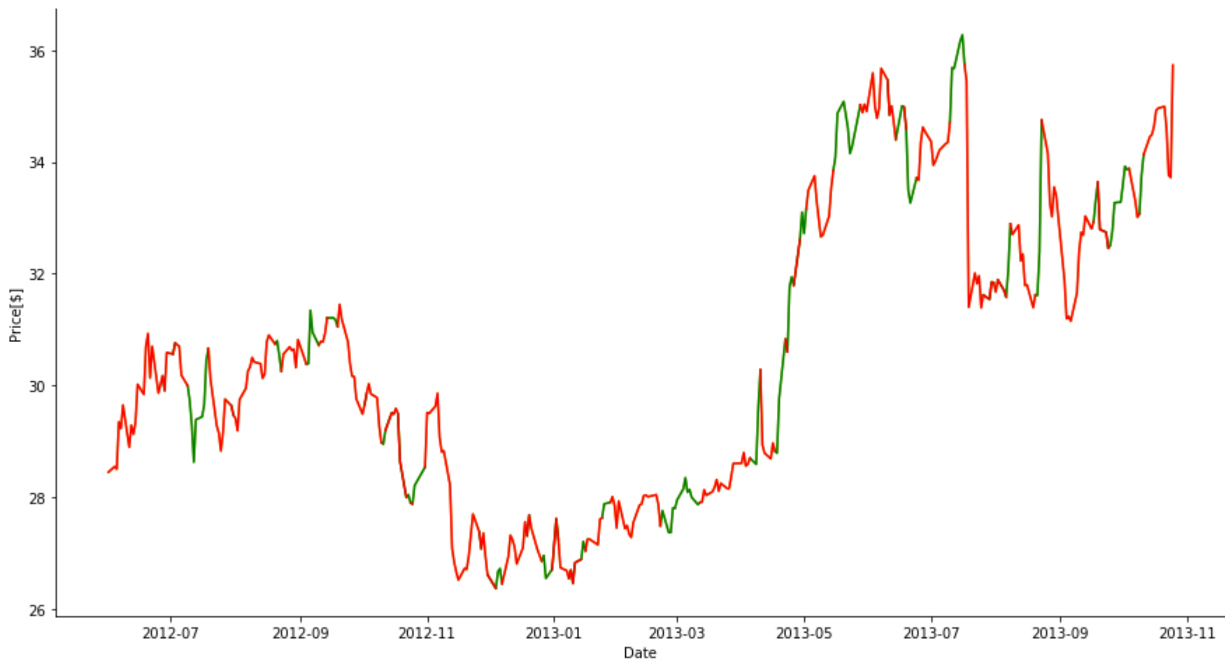

Figure 11 second model results inversed.

5.3 Third model

For the third model, the word feature size was increased from 500 for each class to 5000 for each. For this model the best results were achieved with 512 and 1024 as hidden layer size, with 0.0001 as learning rate. The models did not however have better accuracy but now the different models seemed to produce some consistency and similarity. In the following figures 12 and 13, the results are compared for two different models.

