TWITTER CEO SENTIMENT ANALYSIS USING TRANSFORMERS: PREDICTING STOCK MARKET CHANGES

MATHIJS GEELEN

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
AT THE SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
OF TILBURG UNIVERSITY

STUDENT NUMBER

2002363

COMMITTEE

dr. J.S. Olier Jauregui
dr. G. Spigler

LOCATION

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science &
Artificial Intelligence
Tilburg, The Netherlands

DATE

May 20, 2021

ACKNOWLEDGMENTS

I would like to thank my supervisor dr. J.S. Olier Jauregui for his time and help during the last couple of months. Without his supervision this thesis would not have been possible.I would also like to thank my sister, for helping me with the structuring and layout of this thesis.Lastly, I would like to thank my girlfriend for her feedback and mostly her support, as well as my parents for their support.

TWITTER CEO SENTIMENT ANALYSIS USING TRANSFORMERS: PREDICTING STOCK MARKET CHANGES

MATHIJS GEELEN

ACRONYMS

ALBERT A Lite Bert. 15

BERT Bidirectional Encoder Representations from Transformers. 14

CEO Chief Executive Officer. 5

DeBERTa Decoding-enhanced BERT with Distangled Attention. 16
DL Deep Learning. 13

EMA Exponential Moving Average. 24
EMH Efficient Market Hypothesis. 4

LSTM Long Short-Term Memory. 4

MHA Multi-Headed Attentio. 14
ML Machine Learning. 7
MLM Masked Language Modelling. 15
MSE Mean Squared Error. 24

NN Neural Network. 9
NSP Next Sentence Prediction. 15

PLM Permutation Language Modelling. 16
PTMs Pre-trained Models. 14

RNN Recurrent Neural Network. 9
RoBERTa Robustly optimized BERT pretaining approach. 16
RSI Relative Strength Index. 24

S&P Standard & Poors. 5
SMA Simple Moving Average. 24
SOP Sentence-Order Prediction. 16
STOA State-Of-The-Art. 8
SVM Support Vector Machine. 7

T5 Text-to-Text Transfer Transformer. 17

LIST OF FIGURES

LIST OF TABLES

CONTENTS

Abstract

This study investigates the importance of CEO reputation on stock price. A major driver of stock price is corporate reputation. The CEO accounts for a large part of the corporate reputation, and thus it may be possible to predict stock price based on CEO reputation. Seven CEOs of the top S&P 500 firms were analysed. CEO reputation was measured using sentiment analysis on daily Twitter data using various Transformer based models. To determine the influence of CEO reputation, Long Short-Term Memory (LSTM) containing Twitter information was used to predict stock price, as well as McNemar's test. LSTM without Twitter information served as a baseline. The results show that including Twitter information does lead to a better LSTM model for stock price prediction, but there is no statistical substantiation for this increase. This research concludes that there is no definitive proof that CEO reputation is a strong predictor of stock price.

1 INTRODUCTION

Stock price forecasting has been a topic of interest in both business and academia for decades. If one could accurately predict stock market movements, the possible economic profits would be endless. Creating an accurate stock price forecasting model, however, has proven to be very challenging due to the random nature and high level of noise of the stock market (Malkiel & Fama, 1970; Oliveira & Meira, 2006; Zhang et al., 2018).

Early work in stock price prediction was based on the random walk theorem and Efficient Market Hypothesis (EMH) (Bollen, Mao, & Zeng, 2011; Fama, 1965). The random walk theorem states that past prices are not reflective of the future. Whether a stock would go up or down was deemed to be completely uncorrelated with past prices. In recent years, however, it has been proven that stock prices do not follow a completely random walk, and thus are future prices dependent on past prices (Gallagher & Taylor, 2002; Kavussanos & Dockery, 2001).

The EMH states that stock price changes are largely driven by new information. With the introduction of the internet, there is a continuous stream of information that is readily available, be it through news articles, social media, or online financial reports. Text mining is the field that analyses this online information. The first attempt at predicting stock prices using text mining was by Wuthrich et al. (1998). The core idea of text mining for stock price prediction is always the same; automatically extract information from unstructured documents and correlate this information with stock prices.

In recent years text mining has been largely used to determine the mood of society regarding a topic. Many researchers have shown that

there is a correlation between online sentiment and stock price (Bollen et al., 2011; Kordonis, Symeonidis, & Arampatzis, 2016; Li, Xie, Chen, Wang, & Deng, 2014; Nguyen & Shirai, 2015; Pagolu, Reddy, Panda, & Majhi, 2016; Qasem, Thulasiram, & Thulasiram, 2015; Rao, Srivastava, et al., 2012). By analysing how the public feels regarding a company, it is possible to predict the short-term trend of the respective stock price.

The feelings of a large public can be analysed using sentiment analysis. Sentiment analysis, or opinion mining, is the field of finding the opinion of authors about specific entities, through self-automated tools (Chaudhuri, 2019; Feldman, 2013). A popular platform for gathering information for sentiment analysis is Twitter (Baziotis, Pelekis, & Doulkeridis, 2017; Bollen et al., 2011; Cakra & Trisedya, 2015; Guo & Li, 2019; Ismail, Harous, & Belkhouche, 2016; Kordonis et al., 2016; Pant, Neupane, Poudel, Pokhrel, & Lama, 2018; Pota, Ventura, Catelli, & Esposito, 2021; Rao et al., 2012; Sprenger, Tumasjan, Sandner, & Welpe, 2014; Valencia, Gómez-Espinosa, & Valdés-Aguirre, 2019). Twitter is a social media platform where people from all over the world can share their emotions, activities, opinions and observations about any subject. Communication on Twitter goes via so-called "tweets". A tweet is a short message composed of up to 140 characters, containing a combination of text, videos, and photos. Twitters massive amount of data represents a relevant source for gathering people's views and opinions (Smailović, 2015).

It is clear from the literature that an organization's reputation is closely related to stock price (Black, Carnes, & Richardson, 2000; Kossovsky, Greenberg, & Brandegee, 2012; Vergin & Qoronfleh, 1998). An important driver of corporate reputation is the Chief Executive Officer (CEO). Gaines-Ross (2000) showed that the CEO can represent up to 45% of the company's overall reputation. Since the reputation of the CEO is a major driver of corporate reputation, it might be possible to predict stock price based on CEO reputation.

This research aimed to discover whether CEO reputation is indicative of stock price. Sentiment analysis on Twitter data was used to determine the feelings of a large public regarding a CEO. This sentiment was then used to predict the stock price trend. CEOs from the top eight Standards & Poors (S&P) 500 organizations were analysed.

Reputation cannot be directly measured. This thesis assumes that using sentiment analysis on a large corpus of Tweets mentioning a CEO is representative of his/her reputation. Reputation in this thesis, refers to the average sentiment regarding a CEO.

The main question this thesis will address is:

> *To what extend is CEO reputation, measured using sentiment analysis, predictive of stock price changes?*

Sub-questions that this thesis will address are:

RQ1 *To what extent is the amount of internet traffic predictive of stock price movements?*

RQ2 *To what extend is there a delay between Twitter information and its reflection in stock price?*

RQ3 *To what extent do Transformer models improve sentiment analysis classification?*

RQ3 *To what extent does ensemble of Transformer models outperform individual classifiers?*

2 RELATED WORK

This thesis aimed to predict stock price trend using CEO reputation measured using sentiment analysis. Stock price prediction coupled with sentiment analysis has been a topic of interest among many authors. Bollen et al. (2011) were the first to show a correlation between Twitter sentiment and stock price. They used past prices as well as sentiments derived from tweets as inputs in a Fuzzy neural network to predict the movement of the Dow Jones Industrial Average index. They concluded that there is a strong correlation between Twitter sentiment and stock price. Mittal and Goel (2012) replicated Bollen et al. their experiment, confirming their results. Dickinson, Hu, et al. (2015) as well as Pagolu et al. (2016), used Pearson correlation to determine the relationship between public sentiment and stock price. Both publications found a strong correlation between public sentiment and stock price. Cakra and Trisedya (2015) used linear regression to predict stock price. They concluded that using historic prices, as well as sentiment polarity as inputs performs best. Guo and Li (2019) used Twitter data to compute a sentiment score. They then used this sentiment score in a linear regression to predict the following days stock price.

Many authors used support vector machine (SVM) to predict stock price. Sprenger et al. (2014) used SVM to show that organizations in the S&P top 100 closely follow their Twitter reputation. Kordonis et al. (2016) continued on the work of Sprenger et al., showing that the same conclusion holds for sixteen large technological firms. Li et al. (2014) found that including sentiment polarity increases the accuracy of SVM for stock price prediction. Nguyen and Shirai (2015) showed an increase of 6% in predictive power when including sentiment polarity over using only historic prices as inputs in an SVM. Kalyani, Bharathi, Jyothi, et al. (2016) tested various machine learning (ML) methods and ultimately concluded that SVM performed best. Batra and Daudpota (2018) analysed the relationship between public opinion and Apple's stock price. Their results showed that SVM can predict the relation between sentiment polarity, historic prices, and Apple's stock price.

In recent years long short-term memory (LSTM) has become the most common and successful method for stock price prediction. Y. Liu, Qin, Li, and Wan (2017); Souma, Vodenska, and Aoyama (2019); Tsantekidis et al. (2017); Valencia et al. (2019) showed that LSTM outperforms SVM for stock price prediction. S. Das, Behera, Rath, et al. (2018) concluded that the ability of LSTM to store past information, and thus stock price trend, makes it most suited for stock price prediction. Pant et al. (2018) used LSTM to predict bitcoin price. Jin, Yang, and Liu (2019) showed that LSTM with attention mechanism outperformed regular LSTM for stock

price prediction.

No prior research was found on the influence of CEO reputation on stock price. Direct comparison between prior research and this research is therefore difficult. Another difficulty is the difference in timeframes and different stock data. Prior stock price trend prediction results have been published by Nelson, Pereira, and de Oliveira (2017), who obtained an accuracy of 54.3% in stock price trend prediction using LSTM, Wen, Li, Zhang, and Chen (2019), who obtained an accuracy of 55.9% for SP 500 stock price prediction, and Picasso, Merello, Ma, Oneto, and Cambria (2019), who obtained an 80% return on their hypothetical portfolio, traded using LSTM predictions.

Most of the above listed authors use very simple sentiment analysis methods. State-of-the-art (STOA) sentiment analysis methods result in better sentiment predictions, which in turn may lead to better stock price trend prediction. This thesis will test various STOA sentiment analysis methods for sentiment analysis. Further, this thesis will determine whether CEO reputation is a predictor of stock price.

