Department of Computer Science

Submitted in part fulfilment for the degree of MEng.

Predicting the direction of stock price movements from historical data and sentiment analysis of Tweets

Matthew Turton Parry

30th April 2019

Supervisor: Daniel Kudenko

Contents

List of Figures

List of Tables

Executive Summary

In this paper, machine learning and sentiment analysis techniques are brought together in order to make financial predictions relating to the stock of Tesla Inc. In particular, long short-term memory neural networks (LSTM) are used to solve the binary classification problem of predicting the direction of daily Tesla stock price movements (either 'up' or 'down'). By using these LSTM's, an answer for the following question can be made; Does introducing sentiment information (relating to a stock) as an additional input to an LSTM help improve its accuracy when predicting the direction of daily stock price movements?

The problem of predicting the direction of daily price movements of financial assets with the help of sentiment information has been the focus of many previous works. Many of these papers use posts (tweets) from the microblogging website Twitter to generate their sentiment statistics. Out of all the previous works discussed in this paper, none of them use long short-term neural networks (LSTM) in order to carry out their predictions. However, it has been shown that LSTM's perform very well when learning from time series. Therefore, in this paper LSTM's will be used to predict the direction of price movements.

Firstly, tweets related to Tesla stock that were posted within a specified date range are gathered. The tweets are then cleaned of their Twitter specific content such as hashtags. A naive Bayes sentiment classifier is used to classify each of the gathered tweets are either positive or negative. A naive Bayes model was chosen due to its popularity and previous successes with binary sentiment classification. The classifier was trained on a Twitter corpus that contained examples of real positive and negative tweets, and achieved an accuracy of 79.5%.

Daily sentiment statistics were produced for the gathered tweets. For each day in the date range of interest, the percentage of positive tweets for that day was calculated. This collection of percentages formed a time series.

The historical price data time series of the Tesla stock was pre-processed before being used. Firstly, gaps in the price data such as weekends were interpolated. The difference in price between consecutive days was calculated and used from this point going forwards. This was used rather than the actual stock prices so that the time series would be stationary, an

ideal property of LSTM inputs. These values were also normalised such that the standard deviation was 1.

For each architecture in a range of LSTM architectures, two networks were produced, one taking the pre-processed historical price data time series as input (known as the baseline network), and the other taking this plus the additional daily sentiment statistics time series as input (known as the Twitter network). Each network was initialised, trained and tested 10 times, and an accuracy and confusion matrix was recorded for each. A mean accuracy and standard deviation for the baseline and Twitter networks could then be calculated per architecture. One of the hyperparameters that changed between architectures was the LSTM time lag, which corresponds to the number of elements of a time series the network may see at any one time.

With an LSTM time lag of 5 all baseline networks performed extremely poorly, with all except 2 achieving an accuracy below 50%. Although the mean accuracies for the Twitter networks were not as high as seen in other time lags, they remained at an acceptable level. The LSTM time lag of 4 recorded the highest mean accuracy for both the baseline (60.9%) and Twitter (60.3%) networks. However, there were large inconsistencies with the mean accuracies of each Twitter network, showing that LSTM hyperparameters have a large impact in accuracy. The baseline networks had a smaller range of mean accuracies. When the LSTM time lag was 3, it was the baseline networks that showed inconsistencies between architectures. In most cases with this time lag, the Twitter networks outperformed their baseline counterparts.

The findings suggest that although the sentiment statistics may not improve the accuracy of a baseline architecture that performs relatively well (>55%), they do help the network achieve a minimum acceptable accuracy when baselines perform particularly badly. One point of view may be to say sentiment statistics should be included in an LSTM network solving the problem at hand because the findings suggest they will maintain a minimum level of accuracy. On the other hand the highest accuracy recorded was by a baseline network, and so a search for the optimal hyperparameters for baseline networks is all that is required to solve the classification problem.

There are many directions future work could take. One suggestion is to repeat the experiment with more than one years worth of data (as used here), and aim to gather more tweets per day. Another change could be to gather tweets relating to the company as a whole, and not just their stock, as this was the way many related works gathered their tweets. Repeating the experiment multiple times, each time with a different company to focus on would allow more general conclusions to be drawn.

1 Introduction

Forecasting and predicting has been at the forefront of the financial world throughout its existence. The prospect of being able to predict the future price movements of commodities and stocks has lured many businesses and investors into this area. In the foreign exchange market alone, non-financial entities contribute to around $400 billion of the $5 trillion (USD) that is traded daily [1].

With the progression of technology in the last couple of decades it is possible to write computer software to attempt to find patterns in the historical price data of financial assets, in order to help predict future price movements. Prior to this, more traditional forecasting methods were used which were slower and required much more domain knowledge. Software for financial forecasting falls under the category of artificial intelligence, and in particular, machine learning. Machine learning, as defined by Kevin Murphy in his MIT press book, is "a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or perform other kinds of decision making under uncertainty" [2]. If the historical price data of financial assets is passed to a machine learning algorithm, then perhaps patterns will be discovered allowing future price predictions to be made. Machine learning algorithms are not limited to just one input, and so additional information related to the asset could be passed as inputs, to potentially uncover deeper patterns and improve predictability.

When it comes to stocks, investor opinion can have a large impact on price movements. For instance, if investors believe that a company they are invested in is going to have a worse than expected earnings report, then they may be inclined to sell their stock before the earnings are announced. This will increase the the supply of stocks, and consequently lower the stock price. It is reasonable to assume that some investors in this scenario would share their thoughts on social media. Social media posts may contain an insight into investor opinion, which if extracted correctly, may be a powerful datum to pass to a machine learning algorithm.

Sentiment analysis is the name given to techniques that attempt to extract meaning from text, representing this quantitatively [3]. This form of analysis could be applied to the social media posts of investors to understand their opinions. The resulting statistics from applying sentiment analysis could be passed to machine learning algorithms, potentially improving their

prediction accuracies.