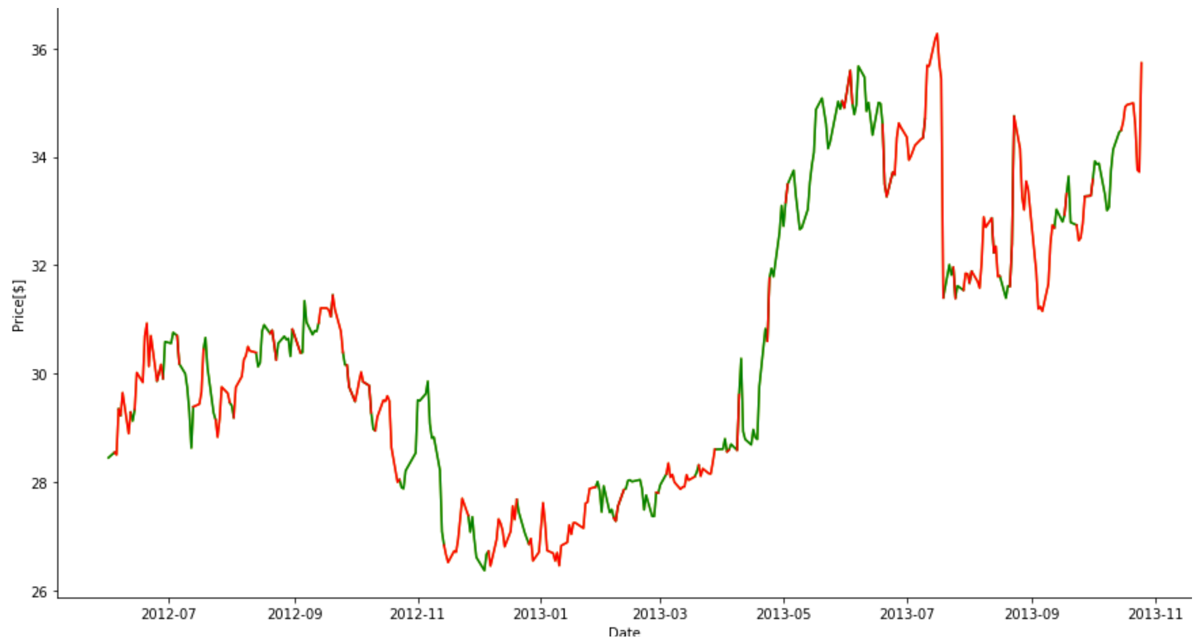

Figure 12 third model results with hidden layer size as 512.

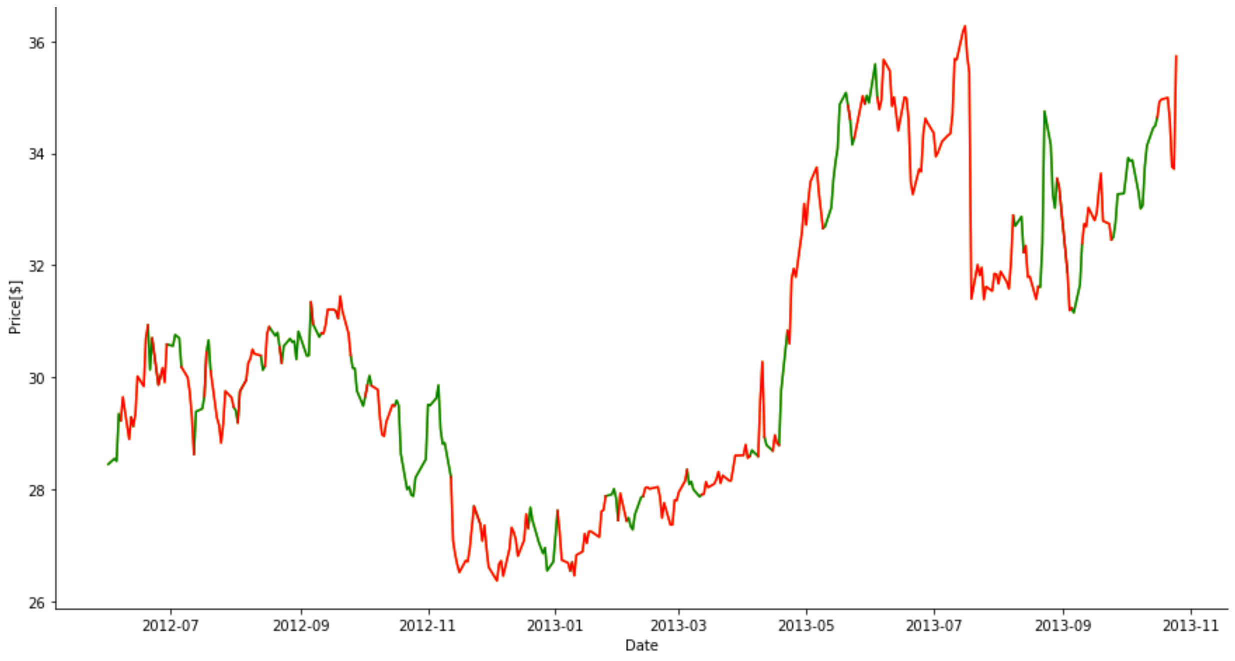

Figure 13 third model results with hidden layer size as 1024.

This could indicate that the model was able to learn in some level, but a more likely scenario is that the results were just pure coincidence. To test this hypothesis, the models were trained again multiple times and as predicted, the results were mostly not the same or even similar. The reason for this randomness could be that, because the weights are randomly initialized when the model is defined and thus, the output of training in this case is also different because the model cannot actually generalize the input data. This means that the model can only figure that the best method for minimizing error is by generating predictions in between the two classes and the random weight initialization controls on how the final individual predictions fluctuate around 0.5. Also, one notable matter is the fact that the loss value for the trained model would always be about 0.690-0.695. This means that though some models seem to be better than others, the models are not actually that far apart, and this strengthens the perception of randomness.