3 METHOD

In this section, the LSTM model which was used for stock price prediction, as well as McNemar's test, which was used to determine the significance of the LSTM results, will be explained. First recurrent neural network (RNN), on which LSTM is based, will be introduced shortly.

3.1 *Recurrent Neural Network and Short-Term Memory*

What sets recurrent neural networks (RNN) apart from traditional neural networks (NN) is that RNN allows information to persist. Previous information might be connected to the present task, and traditional NNs are not able to take this previous information into account. RNN uses previous predictions as input for the present task and is thus able to use past information to predict the future. Non-RNN models only use snapshots of the information at time t to predict targets at time $t + \delta$. These models ignore sequential relations between time intervals Li, Wu, and Wang (2020).

RNN has a very short memory and can only learn short-term dependencies (Cliche, 2017). Stock price prediction might require remembering historic information from many predictions ago. This is where LSTM comes in. LSTM networks are a kind of RNN that is capable of learning long and short-term dependencies. The core concept of LSTM's is the cell state. Cell state acts as the memory of the network, in which past predictions are saved. The cell state only carries relevant information. Information gets added or removed to the cell state via gates. These gates use sigmoid activation functions to map values between 0 and 1. There are different gates in an LSTM; forget gate, input gate and output gate. These gates are formally defined as (Cliche, 2017):

$$f_t = \sigma\left(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f\right) \tag{1}$$
$$i_t = \sigma\left(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i\right) \tag{2}$$
$$o_t = \sigma\left(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o\right) \tag{3}$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh\left(W_c \cdot x_t + U_c \cdot h_{t-1} + b_c\right) \tag{4}$$
$$h_t = o_t \circ \tanh(c_t) \tag{5}$$

where f_t is the forget gate, i_t the input gate, o_t the output gate, c_t the cell state, h_t the prediction (regular hidden state), σ the sigmoid function, $\circ$ the Hadamard product, and x_t a matrix containing the sentiment polarity and stock price. The parameter set $\{U_f, U_j, U_o, U_c, W_f, W_j, W_o, W_c\}$ correspond to the weight matrices of different gates and $\{b_f, b_j, b_o, b_c\}$ corresponds to the bias terms of the different gates (Cliche, 2017; Jin et al., 2019). At time t, given the input vector x_t, and the previous hidden state h_{t-1}, the LSTM

calculates the hidden state h_t.

The forget gate decides which information is kept and which is forgotten (Phi, 2018). The previous hidden state and new information are aggregated and passed through a sigmoid function. Values close to 1 are kept and values close to 0 are forgotten. Figure 1 explains the workings of the forget gate using diagram notation.

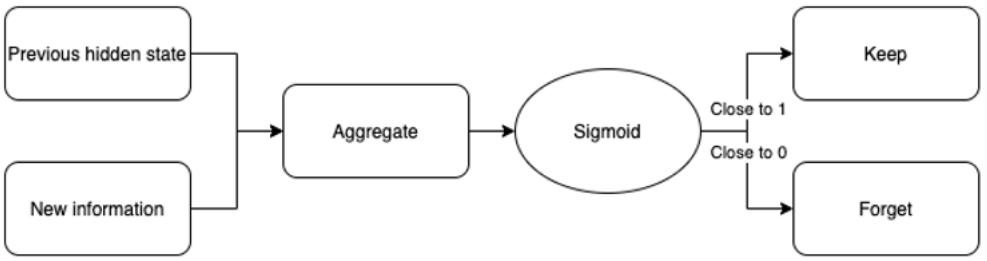

Figure 1: Graphical representation forget gate LSTM

The input gate updates the cell state (Phi, 2018). Again, the previous hidden state and new information are aggregated and passed through a sigmoid. These aggregated values are also passed through a tanh function. The outputs of the sigmoid and tanh are multiplied. The sigmoid function is used to distinguish important features from not important. Not important values are close to 0, and thus is the product of sigmoid and tanh close to 0. Figure 2 explains the workings of the input gate using diagram notation.

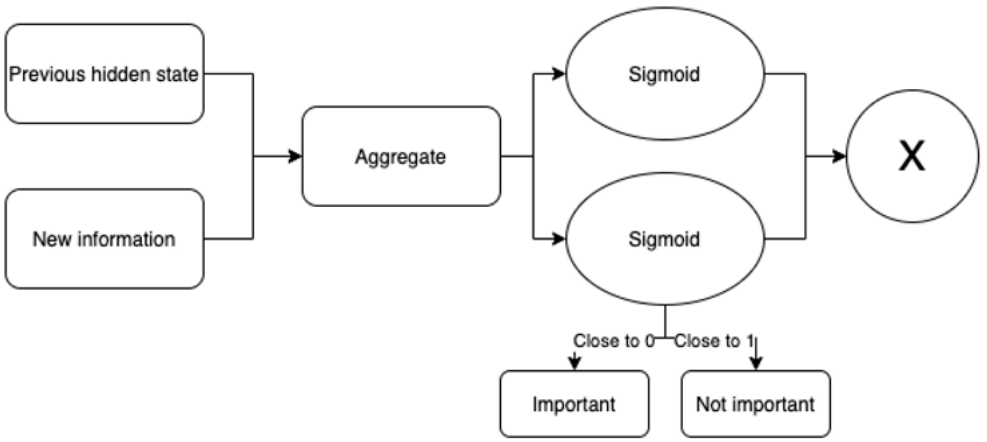

Figure 2: Graphical representation input gate LSTM

The above two gates are needed to update the cell state. The forget gate determines which information is kept, and the input gate updates the

cell state. Figure 3 explains the process of updating the cell state using diagram notation.

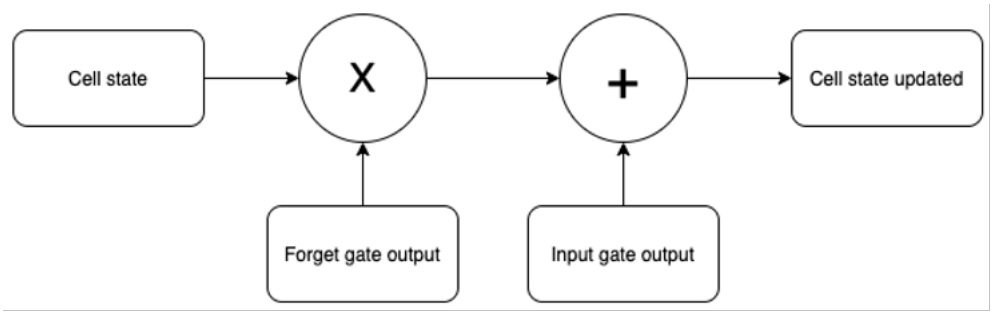

Figure 3: Graphical representation cell state update LSTM

The final step is the output gate. The output gate is used for new predictions (Phi, 2018). As noted above, LSTM uses new and past information for predictions. First, the previous hidden state and new information are passed through a sigmoid. Next, the updated cell state is passed through a Tanh. Both outputs are multiplied and result in a prediction. Figure 4 explains the workings of the output gate using diagram notation.

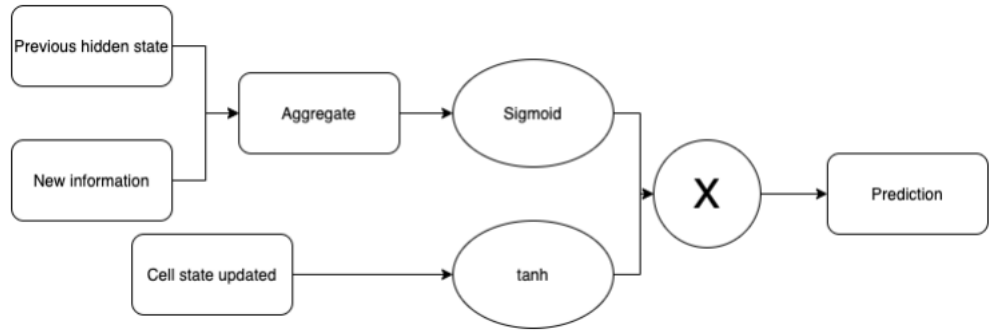

Figure 4: Graphical representation output gate LSTM

3.2 *McNemar's Test*

The McNemar's test (McNemar, 1947) is used to determine whether including sentiment, volume, or both, results in a statistically different LSTM model. The idea is that by including sentiment, volume, or both, as a variable, the model differently learns how to map inputs to output. The McNemar's test is used to test the above hypothesis and determine whether the difference in models is statistically significant.

The McNemar's test is a statistical test on a 2x2 classification table (table 1) to determine whether there is a statistical difference between two

	Model 1 up	Model 1 down	Row total
Model 2 up	a	b	$a+b$
Model 2 down	c	d	$c+d$
Column total	$a+c$	$b+d$	N

Table 1: McNemar's classification table

models (Lu, 2010). It assumes dichotomous data, meaning that it requires nominal data, or categorical data, of two classes. In this research the dichotomous data are the trend predictions of the LSTM model, being up and down. McNemar's test does not require the two classes to be from different samples, as many other statistical tests do. Rather it is used to test the statistical difference of two models on the same set of data.

The null hypothesis is that two models are the same, or rather the probability of each model's outcome is the same. The alternative hypothesis is that two models are different, or rather that the probability of each model's outcome is different. More formally:

$$p_a + p_b = p_a + p_c \tag{6}$$
$$p_b + p_d = p_c + p_d \tag{7}$$

$$H_0 : p_b = p_c \tag{8}$$
$$H_1 : p_b \neq p_c \tag{9}$$

The McNemar's test is defined as:

$$X^2 = \frac{(b-c)^2}{b+c} \tag{10}$$

Given that b and c are sufficiently large $(b + c > 25)$, X^2 has a chi-squared distribution with 1 degree of freedom. If X^2 is sufficiently large, the null hypothesis can be rejected with confidence level a, and the two models are statistically different with confidence level a. In this research a is taken to be 95%. To reject the null hypothesis with 95% confidence a X^2 score of 3.841 or higher is needed.

4 SENTIMENT ANALYSIS

In this section sentiment analysis, the current STOA sentiment analysis methods, and ensemble learning will be explained

4.1 *Background*

Sentiment analysis, or opinion mining, is the field of finding the opinion of an author about specific entities, through self-automated tools (Chaudhuri, 2019; Feldman, 2013). An entity can be any noun, like persons, events, and topics. The objective of sentiment analysis is to classify whether an opinion is positive, negative, or neutral (Chandrakala & Sindhu, 2012; Pang & Lee, 2008).