In this paper, machine learning techniques and sentiment analysis are brought together to attempt to predict the direction of stock price movements. Many types of artificial neural network architectures will be tested, all attempting to predict the price movements of the same stock. For each architecture two networks will be created, one taking the historical price data as input, and the other taking both the historical price data and sentiment statistics from the analysis of Twitter posts related to the stock. This paper is an extension of previous works that have attempted to carry out similar experiments, the difference here being that long short-term memory neural networks will be used.

2 Literature Review

2.1 Financial forecasting

According to the Efficient Market Hypothesis (EMH) the price of a financial asset reflects all available information about it, and future prices follow a random walk making them unpredictable [4]. Intuitively, if financial assets were predictable then investors would be able to generate unlimited wealth. Financial time series data is usually incredibly noisy [5], which may be a leading factor to the unpredictable nature of financial assets.

The EMH is controversial, as many banks, hedge funds and investors rely on accurately predicting price movements of financial assets. Paul J. Darwen demonstrated that his feature selection software could find a detectable difference between true stock data, and random data that had been sampled from a log-normal distribution of the true data [6]. This implies that there is an element of predictability within stock price time series data, contradicting the EMH.

The EMH will be tested in this paper, as an attempt will be made to predict the price movements of a financial asset. The type of financial assets that will be predicted are stocks.

2.1.1 Stocks

A stock is a share of ownership in a company that can be bought and sold on an exchange [7]. The demand and supply for a stock governs its price, which varies over time for numerous reasons. Better than expected company earnings will make the stock more attractive to investors, increasing the price. Stock dilution, when a company issues new stock, will lower its price due to both the increase in supply and the negativity surrounding new stock offerings [8].

Public opinion and interpretation of news also affects stock prices. Gyozo Gidofalvi trained a naive Bayes text classifier to predict either up, down or unchanged price movements of stocks, given news articles related to them [9]. Although overall predictability was low, there was a significantly better success rate in predicting stock price movements within 20 minutes

of news articles being released to the public.

2.1.2 Traditional forecasting methods

Traditionally, there have been two main methods when it comes to making financial predictions; technical analysis and fundamental analysis [5]. Fundamentalists tackle the prediction problem from an economists point of view. Their intrinsic belief is that the laws of demand and supply are the only factors that will affect stock prices. As such, they analyse macroeconomic data and the news to determine how demand and supply may be changing [10]. Examples of data that will be studied include inflation rates, overall stock market performance and central bank policies.

On the other hand, technical analysts pay no attention to economic factors or news, and instead focus solely on the historical price and volume data of their chosen financial asset [11]. They justify this by assuming that news and events are fully reflected in market prices, meaning they use fundamentals indirectly. The main assumption of a technical analyst is that historical price patterns repeat themselves over time. If investors can detect a newly forming pattern, they will be able to exploit this by making early trades to beat the market. However, these patterns are very subjective and hard to detect before they are complete, at which point little to no profit can be made [12].

With the advancement in computing power over the last few decades, machine learning algorithms are being utilised to learn from and predict price movements. Inputs to such algorithms may be purely technical or fundamental, or a combination of the two. One popular type of machine learning algorithm used for financial predictions is the artificial neural network.

2.2 Artificial neural networks

Artificial neural networks (ANN) aim to model biological neural networks abstractly, by representing neurons as vertices in a directed graph [13]. The aim of an ANN is to be able to learn and make predictions, based on previous observations. They can be trained to solve regression problems, where predictions are real values, or classification problems, where predictions categorise the input into one of many classes.

Warren McCulloch and Walter Pitts introduced the first artificial neuron (the TLU) in their 1943 paper [14], and since then countless neuron and network designs have been developed, solving an array of different problems.

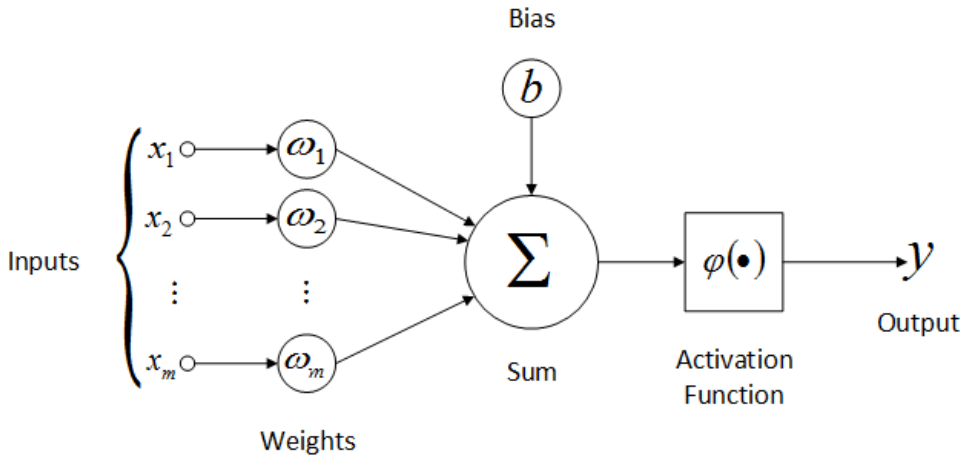

Figure 2.1: The structure of a perceptron [1]

Figure 2.1 shows the structure of the most common type of artificial neuron, the perceptron [15]. The inputs (connections) of a perceptron have a weighting associated with them. When an impulse (real value) travels through a connection its magnitude is multiplied by the weight of that connection. The weighted sum of the inputs is calculated and passed to the activation function, which may be any arbitrary mathematical function which takes a single input and produces a single output. Each perceptron has a single bias input. This input will always have the value of 1, however the weight associated with it may change throughout the learning process. The bias is required as it has the ability to shift the activation function output, much like a bias is required in the equation of a straight line, $y = ax + b$.

The chosen activation function dictates the behaviour of the perceptron. Although any arbitrary function can be used, there is a small collection of commonly used activation functions [16] [17]. Some of these functions are shown below.

$$\varphi(x) = \frac{1}{1+e^{-x}} \qquad \varphi(x) = \frac{\sinh(x)}{\cosh(x)} \qquad \varphi(x) = e^{-|x|}$$

The first is the sigmoid function, which produces outputs in the range of 0 to 1. This is often used in the last perceptron of a network solving a binary classification problem, whose output represents the probability of the input belonging to class one [18]. Hyperbolic tangent is the next function, which has outputs in the range -1 to 1. These are often used in hidden layers to normalise the internal representation of the network [19]. Finally the last function shown above is an exponential.