5.4 Fourth model

As the research questions regarded whether the textual features would have any impact on predicting the stock markets, a model using only market data was also constructed. As the amount of features was now only 10, the amount of hidden layers had to be on a scale of 9 to 1, per (Heaton, 2017). The models were trained multiple times with different parameters and each model did converge quite well on training phase. Also, the results were very similar compared to each other. Compared to the text-based models however, the models did favor one class over the another every time and the difference between predicted probability was noticeable larger. Where the text-based models sentiment value fluctuated on each side of 0.5, the market data -based model had some values either at about 0.55 or at about 0.45. Also, the losses of the trained models were generally higher for the models using only market data, at about 0.700-0.715 range, whereas the text-based model's loss was around 0.690-0.695. The difference is very small, but the values were consistently calculated on the specified ranges and considering how inaccurate both models are, it does indicate some difference between the models.

A model that used the topic data along with the market data was also constructed, to examine if the topics have correlation with the movement of the stock markets. The results did not differ from the ones achieved with using only market data in regards of the model always predicting only one class, and the predicted probabilities difference seemed to only increase. The loss ranged from 0.705 to 0.755, which indicates that the model had gone worse. Also, the accuracy was always either 0.487 or 0.513, depending on whether the model predicted only negative or positive class.

Since the topics seemed to only worsen the outcome, for the final model they were discarded, and the model was trained by using same features and parameters as in model three. The results were similar, with the loss ranging from 0.690 to 0.695 and accuracy from 0.508 to 0.544.

5.5 Discussion

The results somewhat indicate that the word combinations achieved through BNS do have some positive effect on the predictions. However, it is hard to actually tell if this happens because of the features or because how the LSTM works with such high number of features. The randomness of the results was not necessarily surprising for such a complex problem and in this case, would probably add to the Random Walk and Efficient Market hypotheses.

According to market efficiency, the public information is almost instantly included in the stock prices and if that is truly the case, a better approach on predicting stock markets would be to use intraday market data which would be more appropriate on capturing the sentiment of single news published during the market hours. In this work the approach was to use all of the news published during market hours to predict the closing price of the current day. In the methods used in this work, the time interval from publication of certain news article and the closing price can be relatively large and the effect of the news article will with no doubt get lost by some degree in the noise produced by other information during the day. However, the problem of acquiring intraday stock market data can be tricky and most, if not all of the complete datasets are behind paywalls.

One central problem on this work and in deep learning as a whole is the need of large datasets which were very scarce on financial news as the topic. The dataset used in this work contained some 110000 news articles on a range of about 7 years. Because the acquired stock market data was only by the day and thus, the articles had to be fitted on the same scale, so each day's separate articles were merged into one. Now the dataset used in the modeling contained only 1784 datapoints, which is low amount especially for recurrent architectures. Even though the suitability of the financial news dataset was reviewed as fitting for the problem when examining the results of (Ding et al., 2014), it is notable that in the mentioned work, the researchers used also another dataset consisting of over 400 000 news articles. This dataset was not however available anymore due to copyright issues. Thus, 110000 news articles can be too small for a problem of this complexity and when the articles are divided for each day, it calculates to about 60 articles per day. However, only 5 days of the week are trading days and added with some holidays the total amount of trading days in a year calculates as 252. This means that only about 70% of the articles could be directly mapped with the market data of the publication date. The last 30% of the articles had to be included either on the next day if the publication time was past market hours or worse, included to the next weekday over weekend. This could have had a negative effect on finding accurate sentiments through BNS, since every Monday was particularly littered with the news information and all the news produced over weekend would be classified based on the Monday's market movement.

One matter affecting the outcome could also be that in this work all of the news's were from the same news provider Reuters. It could be argued that because the news articles are from the same provider, it is harder to grasp the actual meaningful information from the variety of published articles. This is because it is unlikely that some single article published by one provider will affect the markets but if the same topic occurs in multiple news from various providers, the likelihood of market affection is greatly increased. The results could also be better if the financial news included targeted articles, which would e.g. include the to be predicted company's name. In this work the articles consisted of general financial news that the named news provider had published on each day and it would include news about the observed company only occasionally.

Even though most of the improvement to be made in future work targets on all levels of feature modeling, the deep learning model could also be enhanced. For example, adding so called attention layers along with the LSTM architecture has been proven to capture the most relevant time steps and thus increase the prediction accuracy. (Li et al., 2018)

6 SUMMARY

The research on using machine learning approaches for predicting stock markets has gained popularity in the recent years. There is a plethora of different approaches, methods and models for achieving results on the matter. In this work the approach was to use news data alongside with market data for predicting stock market movements. From the variety of different methods for extracting news data for machine learning purposes, the ones used in this work were 2-word combinations selected with BNS and LDA alongside with LSTM neural network. These methods were selected mainly because they were not used with LSTM networks within literature and had achieved good results when used with different machine learning models other than neural networks. The aim of the work was to examine the performance of the selected methods alongside LSTM networks as well as compare the results with the Efficient Market hypothesis.