An opinion can be mined on document-level, sentence-level or entity-level (Dey, Sarddar, Sarkar, Bose, & Roy, n.d.; B. Liu, 2012; Pozzi, Fersini, Messina, & Liu, 2016). In document-level analysis an entire document is analysed, and a single opinion is extracted on a single entity. The entire document is either positive, negative, or neutral regarding this entity. Sentence-level analysis aims to identify the sentiment of every sentence (Medhat, Hassan, & Korashy, 2014). Sentence-level analysis assumes that every sentence, that does contain a sentiment, is targeted at a single entity. The result is a number of sentiments, equal to or smaller than the number of sentences, on a variety of entities. In entity-level analysis, every opinion regarding every entity in a document is identified. The idea is that every opinion is paired with an entity. Within a single sentence, there can be multiple opinions and multiple entities. For example, the sentence "That book on opinion mining was great, but the binding was poor." contains an opinion regarding the content and the binding. The content quality is positive, but the binding quality is poor. Entity-level analysis identifies all of the opinion and entity combinations, whereas sentence and document-level analysis aggregate the opinions and assigns it to a single entity. This thesis deals with document-level sentiment analysis; determining the overall sentiment of a tweet regarding a CEO.

The most popular methods for sentiment analysis are ML methods. ML methods often use supervised classification, meaning that a classifier is trained on labelled data. Sentiment can be expressed as positive, negative, or neutral, resulting in a three-class classification problem. After training the classifier, it can be used on new, unseen data in the hopes that it generalizes well. The most powerful ML methods are deep learning (DL) methods.

DL, a division of ML, is based on artificial neural networks. DL methods can process raw data and automatically discover the representations needed

for classification (LeCun, Bengio, & Hinton, 2015). Raw data is fed into an input layer, which then uses a non-linear transformation to transform the data into a more abstract level. The next layer uses this non-linear transformation as its input layer and again transforms this. Given enough layers, very complex functions can be learned.

The sentiment of a word largely depends on the domain and context in which it exists (A. Das & Gambäck, 2012; Thelwall & Buckley, 2013). By not taking the context and domain into account words that on their own do not bear sentiment, but in the larger domain do, can be overlooked. A word can also have different meanings depending on the context. DL methods are the only methods that can distinguish different meanings for the same word. It accomplishes this using word embeddings.

Word embeddings were first introduced by Collobert and Weston (2008). It was not until 2013, however, that word embeddings became popular when Mikolov, Chen, Corrado, and Dean (2013) created the word2vec model. Word embeddings create vector representations of words, where similar words are closely located in a pre-defined feature space. The idea is that similar words often occur in a similar context. The most popular methods for word embeddings are word2vec and GloVe (Pennington, Socher, & Manning, 2014).

Glove and word2vec are first-generation pre-trained models (PTMs). A downside of these first-generation PTMs is that they operate on word-level. Second-generation PTMs, like BERT, XLNet, and T5 can analyse entire sentences, making them handle context better. These models use the Transformer architecture (Qiu et al., 2020; Vaswani et al., 2017).

4.2 *Transformer*

Transformer uses an encoder-decoder structure (Bahdanau, Cho, & Bengio, 2014; Cho et al., 2014) with attention mechanism. The encoder transforms a variable input sequence into a fixed-length sequence z of vector representations. This vector representation contains contextual meaning and positional information. The decoder takes z as its input and generates an output. Each step is auto-regressive, meaning that the previously generated output is added to the current input (Graves, 2013). The main component of Transformer is the Multi-Headed Attention (MHA) mechanism.

The MHA is composed of N attention mechanisms. The attention mechanism was first proposed by Bahdanau et al. (2014). It learns how each word is related to all other words in the input, whilst maintaining context. The idea is that not all words in a sentence are as informative. Some words hold more information than others. The attention mechanism tries to capture this property that some words hold more information.

It assigns weights to the input words, where the most important words receive a higher weight.

The current STOA sentiment analysis methods are all based on the encoder part of Transformer and discussed in the next section. All these models have been trained on an extremely large corpus to gain a general understanding of language. They differ in how they obtain this general understanding. The models are compared based on their accuracies on the SST-2 dataset, which is part of the GLUE benchmark.

4.3 *Sentiment Analysis Transformer Models*

Devlin, Chang, Lee, and Toutanova (2018) introduced BERT (Bidirectional Encoder Representations from Transformers). BERT uses the encoder part of Transformer to take in entire sentences. BERT was trained using Masked Language Modeling (MLM) to learn dependencies among words, as well as next sentence prediction (NSP) to learn relationships between sentences. MLM randomly masks words in a sentence. The idea is that BERT predicts these masked words by looking at all other words in the sentence. Masked words are replaced by a [MASK] token. This token creates a mismatch between training and implementation since real-world examples do not carry such tokens. To overcome this, 80% of the masked words is replaced with [MASK] token, 10% with a random token and 10% is left unchanged. NSP comes down to predicting whether sentence B follows sentence A. Two random sentences are chosen from a corpus and BERT decides whether B follows A, which is the case 50% of the time. Using MLM and NSP, BERT was able to get a better understanding of language than any other model at the time. BERT attained an accuracy of 94.9% on SST-2, being the top score when introduced.

Lan et al. (2019) introduced ALBERT (A Lite BERT). A downside of BERT is its size and thus computational cost. Lan et al. tried to find a solution to decrease the computational cost, whilst maintaining performance. To accomplish this ALBERT incorporates two techniques to reduce complexity: factorized embedding parameterization and cross-layer parameter sharing. ALBERT factorizes the parameters of the word embeddings. Factorization splits the original embedding matrix into input-level embeddings (size E) and hidden-layer embeddings (size H). E captures context-independent information and H captures context-dependent information. In BERT $E = H$. Since the goal is to learn context-dependent representations, Lan et al. argue that H is more important and should be larger than E. Taking a vocabulary size V, the total number of parameters of BERT is $O(V \cdot H = E)$. For ALBERT this is $O(V \cdot E + V \cdot H)$. Having $H >> E$ reduces the total number of parameters significantly. Cross-layer

parameter sharing enables sharing parameters from previous layers in layers blocks. ALBERT shares all parameters across all layers, resulting in a massive decrease of parameters. Besides the complexity reduction techniques, ALBERT also uses sentence-order prediction (SOP). BERT's NSP proved to be ineffective since it ends up focusing more on topic prediction rather than coherence. With SOP two consecutive sentences are taken from a corpus and segments from these sentences are swapped. ALBERT needs to determine whether parts of a sentence are swapped or not. Ultimately ALBERT attained an accuracy of 93.4%, being slightly lower than BERT, but with eighteen times fewer parameters.

XLNet (Yang et al., 2019) uses permutation language modelling (PLM) to overcome some of the shortcomings of BERT. BERT corrupts its input using [MASK] tokens, which creates discrepancy in real-world applications. Further, it neglects dependencies between masked positions. BERT acknowledged the downside of masking but did not find a real solution. PLM predicts the probability of a word using a sample of the permutations of all other words in a sentence. Permutations are all the possible orders of words. Given a sentence of length T, there are $T!$ permutations. For a sentence x, permutation z is being sampled one at a time to compute the likelihood $p_\theta(x_t)$. The parameter θ is shared across all permutations and thus, in expectation, x_t should be conditioned on every possible element in x. Using PLM XLNet overcomes the shortcomings of masking. XLNet is also able to overcome BERT's shortcoming of neglecting dependencies between masked positions since it is conditioned on all positions. Ultimately XLNet attained an accuracy of 97.1% on SST-2.

Li and Fourches (2020) created RoBERTa (Robustly optimized BERT pretraining approach), an improved version of BERT. Li and Fourches concluded that BERT was significantly undertrained. RoBERTa is based on the same architecture as BERT, but uses more training data and better parameter settings. Li and Fourches, just as Lan et al. (2019) note that NSP does not perform as expected. They decided to remove the NSP objective altogether. Eventually optimizing the parameters over more training data resulted in an accuracy of 96.7% on SST-2.

DeBERTa (He, Liu, Gao, & Chen, 2020) differs from BERT in two novel ways. First DeBERTa uses disentangled attention. Each word in DeBERTa is represented by two vectors, one for its content and one for the position. This is unlike BERT which uses one vector, composed of the sum of content and position. By using two vectors, the attention weights among words can be computed with better accuracy, since the attention weight of a word pair depends not only on their contents but also on their position. DeBERTa better captures this positional information, resulting in a better attention mechanism. Second, DeBERTa uses enhanced mask decoder. Like BERT,

DeBERTa uses MLM. He et al. (2020) argue that a word largely depends on its absolute position in a sentence. Two similar words may belong at different parts of a sentence. To account for this absolute position requirement, DeBERTa incorporates absolute word position embeddings right before decoding the masked words. Using these changes, DeBERTa managed to obtain an accuracy of 97.5% on SST-2.

T5 (Text-to-Text Transfer Transformer), developed by Raffel et al. (2019) takes text as an input and produces text as an output, regardless of the task. By viewing every task as a text-to-text framework, the same model, hyperparameters, and loss function can be used for all tasks. A specific task prefix is added to the input so that the model knows what to do. For multilabel classification, for example, the prefix "Multilabel-classification" is added. T5 uses MLM to learn dependencies. To speed up training it replaces multiple words with a single masking token. The objective of T5 is to predict this short sequences of masked words. Ultimately, T5 attained an accuracy of 97.5% on SST-2.

ELECTRA (Clark, Luong, Le, & Manning, 2020) differs from BERT only in how it corrupts its input to learn word dependencies. Instead of MLM, inputs are corrupted by replacing words with plausible alternatives. ELECTRA is trained to determine whether each word in a sentence is corrupted or not. This method is more efficient than MLM, since it incorporates all inputs rather than only the masked tokens. Using this method ELECTRA managed to obtain an accuracy of 97.1% on SST-2.

4.4 *Ensemble Learning*

Ensemble learning combines multiple models into a single predictor. The idea behind ensemble is that many know more than one. Simple techniques like majority vote and weighted average exist, but also more advanced like stacking, boosting, and bagging (Sewell, 2008). Stacking means using the predictions of multiple classifiers as inputs for a new meta-classifier. The relationships between inputs and outputs in stacking are often not as complex, since the input already contains processed information in the form of probabilities. Therefore, simple models like logistic regression, decision-tree, or SVM perform well (Pavlyshenko, 2018).