[1] https://www.researchgate.net/publication/320270458/figure/fig1/AS:551197154254848@1508427050805/Mathematical-model-of-artificial-neuron.png

2.2.1 Feedforward neural networks

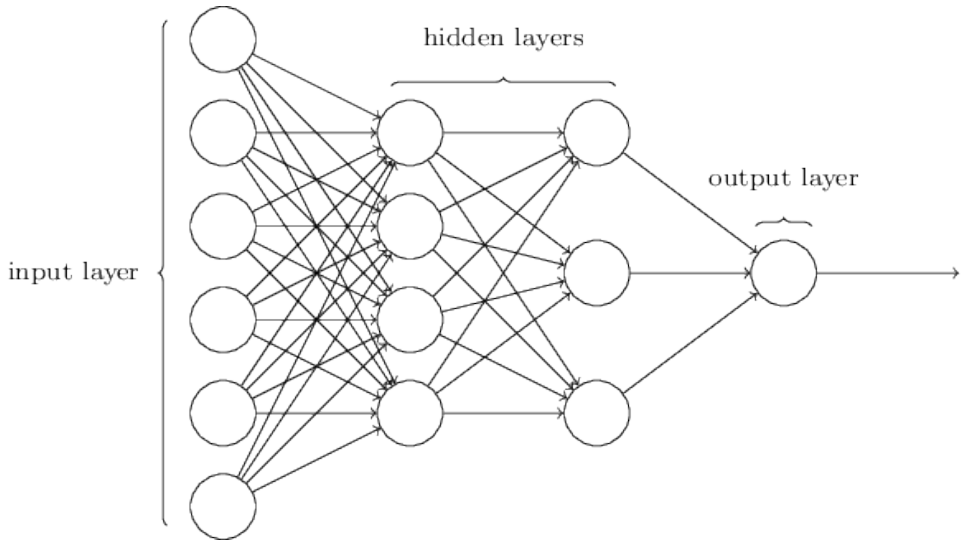

Figure 2.2: The structure of an artificial neural network [2]

An ANN is constructed by arranging perceptrons (nodes) together into an ordered set of groups called layers [13]. Every node from one layer is connected to every node in the next consecutive layer. The first layer is known as the input layer and is where data enters the network, one feature per node. The final layer is known as the output layer, and is where the prediction of the network is obtained. As data enters the input layer, the individual perceptrons calculate their output and then propagate this to the next layer, repeating until the output of the final layer is determined.

A feedforward neural network is an ANN that contains no cycles. Such a network is shown in figure 2.2. The output of one layer does not affect the output of any preceding layers; in other words, there is no feedback [20].

2.2.2 Recurrent neural networks

Recurrent neural networks (RNN), popularised by John Hopfield, are ANN's that contain at least one cycle [21]. Cycles may be within a single layer, or from one layer to a preceding layer. Impulses are sent through cycles after a time delay (usually one forward pass through the network) such that there are never any infinitely updating loops. RNN's are often used when at least one of the features is a time series, as the cycles give the network the ability to detect patterns within sequences [22].

[2] http://neuralnetworksanddeeplearning.com/images/tikz11.png

2.2.2.1 Long short-term memory neural networks

One of the biggest drawbacks with standard RNN's is their inability to remember information over large intervals [23]. This is because during learning, error gradients have a tendency to either get extremely large or vanish completely over a large number of cycle iterations [24]. Devised by Sepp Hochreiter and Jurgen Schmidhuber, long short-term memory neural networks (LSTM) are an extension of standard RNN's. They combat the drawbacks of standard RNN's by having the ability to retain information about distant data points, up to 1000 steps away [25].

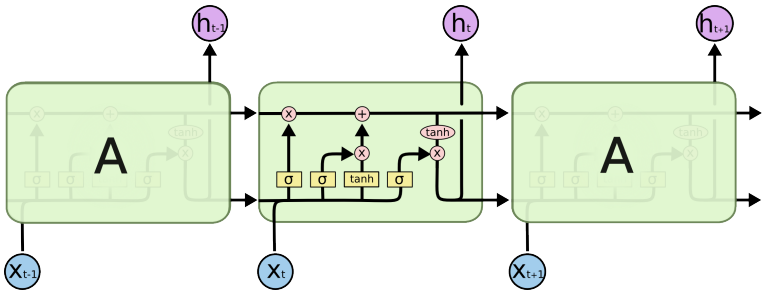

Figure 2.3: A long short-term memory neuron [3]

An LSTM cell can be seen in the middle of figure 2.3. The cell uses a series of gates to control its internal memory, which consists of representations of previously seen data points. Although any activations may be used at the gates of a cell, the original paper uses the hyperbolic tangent and sigmoid functions.

Figure 2.3 as a whole shows a complete LSTM neuron, which is made up of a finite number of cells (in this example, 3). When an input is given to an LSTM neuron, it must be in the form of a time series. Formally, an input X to a neuron at time t of a time series s is a list of values, the length of which is defined by a time lag factor n.

$$X_t = \{s_t, s_{t-1}, \ldots, s_{t-n}\}$$

Each cell in the LSTM neuron is responsible for handling one of the elements of the list X_t.

2.2.3 Training

When solving a supervised learning problem, many instances of input-output pairs need to be collected that are representative of the real world

[3] http://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-chain.png

data [26]. This collection is often referred to as the training set. When training ANN's, each input of the training set is fed through the network and an error between the network's predicted output and the true output is measured [27]. There are several ways an error can be calculated. One of the most popular ways is to use the root mean squared function, which works particularly well for regression problems. Cross-entropy loss, also known as log loss, is a specialised error function for networks that predict a single probability in the range 0 to 1. This makes it very useful for binary classification problems.

Once errors are measured they can be used to train a network. Training a network involves an algorithm known as backpropagation.