In the research, the models were trained multiple times using different features to examine the effect of different input. It was concluded that the results for every trained model were inaccurate and the models could not produce reliable results. However, the 2-word combinations selected with BNS-method did produce different results with a slightly lesser loss value. Nonetheless, the results were still unreliable, and they presented random variation between the models trained with same input and parameters. This was concluded as the model's inability to generalize the input data.

From the research it can be said that due to the complexity of stock markets, predicting them requires a lot more subtle approach than the rather general one implemented in this work. The inaccurate results could be explained with insufficient news data and unsuitability of the approach or model and its selected parameters. The used news dataset could be too small for such a complex problem and because it contained news from only one provider, it can be insufficient on grasping the overall sentiment of global economy. The approach of predicting stock movements on a scale of days could also be wrong and when looking at Efficient Market hypothesis, more subtle approach of intraday predictions would suit the problem better as it claims that public news reflects on the markets almost instantly. One factor is also the difficulty of training LSTM networks where the correct parameter and layer selection is crucial.

REFERENCES

Aggarwal, C.C., Zhai, C., 2012. Mining Text Data. Springer US.

Anaconda, Inc, 2019. Anaconda [WWW Document]. Anaconda. URL https://www.anaconda.com/ (accessed 12.12.19).

Baltazar, G., 2018. CPU vs GPU in Machine Learning [WWW Document]. URL https://blogs.oracle.com/datascience/cpu-vs-gpu-in-machine-learning (accessed 12.13.19).

Barron’s, 2019. Microsoft’s Big Weighting Makes It Sole Stock in S&P 500 ‘4% Club’ [WWW Document]. URL https://www.barrons.com/articles/microsofts-weighting-stock-sp-500-51558115949 (accessed 12.29.19).

Berthold, M.R., Borgelt, C., Höppner, F., Klawonn, F., 2010. Guide to Intelligent Data Analysis: How to Intelligently Make Sense of Real Data. Springer Science & Business Media.

Blei, D.M., Ng, A.Y., Jordan, M.I., 2003. Latent Dirichlet Allocation. J. Mach. Learn. Res. 3, 993–1022.

Brockwell, P.J., Davis, R.A., 2016. Introduction to Time Series and Forecasting, 3rd ed, Springer Texts in Statistics. Springer International Publishing. https://doi.org/10.1007/978-3-319-29854-2

Brownlee, J., 2019. Understand the Impact of Learning Rate on Neural Network Performance. Mach. Learn. Mastery. URL https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/ (accessed 12.12.19).

Brownlee, J., 2018a. A Gentle Introduction to Dropout for Regularizing Deep Neural Networks. Mach. Learn. Mastery. URL https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/ (accessed 12.13.19).

Brownlee, J., 2018b. Difference Between a Batch and an Epoch in a Neural Network. Mach. Learn. Mastery. URL https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/ (accessed 12.13.19).

Dang, M., Duong, D., 2016. Improvement methods for stock market prediction using financial news articles, in: 2016 3rd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS). Presented at the 2016 3rd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS), pp. 125–129. https://doi.org/10.1109/NICS.2016.7725636

Deeplearning4j, 2019. Early Stopping | Deeplearning4j [WWW Document]. URL https://deeplearning4j.org/docs/latest/deeplearning4j-nn-early-stopping (accessed 12.13.19).

Ding, X., Zhang, Y., Liu, T., Duan, J., 2014. Using Structured Events to Predict Stock Price Movement: An Empirical Investigation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Presented at the EMNLP 2014, Association for Computational Linguistics, Doha, Qatar, pp. 1415–1425. https://doi.org/10.3115/v1/D14-1148

Fama, E.F., 1995. Random Walks in Stock Market Prices. Financ. Anal. J. 51, 75–80. https://doi.org/10.2469/faj.v51.n1.1861

Fama, E.F., 1970. Efficient Capital Markets: A Review of Theory and Empirical Work*. J. Finance 25, 383–417. https://doi.org/10.1111/j.1540-6261.1970.tb00518.x

Fischer, T., Krauss, C., 2018. Deep learning with long short-term memory networks for financial market predictions. Eur. J. Oper. Res. 270, 654–669. https://doi.org/10.1016/j.ejor.2017.11.054

Forman, G., 2003. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. J. Mach. Learn. Res. 3, 1289–1305.

Goyal, P., Pandey, S., Jain, K., 2018. Deep Learning for Natural Language Processing : Creating Neural Networks with Python. Apress.