Boosting (Freund & Schapire, 1997) involves converting weak models into stronger ones to remove bias. Homogenous models are created sequentially and try to overcome the weaknesses of their predecessors (An & Kim, 2010). The most popular boosting algorithms are AdaBoost, CatBoost, lightGBM, XGBoost and gradient boosting (Rahman et al., 2020). The basic idea is the same for all models; create a homogeneous classifier, build on the errors of its predecessors, and combine the results to minimize loss.

Bagging (Breiman, 1996) creates multiple independent homogeneous models, each on a different subset of the data. The most popular way of creating subsets is using bootstrap aggregation (Dietterich, 2000). Here m examples are randomly drawn from the training data with replacement. By using random samples multiple independent models are created which combined results in lower variance.

5 METHODOLOGY

This section discusses the proposed scheme of this research. Figure 5 gives an overview of all the steps taken in this research. First various sentiment classifiers were created and compared. Next the gathered tweets on the CEOs were analysed and finally this information was used to predict stock price.

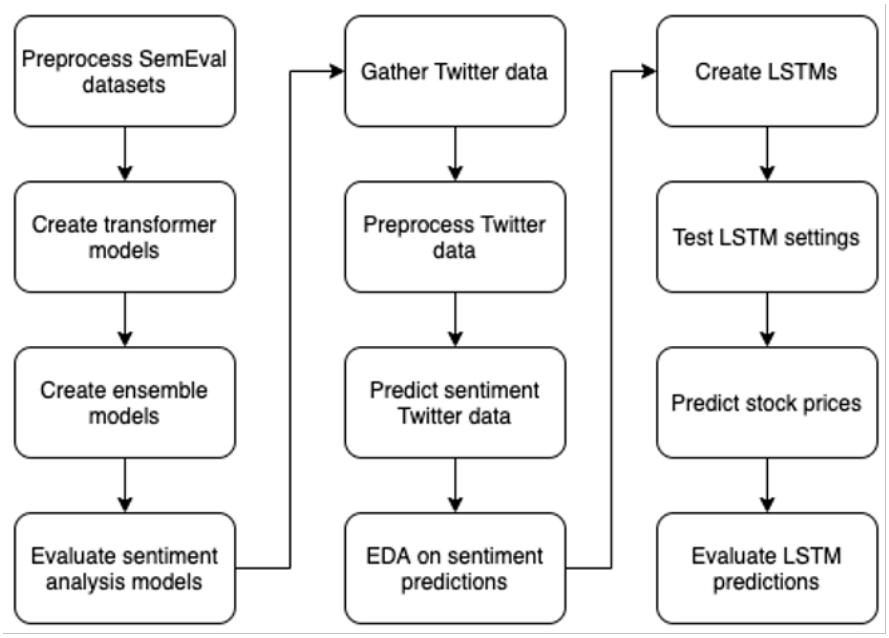

Figure 5: Research workflow

5.1 *Datasets*

The datasets used in the research are explained below.

1. Labelled training and test dataset to train the sentiment classifiers and evaluate their performance. The datasets from the SemEval-2017 challenge were used for both training and testing. The training set is an aggregation of earlier versions of the SemEval, consisting of 50,063 labelled instances, of which 45% is neutral, 40% positive and 15% negative. The test set contains 11,907 instances.

2. Twitter data on CEOs from 2017 to 2020. This data was gathered using Twint, a Python module that allows mining of Twitter data without limit. The CEOs Twitter username, as well as their full names, were used as keywords when mining tweets. CEOs from the top

eight S&P 500 companies were analysed. Figure 6 shows an overview of the CEOs and the number of acquired tweets. Elon Musk (Tesla), was excluded from this research due to the enormous amount of internet traffic he receives, which due to lack of computational power was impossible to analyse.

3. Daily financial stock data on the organizations of the relative CEO's, provided by Yahoo Finance from 2017 to 2020. The data consists of the stocks daily closing price. 2017 till 2019 was used as training data, and 2020 as testing data.

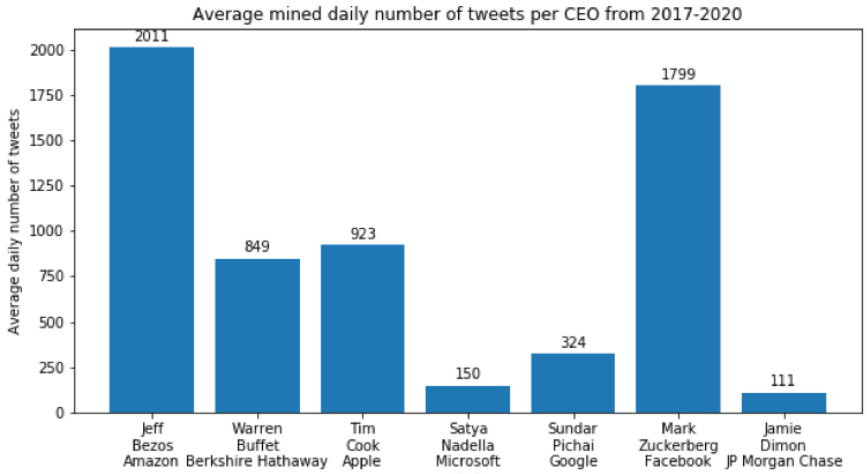

Figure 6: Average acquired daily number of tweets per CEO

5.2 *Sentiment Analysis*

5.2.1 *Preprocessing*

Data preprocessing is a necessary step for any NLP task and is done to reduce noise, remove uninformative information and improve classification accuracy (Asghar, Khan, Ahmad, & Kundi, 2014; Haddi, Liu, & Shi, 2013). Since language on Twitter is very informal and often contains abbreviators, misspellings, slang, links, and special characters, Twitter-specific techniques, as well as regular NLP techniques, are needed to clean the data (Smailović, 2015). Singh and Kumari (2016) investigated which preprocessing steps work best for sentiment analysis and ultimately created a pipeline of steps. Baziotis et al. (2017) created a preprocessing tool for their submission at SemEval-2017. They used many of the same steps as Singh and Kumari, but instead of removing special characters, they replaced them

with special tokens like <hashtag>. To determine the best preprocessing method, both methods were compared using XLNet. Average recall using no preprocessing was used as a baseline. The results are listed in Table 2. Using the preprocessing steps by Baziotis et al. increased average recall by 2.1%, whereas the method by Singh and Kumari decreased average recall by 10.2% over the baseline.

Method	Unprocessed	Singh and Kumari (2016)	Baziotis et al. (2017)
Average recall	0.700	0.628	0.715

Table 2: Average recall of different preprocessing techniques

5.2.2 *Fine-tuning Transformer models*

All the models mentioned in Section 4.3 are not trained for specific tasks, but rather to have a general understanding of language. To maximize their performance, these models need to be finetuned (Qiu et al., 2020). There are different approaches for finetuning: (i) retrain or update the entire model (all parameters), (ii) freeze part of the model and only update another part and (iii) freeze the entire model. In all cases finetuning is done by adding additional NN layer(s) on top of the model, which learn how to map inputs to outputs. A labelled dataset is required to update the NN parameters via an optimizer which minimizes the loss function.

Parameters of the NN are assigned randomly. Pre-training is done to better initialize the NN parameters (Alt, Hübner, & Hennig, 2019). When finetuning, a very small learning rate should be used to avoid overfitting (Mosbach, Andriushchenko, & Klakow, 2020). The most used method is to freeze all layers for pretraining and unfreeze them for finetuning. This method is also used in this research. The pre-training parameters can be found in Table 3. These parameters are based on the original papers. For finetuning a slightly lower learning rate provided better results. All other parameters were kept the same.

Parameters	**BERT**	**ALBERT**	**RoBERTa**	**DeBERTa**	**ELECTRA**	**XLNET**	**T5**
Dropout rate	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Learning rates	$1e^{-5}$	$1e^{-5}$	$2e^{-4}$	$2e^{-4}$	$1e^{-5}$	$2e^{-5}$	$1e^{-3}$
Weight decay	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Learning rate decay	linear	linear	linear	linear	linear	linear	linear
Adam e	$1e^{-6}$	$1e^{-6}$	$1e^{-6}$	$1e^{-6}$	$1e^{-6}$	$1e^{-6}$	Adafactor
*Adam b*1	0.9	0.9	0.9	0.9	0.9		
*Adam b*2	0.999	0.999	0.999	0.999	0.999		

Table 3: Pretraining parameters for Transformer models

5.2.3 *Sentiment analysis results*

All models mentioned in Section 4.3 were tested, as well as Cardiff, which is a RoBERTa model trained on Twitter data. The models were compared on their average recall, as determined by the organizers of SemEval-2017 (Rosenthal, Farra, & Nakov, 2017). The highest score of the SemEval-2017 was used as a baseline. This score is 0.681 by Baziotis et al. (2017). The models are also compared to the model by Azzouza, Akli-Astouati, and Ibrahim (2019), who obtained an F1-score of 0.718.

In line with Dodge et al. (2020), various seeds were tested since the initialization of the weights is of great influence on the final performance. This thesis followed their proposed method of trying many models, early stop the majority and continue with the most promising ones. Ten random seeds were selected and trained for one epoch. The top three best performing models were then fully finetuned. The difference between seeds was as large as 9%. Due to computational limitations, only ten seeds were tried. The highest score per model is listed in Table 4.

Metric	**BERT**	**ALBERT**	**RoBERTa**	**DeBERTa**	**ELECTRA**	**XLNET**	**T5**	**Cardiff**
Average recall	0.683	0.695	0.712	0.699	0.715	0.720	0.702	0.719
F1-score	0.678	0.692	0.699	0.689	0.686	0.708	0.689	0.702

Table 4: Accuracies on SemEval-2017

The outputs of the individual models were used in the various ensemble techniques explained in Section 4.4. The outputs are probability scores of whether a tweet is positive, negative, or neutral. For stacking, boosting, and bagging decision tree, logistic regression and SVM were tested as weak learners with varying hyperparameters. The highest obtained scores are listed in Table 5.

Metric	**Stacking**	**Bagging**	**Boosting**	**Majority vote**	**Weighted vote**
Average recall	0.713	0.711	0.708	0.727	0.731
F1-score	0.710	0.708	0.709	0.719	0.723

Table 5: Ensemble accuracies on SemEval-2017

5.3 *Stock price prediction*

The gathered tweets on the CEOs were preprocessed using the method by Baziotis et al. (2017). ELECTRA was used as the model to generate sentiment predictions, as it obtained the highest F1-score of any standalone

Name	Meaning
MA10	10-day simple moving average
MA20	20-day simple moving average
MA50	50-day moving average
Diff	Difference EMA12, EMA26
DEA	9-day exponential moving average of DIFF
RSI6	6-day relative strength index
RSI12	12-day relative strength index
RSI24	24-day relative strength index

Table 6: Technical indicators

model. The additional performance of weighted vote ensemble does not outweigh its computational cost and was thus not used.