2.2.3.1 Backpropagation

For his 1974 thesis, Paul Werbos formulated a gradient descent algorithm called backpropagation [28]. The algorithm can be applied to a function that attempts to match a set of inputs to a set of corresponding outputs. For each input, backpropagation calculates the gradient of the error produced by the function.

It wasn't until 1986 that backpropagation was applied to ANN's. David Rumelhart, Geoffrey Hinton and Ronald Williams published a paper demonstrating how the backpropagation algorithm could be used to train ANN's [29]. Previous training algorithms such as the perceptron-convergence procedure were unable to train networks with hidden layers [30], however, backpropagation could now be used in conjunction with new algorithms to train networks of any topology. Some problems, such as non-linearly separable classification, require at least one hidden layer in the network, which made backpropagation a breakthrough.

2.2.3.2 Optimisers

To train an ANN, optimiser algorithms are required. Optimisers aim to change the weights of a network based on the error gradients calculated by backpropagation. One of the most popular optimisers is the stochastic gradient descent algorithm (SGD) [31]. For each weight, a factor (known as the learning rate) of the error gradient is subtracted. This is repeated many times, recalculating the gradient of the error at each iteration, until terminating criteria are met.

Many other optimisers exist. AdaGrad works in the same way as SGD, except each weight has its own learning rate, rather than a global learning rate [32]. This often outperforms SGD when the training data is sparse. RMSProp is a similar algorithm to AdaGrad, which also features per weight

learning rates [33]. Adam is an extension of both AdaGrad and RMSProp, and has been shown to perform better than both of these on a range of different examples [34].

ANN's take real values as inputs, however, other machine learning algorithms exist that can learn from other types of data. One example that will be looked at in this paper are sentiment analysis algorithms, which are used to extract the meaning of passages of text.

2.3 Sentiment analysis

Natural language processing (NLP) is an area of artificial intelligence concerned with the interaction between humans and computers via a natural language [35]. There are many applications of NLP, for example, voice dictation. In this case a computer needs to understand and convert sound signals spoken by the user into words. A major challenge of this is that there exist many words that are pronounced the same way, but have different spellings and meanings. The computer must be able to identify the intended meaning of the user, by attempting to understand the context of what is being said. Statistical models are used to represent natural languages, which can be trained over time using machine learning algorithms [36].

Sentiment analysis is a form of natural language processing which attempts to extract the subjective meaning and emotion of passages of text [3]. Sentiment analysis can be applied to any length of text to predict a range of different properties, such as polarity which represents the extent to which the text is positive, and subjectivity which represents the degree to which the text is fact or opinion.

2.3.1 Types of sentiment models

Before statistical models were introduced, rule based approaches were used to analyse natural language [37]. These rule based methods often employed grammars to represent features of natural language. Due to the nature and ambiguities of natural language, such models often performed poorly, as described by Roger Schank and Robert Abel [38]. They suggest the reason for this is because rule based approaches do not take into account the context of the text.

Statistical models on the other hand tend to perform much better [36]. They usually require at least two sets of data; lexicons and text corpora [39]. A lexicon is a word list, like a dictionary, containing every possible word that could be encountered. Each word in a lexicon often has additional

information associated with it. This usually includes its part-of-speech which defines the role of the word, for example, a noun. A text corpus is a large body of text that is representative of the language being analysed, and often contains extra information about the individual sentences or examples within it. A text corpus, for example, could be a collection of sentences, where each sentence has been manually classified as positive or negative.

Even with a lexicon and text corpus to use as training material, there are still some issues that can arise due to the nuances of natural language [39]. For instance, a word deemed to contain emotion by a lexicon, such as "good", may be used in a sentence with no sentiment, such as "Is the 2018 MacBook Pro good?". Another example is sarcasm, which often exaggerates the opposite of the true opinion.

One highly used statistical model, also used in machine learning, is the naive Bayes classifier [36]. With this model a set of features is defined, where each feature represents a word or phrase. The set of features may be a lexicon, or a subset thereof. The value of each input feature may represent the number of times that word or phrase is found in the input text, or alternatively, if the word or phrase is present or not.

2.3.2 Word representation

Although words alone may be used as features for sentiment classification algorithms, there are other representations that may be more useful. N-grams are sequences of words, N in length, where N may be any positive integer [39]. Using the number of times an N-gram is present in an input as a feature could be used. Using N-grams as features can help the classifier identify the context of the text by viewing more than one word at a time, potentially overcoming some of the issues previously discussed.

Vector representation of words is also a popular choice. In this representation, words within a text corpus are assigned a vector in a high dimensional space (often many hundreds of dimensions) [40]. Words that are deemed similar, such as "London" and "England" have a small distance between them within the vector space. This allows the models to use similarities between words to help solve problems.

2.3.3 Training sentiment models

Like any supervised learning problem, training examples representative of the real world problem need to be collected. With sentiment analysis it often takes a long time to collect such data because each natural language example must be read and analysed by a human first.

Once a text corpus has been created it can be used as a training set for a classifier. One of the first works to do this was by Bo Pang, Lillian Lee and Shivakumar Vaithyanathan, in which they applied a naive Bayes classifier (amongst other machine learning models) to the sentiment classification of movie reviews [41]. They achieved a classification accuracy of 78.8%. For reference, in a paper where 3 independent individuals had to classify text into 3 categories, they were all in agreement only 36.7% of the time, and 2 out of the 3 agreed 50.1% of the time [42]. In 13.2% of cases they all disagreed, highlighting the ambiguous nature of sentiment classification. In general, it has been shown that humans agree on the sentiment of text samples only 70-79% of the time [43]

Prior to the internet, one of the only ways of gathering large quantities of natural language data was to issue surveys to the public or digitise books. With today's technology, internet sites that are rich with natural language such as forums and social media can be mined to create text corpora. Alexander Pak and Patrick Paroubek demonstrated how the social media website Twitter could be used to create a text corpus, which could be later used to train a sentiment classifier [44].