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., Schmidhuber, J., 2009. A Novel Connectionist System for Unconstrained Handwriting Recognition. IEEE Trans. Pattern Anal. Mach. Intell. 31, 855–868. https://doi.org/10.1109/TPAMI.2008.137

Hagenau, M., Liebmann, M., Neumann, D., 2013. Automated news reading: Stock price prediction based on financial news using context-capturing features. Decis. Support Syst. 55, 685–697. https://doi.org/10.1016/j.dss.2013.02.006

Heaton, J., 2017. The Number of Hidden Layers [WWW Document]. Heaton Res. URL https://www.heatonresearch.com/2017/06/01/hidden-layers.html (accessed 12.13.19).

Hochreiter, S., Schmidhuber, J., 1997. Long Short-Term Memory. Neural Comput. 9, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Huang, Z., Xu, W., Yu, K., 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. ArXiv150801991 Cs.

Huynh, H.D., Dang, L.M., Duong, D., 2017. A New Model for Stock Price Movements Prediction Using Deep Neural Network, in: Proceedings of the Eighth International Symposium on Information and Communication Technology, SoICT 2017. ACM, New York, NY, USA, pp. 57–62. https://doi.org/10.1145/3155133.3155202

Kao, A., Poteet, S.R. (Eds.), 2007. Natural Language Processing and Text Mining. Springer London, London. https://doi.org/10.1007/978-1-84628-754-1

Kim, K., 2003. Financial time series forecasting using support vector machines. Neuro-computing, Support Vector Machines 55, 307–319. https://doi.org/10.1016/S0925-2312(03)00372-2

Li, H., Shen, Y., Zhu, Y., 2018. Stock Price Prediction Using Attention-based Multi-Input LSTM, in: Asian Conference on Machine Learning. Presented at the Asian Conference on Machine Learning, pp. 454–469.

Loughran, T., Mcdonald, B., 2011. When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks. J. Finance 66, 35–65. https://doi.org/10.1111/j.1540-6261.2010.01625.x

Mittermayer, M.-, 2004. Forecasting Intraday stock price trends with text mining techniques, in: 37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of The. Presented at the 37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the, pp. 10 pp.-. https://doi.org/10.1109/HICSS.2004.1265201

Nelson, D.M.Q., Pereira, A.C.M., Oliveira, R.A. de, 2017. Stock market's price movement prediction with LSTM neural networks, in: 2017 International Joint Conference on Neural Networks (IJCNN). Presented at the 2017 International Joint Conference on Neural Networks (IJCNN), pp. 1419–1426. https://doi.org/10.1109/IJCNN.2017.7966019

Nguyen, D., 2019. duynht/financial-news-dataset.

Nicholson, C., 2019. Neural Network Tuning [WWW Document]. Pathmind. URL http://pathmind.com/wiki/neural-network-tuning (accessed 12.13.19).

Persio, L.D., Honchar, O., 2016. Artificial Neural Networks Approach to the Forecast of Stock Market Price Movements. Int. J. Econ. Manag. Syst. 01.

Sebastiani, F., 2002. Machine Learning in Automated Text Categorization. ACM Comput Surv 34, 1–47. https://doi.org/10.1145/505282.505283

Siah, K.W., Myers, P., 2015. Stock Market Prediction through Technical and Public Sentiment Analysis [WWW Document]. URL https://www.semanticscholar.org/paper/Stock-Market-Prediction-through-Technical-and-Siah-Myers/528ae8a460bb3afb596e71f5a1c6ad83ad3f284c (accessed 10.22.19).

Tetlock, P.C., Saar-Tsechansky, M., Macskassy, S., 2008. More Than Words: Quantifying Language to Measure Firms' Fundamentals. J. Finance 63, 1437–1467. https://doi.org/10.1111/j.1540-6261.2008.01362.x

Yang, Y., Pedersen, J.O., 1997. A Comparative Study on Feature Selection in Text Categorization, in: Proceedings of the Fourteenth International Conference on Machine Learning, ICML '97. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 412–420.

Zhai, Y., Hsu, A., Halgamuge, S.K., 2007. Combining News and Technical Indicators in Daily Stock Price Trends Prediction, in: Liu, D., Fei, S., Hou, Z., Zhang, H., Sun, C. (Eds.), Advances in Neural Networks – ISNN 2007, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 1087–1096.

Zhou, C., Sun, C., Liu, Z., Lau, F.C.M., 2015. A C-LSTM Neural Network for Text Classification. ArXiv151108630 Cs.

Zhuge, Q., Xu, L., Zhang, G., 2017. LSTM Neural Network with Emotional Analysis for Prediction of Stock Price.