Stocks are only traded on weekdays, but any new information from the weekend might still be of influence on Monday's price. Monday's polarity score is therefore the weighted average of Saturday, Sunday, and Monday. In line with Ko and Chang (2021) the percentage of positive and negative tweets are used as sentiment scores. The total daily number of tweets, referred to as volume, was also used as an input to determine whether this is an indicator of stock price trend.

$$\text{Positive score} = \frac{\text{Number of positive tweets}}{\text{Total number of tweets}} \tag{11}$$

$$\text{Negative score} = \frac{\text{Number of negative tweets}}{\text{Total number of tweets}} \tag{12}$$

The goal of this thesis was to predict stock price trend. To do so, prior day's prices, sentiment scores, Twitter volume, and technical indicators were used as inputs. Bollen et al. (2011) found a delay between Twitter sentiment and its reflection in stock price. Therefore, various delays, ranging from 0 to 5 days, for sentiment scores and volume were tested. More formally, sentiment on day t_{-x} was used as a predictor at day t to predict stock price for day $t+1$.

Li et al. (2020) included technical indicators in their LSTM-model, boosting model performance. Technical indicators reflect trends or fluctuations of the market. The names and meanings of the technical indicators are shown in Table 6.

Simple moving average (SMA) is defined as:

$$\text{SMA}_N = \frac{1}{N} \sum_{t-n:t}^{k} P_t \tag{13}$$

where N is the number of periods considered, t is the current date and P_t is the price at date t. In SMA all observations are weighted equally, meaning that it assumes all observations are equally representative of the future (Ellis & Parbery, 2005).

Exponential moving average (EMA) is defined as:

$$\text{EMA} = P_t \cdot k + \text{EMA}_{t-1} \cdot (1 - k) \tag{14}$$

where $k = \frac{2}{N+1}$. EMA, unlike SMA, emphasizes recent observations more. It assumes that more recent observations are more representative of the future. The more recent an observation, the higher the weight it receives.

Relative strength index (RSI) is defined as:

$$\text{RSI}_N = 100 - \left[\frac{100}{1 + \frac{\text{Average loss}}{\text{Average gain}}} \right] \tag{15}$$

RSI measures how fast and how much prices change. Higher levels of RSI indicate that a stock is overbought and lower levels that it is oversold.

Since each feature value's range is diverse, the descending path of the optimizer is more tortuous and convergence of the model is harder (Wu & Zhao, 2020). To facilitate easier convergence, all features are normalized using z-score, defined as:

$$z = \frac{x - \mu}{\sigma} \tag{16}$$

where x is the observation, μ the mean and σ the standard deviation.

5.3.1 *LSTM model*

The LSTM architecture of this thesis is based on that of Li et al. (2020). The architecture consists of an input layer, two LSTM layers, two dropout layers, and a dense prediction layer (Figure 7). The input layer receives the raw data and feeds it into the LSTM layer. The LSTM layer learns how to map the inputs to outputs. The dropout layer is added to avoid overfitting by preventing co-adaptation between units (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). The Dense layer returns the predicted stock price based on the input variables, the computed weights, and bias term.

Mean squared error (MSE) was used as the loss function to calculate prediction errors and optimize hyperparameter settings. Even though the

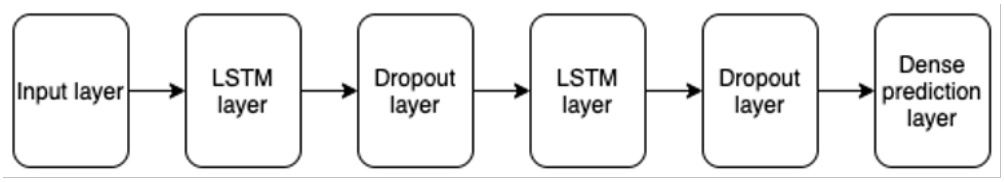

Figure 7: LSTM architecture

goal was stock price trend prediction, which is a classification problem, MSE, which is used for regression optimization, was used. When training the model in a classification setting, the model ended up not learning anything most of the time and simply predicted the majority class. The model was often unable to map inputs to outputs in a way that improved accuracy over predicting the majority class. This is likely due to the random nature of stock prices. Training the model in a regression setting forced the model to predict a value and thus learn how to map inputs to outputs, even when this led to worse performance. This allowed comparison across all created models on their performance. MSE is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{17}$$

where n is the number of datapoints, Y_i the actual values and $\hat{Y}_i$ the predicted values. To use the outputs for trend prediction, any predicted values that are higher than their actual value are classified as "up" and any predicted values that are lower than their actual value are classified as "down":

$$\text{Classification} = \begin{cases} \text{Up} & \text{if } Y_i < \hat{Y}_i \\ \text{Down} & \text{if } Y_i > \hat{Y}_i \end{cases} \tag{18}$$

Adam (Kingma & Ba, 2014) was used as the optimizer for the LSTM. Adam was chosen because it is computationally efficient, requires little tuning and works well for DL problems (Jais, Ismail, & Nisa, 2019; Kingma & Ba, 2014). An important parameter is the learning rate. Larger learning rates will make the loss function converge faster towards a local minimum, but have a harder time achieving optimality due to the larger steps. Smaller learning rates will take longer to converge but may result in a more optimal set of weights. When the learning rate is too small, the model may get stuck in a local minimum and achieve suboptimal result.

An often-used approach is to lower the learning rate as training evolves. Adam finds optimal learning rates for each weight in the network, thereby not requiring the above (Kingma & Ba, 2014). The hyperparameter learning rate λ is the maximum value that the individual weights can receive. Every weight can be updated using a learning varying from 0 (no update) to λ

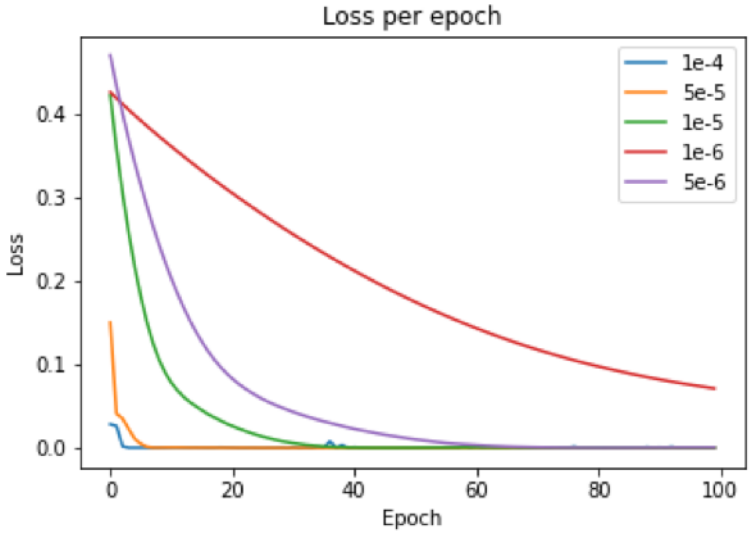

Figure 8: Effect learning rate on loss

Hyperparameter	Settings
Hidden layer size	20, 50, 100, 200
Dropout rate	0.1, 0.2, 0.3
Window size	5, 10, 30, 50
Learning rate	$1e^{-5}, 5e^{-6}$

Table 7: LSTM hyperparameter settings

(maximal update). To determine the best learning rate various learning rates were tested and their performances were compared. Figure 8 shows an overview of the validation loss over 100 epochs. Ultimately a learning rate of $1e^{-5}$ and $5e^{-6}$ were tested as they gave the best results and have a nice smooth curve, which is indicative of a good model.

Various hyperparameters of LSTM were tested, namely the number of hidden layers, dropout rate, random seed, and window size, all other parameters were set to default. Window size refers to the number of past observations that are used to predict the future. Using a window of size w, prediction at time t is based on inputs from t_{-w} till t_{-1}. Table 7 shows the tested hyperparameter settings. All combinations of these hyperparameters were tested.

In line with Dodge et al. (2020) were all hyperparameter settings were trained for 25 epochs. The top three performing models were trained again, but for 50 epochs. Figure 8 shows that after 50 epochs the model has already converged close to its optimum. Ultimately, the best performing model was trained for 1000 epochs with early stopping and checkpoints.

For the final run, the training and validation set were concatenated to have more training data. Early stopping was used to avoid overfitting, potentially improving performance, and speed up training (Prechelt, 1998). Whenever the loss function did not decrease for 50 consecutive epochs, the training was terminated. Checkpoints were used to keep track of the best performing epoch. Whenever the current validation loss was lower than the previous best performing epoch, the checkpoint was updated. This makes sure that the best performing settings are used.

All combinations of possible inputs were tested to determine which worked best for which stock. All models were compared on their accuracy in predicting stock price trend.

Accuracy in this case means the number of times the predicted trend matched the actual trend over the number of trading days.

$$\text{Accuracy} = \frac{\text{Number of times predicted trend = actual trend}}{\text{Number of trading days}} \tag{19}$$

Besides the accuracy, return on investment was used as an evaluation metric, just like Picasso et al. (2019) did. Starting with a hypothetical portfolio of $1000, daily trend predictions were generated using the top performing LSTM model for each stock. When the model predicted up, shares were bought/kept. When the model predicted down, shares were sold/cash is kept. Return is defined as:

$$\text{Return}_t = \frac{\text{Value}_t - \text{Value}_{t=0}}{\text{Value}_{t=0}} \cdot 100\% \tag{20}$$

6 RESULTS

This section discusses the results from the LSTM predictions on 2020 stock data.

6.1 *Baseline*

The baseline was computed by testing all combinations of past prices and technical indicators as inputs in the LSTM. Table 8 shows the highest obtained accuracy per stock, as well as the majority vote accuracy. The majority vote represents the accuracy when always predicting that the price goes up.

When using majority vote, the average accuracy is 53.95%, and when including past prices and technical indicators as variables, the average accuracy is 59.29%. All stocks see an increase in accuracy when using any combination of past prices and technical indicators as inputs. The baseline for further comparison is the average accuracy of 59.29%.