2.3.4 Twitter as a corpus

Twitter is a microblogging service where users may create and share posts known as tweets. Tweets have a character limit of 280, meaning they must be short and concise. On the whole this is an advantage for sentiment analysis, as the amount of text to be analysed is short and there will usually only be one subject being discussed. Twitter boasts 126 million active users daily [45], in addition to 500 million tweets posted per day [46]. All these factors, along with their powerful API, makes Twitter an excellent source of public sentiment [47].

2.4 Related works

The problem of predicting stock prices based on historical price data and additional features has been the focus of many papers.

One of the founding research papers in this field was by Johan Bollen, Huina Mao and Xiao-Jun Zeng, in which they used a 5 layer self-organising fuzzy neural network to predict daily movements of the Dow Jones Industrial Average (DJIA) [48]. They gathered nearly 10 million tweets posted over a year long period (2009) that contained key phrases such as "I feel" or "I am". This was in an effort to only collect tweets containing emotion directly from the author. Tweets were sent through a cleaning algorithm where

Twitter specific content such as hashtags and usernames were removed, as well as URL's and other non-semantic information. Two sentiment analysis algorithms were used, OpinionFinder (OF) and Google Profile of Mood States (GPOMS). OF classifies text as either positive or negative, whereas GPOMS is an extension of a popular psychological rating scale named POMS [49] and classifies text into 6 dimensions; calm, alert, sure, vital, kind, and happy. Sentiment time series were produced. The OF time series was the ratio of positive to negative tweets for each day. GPOMS had 6 time series, one for each dimension, whose values represented the average of that dimension for each day.

A baseline network was created, which took 3 consecutive DJIA values as input in order to predict the next DJIA value. 3 values were chosen because Granger causality analysis showed a correlation between DJIA values and some of the GPOMS dimensions that were lagged behind by 3 days. This baseline network predicted the direction of DJIA movements with an accuracy of 73%. Many network permutations were tested when it came to using the sentiment analysis results as input. When 3 consecutive daily DJIA and OF values were used as input, the accuracy of the network did not change. However, the accuracy of the network grew significantly to 86% when DJIA values and the calm dimension of the GPOMS analysis were used as input. Other input permutations, such as DJIA values, the calm dimension and the sure dimension made prediction accuracy worse than the baseline, in this example 46%.

Many have extended from this work, such as Anshul Mittal and Arpit Goel [50]. In their paper, tweets were gathered and cleaned in the same way, however, the sentiment analysis algorithm classified tweets into 4 dimensions; calm, happy, alert and kind. Many types of machine learning algorithms were tested, including self-organising fuzzy neural networks, linear regression, logistic regression, and support vector machines. Extra data preprocessing steps were made on the DJIA time series. The overall range of DJIA values was made smaller by shifting stationary sections of the time series closer together. Periods of large volatility after the shift were also removed. Although this likely increased the performance of the machine learning models in training, this presumably made them generalise worse with unseen DJIA values as the training data was selective. They found that neural networks outperformed the other types of machine learning models across the board, with the highest average direction accuracy being 75% (with 3 consecutive DJIA, calm dimension and happy dimension values as input).

Tushar Rao and Saket Srivastava also investigated using machine learning models to predict DJIA movements, this time looking at the weekly time scale [51]. They used a naive Bayes classifier to classify tweets as either positive or negative. Instead of a neural network, they used a support vector machine (SVM) with 8 vectors to predict whether the next weekly

DJIA value will increase or decrease from the previous week's DJIA value. Some of the inputs to the SVM were engineered, such as the bullishness feature, which grew larger when the ratio of positive to negative tweets increased. They achieved accuracies as high as 91%.

Others have looked at predicting the movements of single stocks, rather than market wide indices. Venkata Pagolu, Kamal Challa and Ganapati Panda looked at predicting Microsoft stock price movements [52]. Tweets related to Microsoft were gathered and cleaned in the same way as the first paper [48], however they were transformed into two different representations. The first was Word2Vec, where each word is mapped to a 300 dimensional vector were related words are close together. The other was an N-gram representation. Many classifiers, such as random forest and logistic regression were trained using these representations of tweets to predict whether a given tweet is positive, negative or neutral. A variety of machine learning algorithms, which took the last 3 days worth of sentiment analysis results as input features were trained to predict the direction of the next daily stock price. Using an SVM, an accuracy of 71% was recorded. Other models performed worse, such as logistic regression with an accuracy of 69%.

LIi Bing, Keith Chan and Carol Ou carried out a very similar piece of research, looking at the stock of 30 companies listed on the NASDAQ exchange [53]. 15 million tweets posted from the US relating to the 30 companies were gathered and transformed into a vector representation. Tweets were placed into 1 of 5 categories based on a formula taking each word vector from the tweets as input. Once again, it was found that SVM's outperformed other models such as a naive Bayes classifier, with an average accuracy of 76%.

In all the papers discussed so far, adding certain permutations of sentiment inputs to the machine learning algorithms improved their accuracies. Others have looked at using technical indicators as input to their models, in place of sentiment analysis statistics. David Nelson, Adriano Pereira and Renato de Oliveira showed how LSTM's can be applied to predict the price movement directions of Brazilian stocks, by using historical price data and technical indicators as inputs [54]. The historical data was transformed such that it was stationary (constant mean and standard deviation throughout the time series). LSTM's outperformed the baseline models in almost all cases, with an average accuracy of 56%.

Jingtao Yao and Chew Lim Tan applied artificial neural networks to the foreign exchange market in order to predict exchange rate movements [55]. They found that if they gave their neural networks technical indicator inputs, such as a simple moving average, then the normalised mean squared error reduced significantly.

3 Problem Analysis

This paper attempts to find an answer to the following question:

Can Twitter sentiment related to a stock be used to improve the accuracy of a long short-term memory neural network, that predicts the direction of the daily price movements of that stock?

3.1 Extension of previous works

This problem has been looked at in some detail in a range of papers, discussed in the literature review. As such, this paper will be an extension of their works. In all of the papers discussed, artificial neural networks (ANN) and support vector machines (SVM) achieved the highest accuracies out of all the machine learning models tested. ANN's had higher accuracies than SVM's when they were tested in the same setting.

Most of the literature discussed used sentiment analysis as input features for their machine learning models. Some of those papers attempted to use ANN's to predict the movements of the Dow Jones Industrial Average [48] [50]. They used the same type of ANN, a 5 layer self-organising fuzzy neural network. Others looked at predicting the price movements of individual stocks, however they did not use ANN's [52] [53]. Some of the literature did not use any form of sentiment analysis, such as [54], in which a long short-term neural network (LSTM) was used to predict the direction of price movements of individual stocks, by using historical data and technical indicators alone.

To extend from these works this paper will use LSTM's to predict the price movements of an individual stock. These LSTM's will take historical price data and sentiment analysis as input features. LSTM's were chosen because they are designed to learn from time series data and remember information about distant data points, as discussed in the literature review. Not only this, but they have not yet been tested with sentiment inputs when it comes to predicting the direction of stock price movements. Therefore this paper will attempt to discover the accuracy of this untested method.

In all the previous works that required sentiment information, posts from the micro-blogging website Twitter were used. As a result this paper will

also gather and analyse Twitter posts (tweets) to calculate public sentiment information. Most of the previous works gathered tweets that weren't necessarily related to the stocks in question, however in this paper, only tweets that are related to the chosen stock will be collected. This ensures that the accuracies recorded in this paper are a consequence of the sentiment analysis of social media posts related to the chosen stock only, which allows the question stated at the start of this chapter to be answered. In most of the literature the data was preprocessed in the same way, such as how tweets were stripped of their Twitter specific content, therefore these same techniques will be employed here.

3.2 Procedure

To solve the problem at hand a particular stock needs to be selected, and in this paper Tesla Inc stock (TSLA) will be used. Tesla is a manufacturer of electric cars, and was chosen for a variety of reasons. Firstly, Tesla stock is extremely volatile, with an almost equal amount of net up and down daily movements. This minimises any skew in the target data; large skew often makes machine learning models perform worse with unseen data [56]. Tesla and its CEO Elon Musk have become household names, with both seen as controversial figures. Subsequently it should be expected that there exists a large quantity of tweets containing emotion related to the company.

A Twitter search query will be passed to the Twitter API to collect tweets containing emotion related to Tesla stock. A naive Bayes classifier will be trained using a tweet training set to classify them as either positive or negative. All the gathered tweets will be collected into groups, one for each day. The percentage of positive tweets for each day will be calculated, and these values will form a time series.

Two LSTM models will be created. They will attempt to solve a binary classification problem; for a given day, will the Tesla stock's closing price be higher or lower than the previous day's closing price. The first model will act as a baseline to discover prediction accuracy using historical stock price data alone, in the form of a time series. The second will determine whether having an additional input in the form of a Twitter sentiment time series can improve prediction accuracy.

3.3 Potential issues

Out of the related works that gathered tweets, most of them did so from a once publicly accessible database of 476 million tweets posted within 2009 [57]. This database has since been taken down at the request of Twitter, and consequently the tweets used here will have to be gathered manually using the Twitter API. The free tier of the premium API has large restrictions; a maximum of 5,000 tweets may be gathered per month [58]. The payments required to access higher tiers are not justifiable, and as such, due to the timescale of this experiment a maximum of 15,000 tweets will be gathered. This small quantity of tweets may not be enough to consider the results of this experiment valid.

3.4 Requirements

To ensure all goals are achieved, a set of requirements have been defined.

1. **Gather and prepare tweets related to Tesla Inc stock.**

   A search query will be defined. Using the Twitter API tweets will be gathered that match the query. Only tweets that were posted within a given date range will be collected. Tweets will be stripped of their Twitter specific content, such as hashtags and usernames. Non-meaningful content such as URL's will also be removed.

2. **Train a sentiment analysis algorithm and generate daily sentiment statistics for the gathered tweets.**

   A sentiment analysis model will be trained to classify tweets as positive or negative. Each gathered tweet will be classified. For each day the percentage of positive tweets will be calculated.

3. **Train two long short-term memory neural networks to solve the binary classification problem.**

   Both LSTM's will be trained to predict the net movement direction (up or down) of a stock price on a given day. The inputs to each network are given as:

   a) The previous X days worth of stock price data (Baseline network).

   b) The previous X days worth of stock price data and the daily sentiment analysis statistics (Twitter network).

   Many network architectures must be tested and comparisons made. Many values of X (the LSTM time lag) must also be experimented.

3.5 Ethical considerations

This project uses Twitter data that was generated by members of the general public. Although Twitter users acknowledge that their tweets will be made publicly available (via Twitter's terms and conditions), they have the right to delete their posts at any time. Tweets that have already been gathered may later be deleted by the author whilst the project is still ongoing. Once the project is complete, the tweet dataset that was collected will be destroyed.

4 Design and Implementation

4.1 Implementation tools

Many tools are available to carry out the experiment outlined in the previous section. Firstly, a programming language needs to be selected. Python is chosen because many industry standard data processing and machine learning libraries are available within it.

For creating and training artificial neural networks (ANN) the Keras library will be chosen. This is because Keras has the ability to build long short-term memory neural networks (LSTM), something other popular machine learning libraries such as scikit-learn don't have. Networks are constructed at the layer level with a large range of customisations available, such as the number of neurons and activation functions.

In terms of sentiment analysis, it is decided that the TextBlob library will be used to classify Twitter posts. This will be supported by the Natural Language Toolkit (NLTK) library, which contains text corpora used for training sentiment models.

4.2 Data collection

Before any machine learning can take place, the input data needs to be collected. This consists of Tesla's historical stock price data and Twitter posts (tweets) related to the company. Both of these types of data have dates associated with them; the date of the stock price and the date and time that each tweet was posted. Therefore a date range needs to be defined that is common between the two types of data. 1st January 2018 to 30th November 2018 was chosen. December was ignored due to the potential lack of trading and/or Twitter activity over the Christmas period.