Variables	**Amazon**	**Apple**	**Google**	**JP Morgan**	**Facebook**	**Microsoft**	**Berkshire Hathaway**
Majority vote accuracy	55.21	54.62	55.81	50.44	53.43	56.11	53.03
Highest accuracy without sentiment	**55.32**	**59.68**	**62.06**	**58.10**	**61.26**	**59.29**	**59.29**

Table 8: Baseline accuracy per stock

6.2 *LSTM Results*

Table 9 shows the highest accuracies per stock when including sentiment and/or volume as a variable. Again, all possible combinations of inputs were tested. Appendix A (page 44) shows the obtained accuracies for all combinations. The highest accuracy per stock using only sentiment as additional variable, only volume as additional variable, and using both sentiment and volume as additional variables are reported.

When including sentiment as a variable, the average accuracy is 59.74%. When including volume, the average accuracy is 59.31%. When including both sentiment and volume, the average accuracy is 59.67%. On average, including sentiment, volume, or both, boosts model performance and beats the baseline. Using the best performing settings for each stock, the average accuracy increases to 60.47%.

Variables	**Amazon**	**Apple**	**Google**	**JP Morgan**	**Facebook**	**Microsoft**	**Berkshire Hathaway**
Sentiment	57.31	60.87	**60.47**	56.52	**63.24**	**59.69**	60.08
Volume	57.70	60.24	59.68	**59.68**	61.66	58.50	57.71
Sentiment + volume	**57.71**	**60.87**	60.01	56.13	62.85	58.50	**61.66**

Table 9: Accuracy per stock including sentiment, volume or both

To determine whether there is a delay between Twitter information and its reflection in stock price, the top performing settings for each stock were used, with a delay interval from zero to five days for sentiment scores and volume. Table 10 gives an overview of the accuracies per stock using the various time intervals. When including a possible delay, the average accuracy increased to 61,86%.

Delay (days)	**Amazon**	**Apple**	**Google**	**JP Morgan**	**Facebook**	**Microsoft**	**Berkshire Hathaway**
0	57.71	60.87	60.47	**59.68**	**63.24**	59.69	61.66
1	54.54	60.87	59.29	58.89	60.08	58.50	**63.63**
2	55.73	**61.66**	59.29	58.98	59.68	55.73	58.50
3	**59.68**	**61.66**	60.47	58.89	58.50	57.70	62.85
4	54.54	60.47	**61.66**	56.92	60.08	**61.26**	56.12
5	55.34	**61.66**	61.26	58.50	59.29	59.29	54.54

Table 10: Accuracy per stock accounting for delay

On average, including daily sentiment scores and/or volume as variables, and accounting for a delay, the accuracy increased by 2.57% over the baseline accuracy.

The best performing settings for every model, including possible delay, were used to predict stock price trend predictions. These predictions were then used to trade using a hypothetical portfolio of 1000 dollars. When the model predicted up, shares were bought/kept. When the model predicted down, shares were sold/cash was kept. Table 11 shows how much a 1000 dollars of a certain stock on January 1st, 2020, was worth on December 31st, 2020, when not trading at all, when trading conform the baseline LSTM prediction, and when trading conform the best performing LSTM predictions.

The average return when holding a stock from January 1st, 2020, till December 31st, 2020, is 33.83%. When trading conform the baseline LSTM predictions the average return increased to 124.98%. When trading conform the best performing LSTM predictions, all stocks, but Google and Facebook,

Price	**Amazon**	**Apple**	**Google**	**JP Morgan**	**Facebook**	**Microsoft**	**Berkshire Hathaway**
No trading	1715.97	1767.14	1281.20	900.63	1302.12	1384.76	1016.23
Baseline LSTM	1870.86	3004.33	2241.66	1818.24	2866.15	2275.07	1672.47
Best LSTM	2265.97	3330.90	2134.65	1875.82	2823.71	2443.61	1914.89

Table 11: Portfolio value on December 31st after investing $1000 on January 1st 2020

saw an increase in return over the baseline, and the average return increased to 139.85%.

6.3 *McNemar's Test*

McNemar's test was used to test whether the predictions of an LSTM with sentiment, volume, or both, is significantly different from one without, and thus whether the LSTM with sentiment, volume, or both, learns how to map inputs to output differently, with confidence level a. Here a is taken to be 95%. The baseline model is an LSTM including past prices and technical indicators. When there was an observed delay this delay was also implemented when conducting the McNemar's test. The McNemar's test proved to be volatile, therefore the average score over 10 runs was computed.

The Chi-square scores, their standard deviation, and corresponding p-values for including sentiment are listed in Table 12, for including volume in Table 13, and for including both in Table 14. Only Microsoft experiences a statistically different model when including sentiment, volume, or both, but it has a high standard deviation.

Assuming that the McNemar's scores are normally distributed around the mean, a 95% confidence interval for the distribution of the mean can be computed. A 95% confidence lower bound and upper bound can be calculated by:

$$\text{Upper Bound} = \text{mean} + 1.96 \cdot \text{standard deviation} \tag{21}$$

$$\text{Lower Bound} = \text{mean} - 1.96 \cdot \text{standard deviation} \tag{22}$$

When doing so for Microsoft, a 95% confidence interval for McNemar's score including sentiment is [1.967 , 14.979], including volume is [4.099 , 28.489], and including both is [-5.469 , 17.925]. Only for volume is the lower bound high enough to accommodate a statistically significant model.

McNemar's sentiment	**Amazon**	**Apple**	**Google**	**JP Morgan**	**Facebook**	**Microsoft**	**Berkshire Hathaway**
Mean Chi-square	2.883	1.203	0.661	0.248	0.366	8.474	0.396
St. dev. Chi-square	2.526	1.849	0.645	0.372	0.286	3.319	0.275
P-value	0.089	0.273	0.416	0.619	0.545	0.004*	0.529

Table 12: McNemar's test influence sentiment

McNemar's volume	**Amazon**	**Apple**	**Google**	**JP Morgan**	**Facebook**	**Microsoft**	**Berkshire Hathaway**
Mean Chi-square	0.753	2.770	0.829	1.059	0.575	16.294	0.152
St. dev. Chi-square	0.705	2.145	0.604	1.389	0.672	6.222	0.138
P-value volume	0.386	0.096	0.363	0.303	0.448	0.000*	0.697

Table 13: McNemar's test influence volume

McNemar's sentiment *and* volume	**Amazon**	**Apple**	**Google**	**JP Morgan**	**Facebook**	**Microsoft**	**Berkshire Hathaway**
Mean Chi-square	2.471	2.447	0.654	0.294	0.331	6.228	0.158
St. dev. Chi-square	1.387	3.138	0.461	0.302	0.351	5.968	0.144
P-value	0.116	0.118	0.419	0.588	0.565	0.013*	0.691

Table 14: McNemar's test influence sentiment *and* volume

7 DISCUSSION

The goal of this research was to predict stock price trend using CEO reputation. To determine CEO reputation sentiment analysis on Twitter data was performed.

7.1 *Sentiment Analysis*

Sentiment analysis was performed using ELECTRA, which was the best performing model out of the ones compared in 4.3. ELECTRA beat the baseline score of Baziotis et al. (2017) by 5.73%. It did not, however, beat the f1-score by Azzouza et al. (2019) of 0.718. This is likely because Azzouza et al. trained BERT on Twitter data and used the large version of BERT. Cardiff shows that training on Twitter data increases model performance. All compared models were trained by the original authors on extremely large datasets (Wikipedia, BooksCorpus, C4), but not Twitter data. Further, all authors of the compared models found that the larger the model, the better it performs. Due to computational limitations, only the base versions of all models were used.

This research did not perform an extensive hyperparameter search due to computational limitations. The settings proposed by the original authors were used. These settings were not specifically designed for Twitter sentiment analysis. Trying out various settings for all hyperparameters may boost model performance, but it very computational expensive. Besides parameter tuning, trying more seeds may boost performance further. A variation in average recall score of as large as 9% was found between different seeds. Testing more seeds may further boost performance. This, again, is computationally expensive. Many of the best performing sentiment analysis models, as determined on the SST-2 dataset, were tested in this research. The best performing model, ERNIE 2.0, however, was not due to lack of extensive documentation. Testing more of the top performing models, including ERNIE 2.0, could result in a better model for Twitter sentiment analysis.

The outputs of the individual models from Section 4.3 were also used in the various ensemble techniques explained in Section 4.4. The outputs consisted of the probability scores of a tweet being positive, negative, or neutral. Bagging, boosting, and stacking did not lead to an increase in performance. These methods are designed to combine multiple weak models into a stronger model. In this research, the individual models are already very powerful, and bagging and boosting were not able to increase performance further. Bagging and boosting are normally trained on raw data, and not probability outputs. This may be a reason why bagging and

boosting did not lead to an increase in performance.

Unlike bagging and boosting, stacking is designed to use the outputs of various individual models as inputs to create a better model. In this research, stacking did not lead to an increase in performance. This means that the probability outputs of the individual models did not contain enough information to better predict sentiment when used in stacking.

Majority vote and weighted average over all individual models did increase performance. Ultimately weighted average performed best with an average recall of 0.731, beating the score of Baziotis et al. by 7.3%. Weighted average also obtained an f1-score of 0.723, beating the f1-score by Azzouza et al. by 0.70%.

7.2 *Stock Price Prediction*

ELECTRA was used to generate Twitter sentiment predictions on the gathered tweets. The predicted sentiment was then used in an LSTM model to predict stock price. Besides sentiment, the daily number of tweets was also included as a possible variable, as it may be related to the daily sentiment. The average baseline accuracy, which was computed using an LSTM without sentiment, or volume, was 59.29%.

When including sentiment as a predictor, average accuracy increased by 0.45% over the baseline. All stocks, but Google and JP Morgan saw an increase in accuracy when including sentiment as a variable. To determine whether this increase in accuracy was not due to chance, a McNemar's test was conducted. The average McNemar's score over ten runs was computed to offset some of the variance. When including sentiment only Microsoft ended up with a statistically significant different LSTM model. Due to the high standard deviation in McNemar's scores, it can, however,not be definitively concluded that sentiment leads to a statistically different model for Microsoft, since the 95% confidence lower bound does not achieve a statistically significant McNemar's score.

When including volume as a predictor, average accuracy increased by 0.02% over the baseline. Amazon, Apple, Facebook, and JP Morgan were the only stocks who did see an increase in accuracy when including volume. Again, a McNemar's test was conducted to determine whether volume led to a statistically different LSTM. Only Microsoft ended up with a statistically significant different LSTM model, and this time the 95% lower bound was still statistically significant. Microsoft, however, did not see an increase in performance, thus even though the model was statistically different, and thus learned how to map inputs to outputs differently when including volume, it was not better.