Tesla Inc stock price data is to be gathered from Yahoo Finance [59]. Due to the restrictive Twitter API, a carefully designed search query needs to be defined. This query needs to gather enough tweets per day such that the daily sentiment analysis statistics are valid, but not too many such that the API request limit is not reached prematurely. The search query used is as follows: '$Tsla I think -http -www'. The minus sign + word indicates

that the word should not be present in the tweet, so in this case tweets with URL's are excluded. '$Tsla' is the symbol used to represent Tesla stock, so gathered tweets will be related to Tesla. 'I' attempts to get tweets written in the first person, directly from the author, and 'think' aims to get tweets containing some emotion. Similar techniques were employed in previous works [48] [50]. Retweets (when a user shares someone else's tweet) were ignored such that all gathered tweets were unique.

4.3 Data preprocessing

Stock exchanges are shut on weekends and certain bank holidays, therefore no stock price data is recorded on these days. However, tweets are gathered from a continuous timeline regardless of the day they were posted. As a result, stock price data must be interpolated where there is missing data. This will be achieved using the same techniques applied in [50] and [52]. If there is a gap in the data between two days with stock prices x and y, then the interpolated stock price of the first missing day in the gap is set to $(x + y)/2$. This repeats iteratively until there are no more gaps in the series.

ANN's often achieve greater accuracies when their input data is stationary [60], that is to say the mean and deviation is the same through time [61]. The way in which the stock price data will be made stationary is by taking the difference. By taking the difference a new series is produced, whose values represent the difference between each consecutive pair of values in the original series. As an example, given a time series $[3, 5, 9, 8, 2]$, its difference is $[2, 4, -1, -6]$.

Once all the tweets are collected they need to be cleaned. Tabs and new line characters will be replaced with single spaces. References to Twitter usernames and hashtags will be removed from the tweet. Similar procedures were made in other works [52].

After cleaning the tweets they will need to be passed to a sentiment analysis algorithm to classify them as positive or negative. A naive Bayes classifier will be trained using a Twitter corpus of 10,000 tweets that have been manually defined as positive or negative. Once each tweet has been classified, the percentage (as a decimal) of positive tweets for each day will be calculated. These values will form a daily sentiment time series.

Once the above preprocessing steps have been taken there will be two time series; one being the differenced daily stock prices and the other being the percentage of positive tweets for each day. Both time series will be aligned such that the last element of the stock price time series will be the difference between the Tesla stock's closing price on 29th November and

30th November 2018, and the last value of the sentiment time series will be the percentage of positive tweets on 30th November 2018. The length of these time series will be the same, such that they have a common start date.

4.4 LSTM neural networks

Two LSTM classification models will be constructed, one taking only the differenced daily stock price time series as input (baseline network), and the other taking both time series as input (Twitter network). The networks will be trained to predict whether the next day's closing stock price will be lower (class 0) or higher (class 1) than the previous day's.

When training LSTM's a batch size and number of epochs needs to be defined. Batch size is the number of data points passed through the network before the weights are updated. The number of epochs is the number of times every data point in the training set is passed through the network. The batch size will be fixed to 32. Having a much larger batch size can significantly reduced the generalisation quality of the network [62]. The number of epochs will be set to 1000. Having this high a number of epochs allows more weight updates, however this can lead to overfitting. To combat this each LSTM neuron will have a dropout rate of 20%, which is the chance that during the current batch its weights will not be updated or used in error calculations [63]. Having dropouts on LSTM neurons reduces the extent to which they overfit. The adam optimiser will be used to update the weights in the network at the end of each batch, due to the advantages stated in the literature review. A binary crossentorpy function will be used to compute the errors during training, as this is designed for binary classification problems.

One and two hidden-layer architectures will be tested. The input layer will be where the time series inputs enter the network; no weights are involved at this stage. The hidden layers will contain every permutation of 20, 30 and 50 LSTM neurons. The final layer will always consist of a single perceptron with a sigmoid activation function, to give outputs in the range 0 to 1. This output will represent the probability that the input belongs to class 1. An LSTM time lag of 3, 4 and 5 will be tested for each architecture. Each network will be tested 10 times and an average accuracy will be recorded. This is because the weights of each network are randomised initially, therefore repeating identical training on identical architectures can lead to different results. A confusion matrix, showing the number of false positives and false negatives will also be recorded. For each test the training data will be split sequentially 80%/20% into a training set and test set respectively.

5 Results and Evaluation

Results for each of the long short-term memory neural network (LSTM) time lags have been recorded. These results show the difference between the Twitter and baseline mean accuracies and standard deviations, where positive numbers indicate the Twitter network had a larger value than the corresponding baseline network. These results are analysed independently before overall evaluations are made. The full raw results can be found in appendix A. A confusion matrix for each network was recorded, by summing the confusion matrices from the 10 individual tests. A mean confusion matrix per LSTM time lag has been produced. Each architecture has been given a number to uniquely identify it, as shown in figure 5.1.

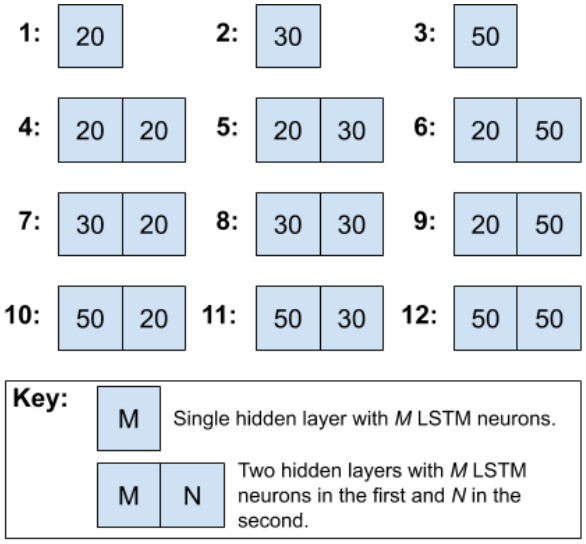

Figure 5.1: The identifier for each architecture

5.1 Sentiment classifier

The naive Bayes sentiment classifier was trained on 80% of the Twitter corpus, and tested on the remaining 20%. The classifier achieved an accuracy of 79.5%. 9,435 tweets were gathered that covered the whole date range of interest (1st January 2018 to 30th November 2018). The vast

majority of the gathered tweets were classified as negative. In 291 days in the date range of the tweets over half of the tweets were negative, whereas there was only 40 days were the opposite was true. This resulted in the daily sentiment statistics having a very small variance (values were typically in the range 0 to 0.5), meaning they may not provide much predictive power to the LSTM networks.