When including both sentiment and volume as variables, average accu-

racy increased by 0.38% over the baseline. This time Amazon, Facebook, Apple, and Berkshire Hathaway were the only models that did see an increase in accuracy. Again, Microsoft was the only model to end up with a statistically significant McNemar's score, but not after accounting for the standard deviation.

Using the best performing settings for each stock, including a possible delay between Twitter information and its reflection in stock price, all stocks, but Google, saw an increase in accuracy. The average accuracy increased to 61.86%, a 2.57% increase over the baseline. Using these best performing LSTM settings, the average return was 139.85%, which is significantly higher than the 80% return of Picasso et al. (2019). Direct comparison to the result of Picasso et al. is difficult due to the difference in time frames. Nontheless, including Twitter information does lead to a substantial increase in return.

7.3 *Limitations*

This thesis assumed that twitter CEO reputation could be measured using sentiment analysis on Twitter data. There is no proof that CEO reputation can be measured using sentiment analysis. Reputation is a very abstract. It describes the feelings of a large public regarding someone. It is only possible to approximate the feelings of this large public using automated means, there is no guarantee that the predictions of a model are correct.

Another limitation is the number of stocks. Analysing more companies and CEOs could give a better indication of the significance of CEO reputation on stock price. Only the largest companies were analysed. Companies who on its own receive a lot of internet traffic. It might be possible that analysing a smaller company, with a less popular CEO, results in a different outcome.

Another limitation is the time frame. As stated above, 2020 was a volatile year for the stock market, where some stocks experienced record positive returns and others large negative returns. The average return of Amazon, for instance, from 2017 till 2019 was 35.70%. In 2020 this was 71.16%. The higher average actual returns of 2020 naturally led to a higher return when trading conform thee LSTM predictions. Using the LSTM architecture for any other year will most likely result in lower returns. For a better understanding of model performance, a larger time frame may be needed.

8 CONCLUSION

This section answers the main research question and the sub-questions as stated in the introduction.

8.1 *To what extend is CEO reputation, measured using sentiment analysis, predictive of stock price trend?*

Using CEO reputation, measured using sentiment analysis, as a variable increased model performance, on average, from 59.29% to 59.74%, a 0.45% increase. The McNemar's test was used to determine whether this increase in performance was significant. None of the analysed stocks experience a statistically different model when including sentiment. CEO reputation alone is thus not a statistically significant predictor of stock price. Nonetheless the average accuracy increased slightly, indicating that including sentiment in stock price predictions may sometimes be beneficial.

When including volume and sentiment as variables, model performance increased to 60.47%, a 1.18% increase over the baseline. Again, none of the stocks experienced a statistically different model. Even though including volume and sentiment does not lead to a statistically significant model, performance was boosted indicating that including both might be beneficial when predicting stock price. Even an increase of 1.18% in model performance is an accomplishment when predicting something as volatile as stock price.

8.2 *To what extent is the amount of internet traffic related to stock price movements?*

Using the daily number of tweets as a variable, model performance, on average, increased from 59.29% to 59.34%, a 0.05% increase. Only four out of the seven stock saw an increase in performance when including volume as a predictor. Only for Microsoft did volume result in a statistically different model. Microsoft, however, did not see an increase in performance. The daily number of tweets alone is not a statistically significant predictor of stock price and did not boost model performance by much, indicating that including volume alone for stock price prediction does not seem to be worth it.

8.3 *To what extend is there a delay between Twitter comments and its reflection in stock price?*

The average accuracy without including a delay, using the best performing settings per stock was 60.47%. When introducing a possible delay, the average accuracy increased to 61.86%, a 1.39% increase. Five out of the seven stocks saw an increase in model performance when including a delay, indicating that any information on Twitter is often not directly represented in stock price. When predicting stock price using Twitter sentiment, a possible delay should be included.

8.4 *To what extent do Transformer models improve sentiment analysis classification?*

The baseline score on the SemEval-2017 dataset was an average recall of 0.681 by Baziotis et al. (2017), who used an LSTM for sentiment prediction. The average recall score obtained by the Transformer models was 0.706, beating the baseline score by 3.62%. The top performing model, ELECTRA, obtained an average recall of 0.720, beating the baseline by 5.73%.

8.5 *To what extent does ensemble of Transformer models outperform individual classifiers?*

Weighted average was the best performing ensemble method and resulted in an average recall of 0.731. Weighted average increased model performance by 3.60% when compared to the average score of all individual models, and it beat ELECTRA, the top performing model stand-alone model, by 1.53%.

REFERENCES

Alt, C., Hübner, M., & Hennig, L. (2019). Fine-tuning pre-trained transformer language models to distantly supervised relation extraction. *arXiv preprint arXiv:1906.08646*.

An, T.-K., & Kim, M.-H. (2010). A new diverse adaboost classifier. In *2010 international conference on artificial intelligence and computational intelligence* (Vol. 1, pp. 359–363).

Asghar, M. Z., Khan, A., Ahmad, S., & Kundi, F. M. (2014). A review of feature extraction in sentiment analysis. *Journal of Basic and Applied Scientific Research*, *4*(3), 181–186.

Azzouza, N., Akli-Astouati, K., & Ibrahim, R. (2019). Twitterbert: Framework for twitter sentiment analysis based on pre-trained language model representations. In *International conference of reliable information and communication technology* (pp. 428–437).

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Batra, R., & Daudpota, S. M. (2018). Integrating stocktwits with sentiment analysis for better prediction of stock price movement. In *2018 international conference on computing, mathematics and engineering technologies (icomet)* (pp. 1–5).

Baziotis, C., Pelekis, N., & Doulkeridis, C. (2017). Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th international workshop on semantic evaluation (semeval-2017)* (pp. 747–754).

Black, E. L., Carnes, T. A., & Richardson, V. J. (2000). The market valuation of corporate reputation. *Corporate Reputation Review*, *3*(1), 31–42.

Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of computational science*, *2*(1), 1–8.

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123–140.

Cakra, Y. E., & Trisedya, B. D. (2015). Stock price prediction using linear regression based on sentiment analysis. In *2015 international conference on advanced computer science and information systems (icacsis)* (pp. 147–154).

Chandrakala, S., & Sindhu, C. (2012). Opinion mining and sentiment classification a survey. *ICTACT journal on soft computing*, *3*(1), 420–425.

Chaudhuri, A. (2019). *Visual and text sentiment analysis through hierarchical deep learning networks*. Springer.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv*

preprint arXiv:1406.1078.

Clark, K., Luong, M.-T., Le, Q. V., & Manning, C. D. (2020). Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Cliche, M. (2017). Bb_twtr at semeval-2017 task 4: Twitter sentiment analysis with cnns and lstms. *arXiv preprint arXiv:1704.06125*.

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on machine learning* (pp. 160–167).

Das, A., & Gambäck, B. (2012). Sentimantics: conceptual spaces for lexical sentiment polarity representation with contextuality. In *Proceedings of the 3rd workshop in computational approaches to subjectivity and sentiment analysis* (pp. 38–46).

Das, S., Behera, R. K., Rath, S. K., et al. (2018). Real-time sentiment analysis of twitter streaming data for stock prediction. *Procedia computer science*, *132*, 956–964.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dey, R. K., Sarddar, D., Sarkar, I., Bose, R., & Roy, S. (n.d.). A literature survey on sentiment analysis techniques involving social media and online platforms. *International Journal Of Scientific & Technology Research*, *1*(1).

Dickinson, B., Hu, W., et al. (2015). Sentiment analysis of investor opinions on twitter. *Social Networking*, *4*(03), 62.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems* (pp. 1–15).

Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., & Smith, N. (2020). Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Ellis, C. A., & Parbery, S. A. (2005). Is smarter better? a comparison of adaptive, and simple moving average trading strategies. *Research in International Business and Finance*, *19*(3), 399–411.

Fama, E. F. (1965). The behavior of stock-market prices. *The journal of Business*, *38*(1), 34–105.

Feldman, R. (2013). Techniques and applications for sentiment analysis. *Communications of the ACM*, *56*(4), 82–89.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, *55*(1), 119–139.

Gaines-Ross, L. (2000). Ceo reputation: A key factor in shareholder value.

Corporate Reputation Review, *3*(4), 366–370.

Gallagher, L. A., & Taylor, M. P. (2002). Permanent and temporary components of stock prices: Evidence from assessing macroeconomic shocks. *Southern Economic Journal*, 345–362.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Guo, X., & Li, J. (2019). A novel twitter sentiment analysis model with baseline correlation for financial market prediction with improved efficiency. In *2019 sixth international conference on social networks analysis, management and security (snams)* (pp. 472–477).

Haddi, E., Liu, X., & Shi, Y. (2013). The role of text pre-processing in sentiment analysis. *Procedia Computer Science*, *17*, 26–32.

He, P., Liu, X., Gao, J., & Chen, W. (2020). Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Ismail, H., Harous, S., & Belkhouche, B. (2016). A comparative analysis of machine learning classifiers for twitter sentiment analysis. *Res. Comput. Sci.*, *110*, 71–83.

Jais, I. K. M., Ismail, A. R., & Nisa, S. Q. (2019). Adam optimization algorithm for wide and deep neural network. *Knowl. Eng. Data Sci*, *2*(1), 41–46.

Jin, Z., Yang, Y., & Liu, Y. (2019). Stock closing price prediction based on sentiment analysis and lstm. *Neural Computing and Applications*, 1–17.

Kalyani, J., Bharathi, P., Jyothi, P., et al. (2016). Stock trend prediction using news sentiment analysis. *arXiv preprint arXiv:1607.01958*.

Kavussanos, M. G., & Dockery, E. (2001). A multivariate test for stock market efficiency: the case of ase. *Applied Financial Economics*, *11*(5), 573–579.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ko, C.-R., & Chang, H.-T. (2021). Lstm-based sentiment analysis for stock price forecast. *PeerJ Computer Science*, *7*, e408.

Kordonis, J., Symeonidis, S., & Arampatzis, A. (2016). Stock price forecasting via sentiment analysis on twitter. In *Proceedings of the 20th pan-hellenic conference on informatics* (pp. 1–6).

Kossovsky, N., Greenberg, M. D., & Brandegee, R. C. (2012). *Reputation, stock price, and you: Why the market rewards some companies and punishes others*. Springer.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

Li, X., & Fourches, D. (2020). Inductive transfer learning for molecular activity prediction: Next-gen qsar models with molpmofit. *Journal of Cheminformatics*, *12*, 1–15.

Li, X., Wu, P., & Wang, W. (2020). Incorporating stock prices and news sentiments for stock market prediction: A case of hong kong. *Information Processing & Management*, *57*(5), 102212.

Li, X., Xie, H., Chen, L., Wang, J., & Deng, X. (2014). News impact on stock price return via sentiment analysis. *Knowledge-Based Systems*, *69*, 14–23.

Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, *5*(1), 1–167.

Liu, Y., Qin, Z., Li, P., & Wan, T. (2017). Stock volatility prediction using recurrent neural networks with sentiment analysis. In *International conference on industrial, engineering and other applications of applied intelligent systems* (pp. 192–201).

Lu, Y. (2010). A revised version of mcnemar's test for paired binary data. *Communications in Statistics—Theory and Methods*, *39*(19), 3525–3539.

Malkiel, B. G., & Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work*. *The Journal of Finance*, *25*(2), 383-417. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1970.tb00518.x doi: https://doi.org/10.1111/j.1540-6261.1970.tb00518.x

McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, *12*(2), 153–157.

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, *5*(4), 1093–1113.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mittal, A., & Goel, A. (2012). Stock prediction using twitter sentiment analysis. *Standford University, CS229 (2011 http://cs229. stanford. edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis. pdf)*, *15*.

Mosbach, M., Andriushchenko, M., & Klakow, D. (2020). On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. *arXiv preprint arXiv:2006.04884*.

Nelson, D. M., Pereira, A. C., & de Oliveira, R. A. (2017). Stock market's price movement prediction with lstm neural networks. In *2017 international joint conference on neural networks (ijcnn)* (pp. 1419–1426).

Nguyen, T. H., & Shirai, K. (2015). Topic modeling based sentiment analysis on social media for stock market prediction. In *Proceedings of the 53rd annual meeting of the association for computational linguistics*

and the 7th international joint conference on natural language processing (volume 1: Long papers) (pp. 1354–1364).

Oliveira, A. L., & Meira, S. R. (2006). Detecting novelties in time series through neural networks forecasting with robust confidence intervals. *Neurocomputing*, *70*(1-3), 79–92.

Pagolu, V. S., Reddy, K. N., Panda, G., & Majhi, B. (2016). Sentiment analysis of twitter data for predicting stock market movements. In *2016 international conference on signal processing, communication, power and embedded system (scopes)* (pp. 1345–1350).

Pang, B., & Lee, L. (2008). *Opinion mining and sentiment analysis.* (Vols. 2(1–2)).

Pant, D. R., Neupane, P., Poudel, A., Pokhrel, A. K., & Lama, B. K. (2018). Recurrent neural network based bitcoin price prediction by twitter sentiment analysis. In *2018 ieee 3rd international conference on computing, communication and security (icccs)* (pp. 128–132).

Pavlyshenko, B. (2018). Using stacking approaches for machine learning models. In *2018 ieee second international conference on data stream mining & processing (dsmp)* (pp. 255–258).

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)* (pp. 1532–1543).

Phi, M. (2018). Illustrated guide to lstm's and gru's: A step by step explanation. *URL https://towardsdatascience. com/illustrated-guide-to-lstms-and-gru-sa-step-by-step-explanation-44e9eb85bf21*.

Picasso, A., Merello, S., Ma, Y., Oneto, L., & Cambria, E. (2019). Technical analysis and sentiment embeddings for market trend prediction. *Expert Systems with Applications*, *135*, 60–70.

Pota, M., Ventura, M., Catelli, R., & Esposito, M. (2021). An effective bert-based pipeline for twitter sentiment analysis: A case study in italian. *Sensors*, *21*(1), 133.

Pozzi, F. A., Fersini, E., Messina, E., & Liu, B. (2016). *Sentiment analysis in social networks*. Morgan Kaufmann.

Prechelt, L. (1998). Early stopping-but when? In *Neural networks: Tricks of the trade* (pp. 55–69). Springer.

Qasem, M., Thulasiram, R., & Thulasiram, P. (2015). Twitter sentiment classification using machine learning techniques for stock markets. In *2015 international conference on advances in computing, communications and informatics (icacci)* (pp. 834–840).

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., & Huang, X. (2020). Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 1–26.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Rahman, S., Irfan, M., Raza, M., Moyeezullah Ghori, K., Yaqoob, S., & Awais, M. (2020). Performance analysis of boosting classifiers in recognizing activities of daily living. *International journal of environmental research and public health*, *17*(3), 1082.

Rao, T., Srivastava, S., et al. (2012). Analyzing stock market movements using twitter sentiment analysis.

Rosenthal, S., Farra, N., & Nakov, P. (2017). Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th international workshop on semantic evaluation (semeval-2017)* (pp. 502–518).

Sewell, M. (2008). Ensemble learning. *RN*, *11*(02), 1–34.

Singh, T., & Kumari, M. (2016). Role of text pre-processing in twitter sentiment analysis. *Procedia Computer Science*, *89*, 549–554.

Smailović, J. (2015). *Sentiment analysis in streams of microblogging posts* (Unpublished doctoral dissertation). PhD Thesis, Jozef Stefan International Postgraduate School. Ljubljana, Slovenia.

Souma, W., Vodenska, I., & Aoyama, H. (2019). Enhanced news sentiment analysis using deep learning methods. *Journal of Computational Social Science*, *2*(1), 33–46.

Sprenger, T. O., Tumasjan, A., Sandner, P. G., & Welpe, I. M. (2014). Tweets and trades: The information content of stock microblogs. *European Financial Management*, *20*(5), 926–957.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Thelwall, M., & Buckley, K. (2013). Topic-based sentiment analysis for the social web: The role of mood and issue-related words. *Journal of the American Society for Information Science and Technology*, *64*(8), 1608–1617.

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017). Using deep learning to detect price change indications in financial markets. In *2017 25th european signal processing conference (eusipco)* (pp. 2511–2515).

Valencia, F., Gómez-Espinosa, A., & Valdés-Aguirre, B. (2019). Price movement prediction of cryptocurrencies using sentiment analysis and machine learning. *Entropy*, *21*(6), 589.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Vergin, R. C., & Qoronfleh, M. W. (1998). Corporate reputation and the

stock market. *Business Horizons*, *41*(1), 19–27.

Wen, M., Li, P., Zhang, L., & Chen, Y. (2019). *Stock market trend prediction using high-order information of time series.*

Wu, H., & Zhao, J. (2020). Self-adaptive deep learning for multimode process monitoring. *Computers & Chemical Engineering*, *141*, 107024.

Wuthrich, B., Cho, V., Leung, S., Permunetilleke, D., Sankaran, K., & Zhang, J. (1998). Daily stock market forecast from textual web data. In *Smc'98 conference proceedings. 1998 ieee international conference on systems, man, and cybernetics (cat. no. 98ch36218)* (Vol. 3, pp. 2720–2725).

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Zhang, L., Wang, F., Xu, B., Chi, W., Wang, Q., & Sun, T. (2018). Prediction of stock prices based on lm-bp neural network and the estimation of overfitting point by rdci. *Neural Computing and Applications*, *30*(5), 1425–1444.

APPENDIX A: LSTM ACCURACY SCORES

Variables	**Amazon**	**Apple**	**Google**	**JP Morgan**	**Facebook**	**Microsoft**	**Berkshire Hathaway**
Only predicting up	55.21	54.62	55.81	50.44	53.43	56.11	53.03
Past prices (PP)	47.83	53.36	52.17	47.83	49.41	47.83	50.99
PP + MA	53.36	59.68	62.06	57.71	61.26	59.29	55.73
PP + DEA	45.85	54.94	51.78	49.01	47.03	47.83	53.36
PP + RSI	50.59	54.94	55.73	47.43	46.64	55.73	58.89
PP + MA + DEA	53.76	56.52	62.06	58.10	59.29	58.50	56.92
PP + MA + RSI	53.36	54.94	57.71	56.13	51.78	59.29	59.29
PP + RSI + DEA	48.62	54.54	53.75	46.64	48.22	54.94	54.94
PP + MA + RSI + Diff	55.32	56.12	57.31	52.96	54.54	56.52	56.92
PP + Sent	54.94	52.96	54.94	49.01	48.61	54.94	53.75
PP + Vol	45.45	54.54	52.56	47.82	48.22	49.80	55.73
PP + Sent + Vol	45.85	51.78	53.36	49.01	48.22	50.99	54.94
PP + Sent + MA	57.31	**60.87**	**60.47**	56.13	**63.24**	**59.69**	52.96
PP + Sent + RSI	48.22	53.36	54.94	50.20	49.40	50.59	52.56
PP + Sent + DEA	46.64	51.78	50.20	47.83	44.66	50.99	53.76
PP + Vol + MA	57.70	60.24	58.89	59.68	61.66	58.10	56.52
PP + Vol + RSI	47.62	53.46	57.96	50.23	48.47	50.62	60.06
PP + Vol + DEA	45.85	53.36	57.31	49.80	48.61	48.22	55.73
PP + Sent + Vol + MA	**57.71**	60.87	60.01	55.33	62.85	58.10	56.52
PP + Sent + Vol + RSI	48.62	53.36	50.99	50.59	49.01	52.17	54.54
PP + Sent + Vol + DEA	47.43	52.57	54.94	47.03	49.40	48.62	51.38
PP + Sent + MA + RSI	55.34	56.92	57.71	51.78	55.73	58.10	60.08
PP + Sent + MA + DEA	53.36	56.13	59.68	55.34	56.13	58.89	58.50
PP + Sent + RSI + DEA	47.81	52.65	52.32	48.44	46.12	52.12	56.80
PP + Vol + MA + RSI	55.73	54.94	58.89	55.73	54.54	58.89	57.71
PP + Vol + MA + DEA	57.70	58.89	59.68	52.17	54.15	58.50	57.31
PP + Vol + RSI + DEA	47.03	54.15	53.36	47.03	48.22	55.73	51.78
PP + Sent + Vol + MA + RSI	53.76	55.73	58.10	52.17	56.12	58.10	57.31
PP + Sent + Vol + MA + DEA	56.52	56.52	59.68	**59.68**	51.38	58.50	**61.66**
PP + Sent + Vol + RSI + DEA	47.43	54.94	52.96	47.04	48.22	55.34	55.34
PP + Sent + MA + RSI + DEA	52.56	56.52	56.92	56.52	52.17	57.71	58.50
PP + Vol + MA + RSI + DEA	56.52	57.70	57.31	52.96	52.57	58.10	56.52

Variables	**Amazon**	**Apple**	**Google**	**JP Morgan**	**Facebook**	**Microsoft**	**Berkshire Hathaway**
PP + Sent + Vol + MA + RSI + DEA	54.15	55.73	58.89	52.97	52.96	56.12	57.70

Table 15: LSTM scores