5.2 LSTM time lag of 3

Table 5.1 shows the difference in mean accuracy and standard deviation between the baseline and Twitter networks for each architecture. Figure A.1 visualises the mean accuracy for each architecture.

Architecture	**1**	**2**	**3**	**4**	**5**	**6**
Δ Mean accuracy	0.0	0.0	2.6	2.5	4.9	2.9
Δ Standard deviation	-0.66	-0.12	-0.08	0.33	0.74	-0.53
Architecture	**7**	**8**	**9**	**10**	**11**	**12**
Δ Mean accuracy	0.8	-1.0	0.9	-2.0	-3.6	-5.4
Δ Standard deviation	2.10	0.53	0.39	0.80	0.63	0.11

Table 5.1: The difference between Twitter and baseline mean accuracies and standard deviations for each architecture, with an LSTM time lag of 3

The single hidden-layer Twitter networks (architectures 1, 2 and 3) achieved the highest accuracies, in the range 58.2% to 58.9%, which were as good as or better than the corresponding baseline networks. They also boasted some of the smallest standard deviations recorded, showing that they consistently achieve these accuracies. It is suspected this is due to the simplicity of training a small number of LSTM neurons in a single hidden layer.

Architectures 4, 5 and 6 which consisted of 20 neurons in the first hidden layer saw the greatest boost in mean accuracy from baseline to Twitter, with increases in the range 2.5% to 4.9%. However, this is because the baseline mean accuracies for these architectures were the lowest (except one) of all networks with a time lag of 3. The reason for this may be due to the small internal representation of the inputs in the first hidden layer, which may be so small that valuable information is lost.

Standard deviations in the remaining architectures were higher in the Twitter networks compared to the baseline networks. For architectures 7, 8 and 9 there was very little difference in mean accuracy. The Twitter networks for architectures 10, 11 and 12 experienced by far the largest

drop in mean accuracy compared to the corresponding baselines, with a reduction up to 5.4%.

On the whole, the baseline networks improved in accuracy as more LSTM neurons were added in the first hidden layer, whereas the Twitter networks appeared to get worse. The reason for this may be because as more LSTM neurons are added in the first hidden layer, the baseline network can learn and represent more complex patterns in price movements. The reason the Twitter networks got worse with the same changes may be due to the fact that the sentiment statistics had more of an impact, but provided very little additional information because of the low variance.

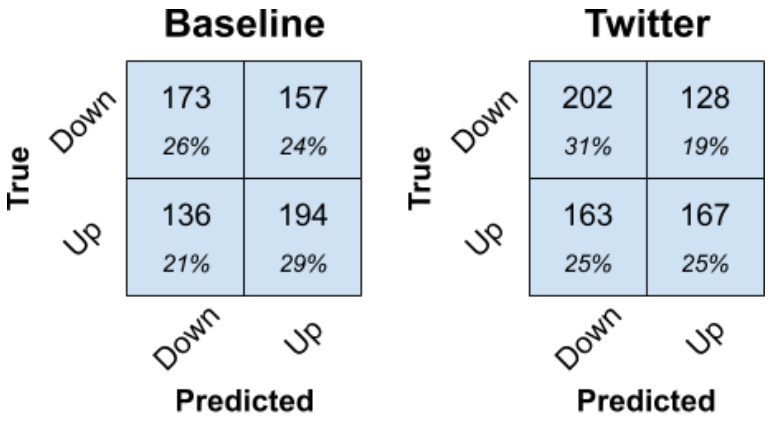

Figure 5.2: Mean confusion matrix for baseline and Twitter architectures with an LSTM time lag of 3

Figure 5.2 shows the mean confusion matrices. The Twitter networks had a large tendency to predict false negatives (25%) compared to false positives (19%). They also predicted down movements 56% of the time, compared to the 47% of the time for baseline networks. This suggests the negatively skewed sentiment statistics are making the Twitter networks predict down more often.

5.3 LSTM time lag of 4

Table 5.2 shows the difference in mean accuracy and standard deviation between the baseline and Twitter networks for each architecture. Figure A.2 visualises the mean accuracy for each architecture.

The Twitter networks for the single hidden-layer architectures (1, 2 and 3) performed much worse than in the baseline networks, with the largest drop in accuracy being 4.1%. Presumably, the increase in dimension of the network inputs (compared with a time lag of 3), and larger size of each input

(compared with the baselines) meant that a much larger single hidden layer would be required to find price movement patterns. This made the Twitter networks underfit. As the number of LSTM neurons increased through these architectures the standard deviation of the baseline networks also increased, meaning they became more inconsistent.

Architecture	**1**	**2**	**3**	**4**	**5**	**6**
Δ Mean accuracy	-3.2	-2.2	-4.1	2.1	2.9	-0.6
Δ Standard deviation	0.70	0.21	-1.66	-2.43	-1.02	0.26
Architecture	**7**	**8**	**9**	**10**	**11**	**12**
Δ Mean accuracy	-0.4	-1.1	-0.7	-3.3	-1.0	-3.4
Δ Standard deviation	-0.92	-0.24	1.69	0.96	-1.50	1.36

Table 5.2: The difference between Twitter and baseline mean accuracies and standard deviations for each architecture, with an LSTM time lag of 4

From architectures 4 to 6, all of the mean accuracies continuously increased, with architecture 6 achieving the greatest mean accuracy across all baseline and Twitter networks. For architecture 6 the Twitter mean accuracy was 60.3% and the baseline mean accuracy was 60.9%. It's worth noting that these were the highest accuracies across all networks tested (over all LSTM time lags).

For the remaining architectures (7 to 12) the baseline networks had a greater mean accuracy than their Twitter network counterparts, although most of the differences in mean accuracy were very small. Twitter networks for architectures 10, 11 and 12 saw a much reduced mean accuracy, close to the levels found in the first 3 architectures.

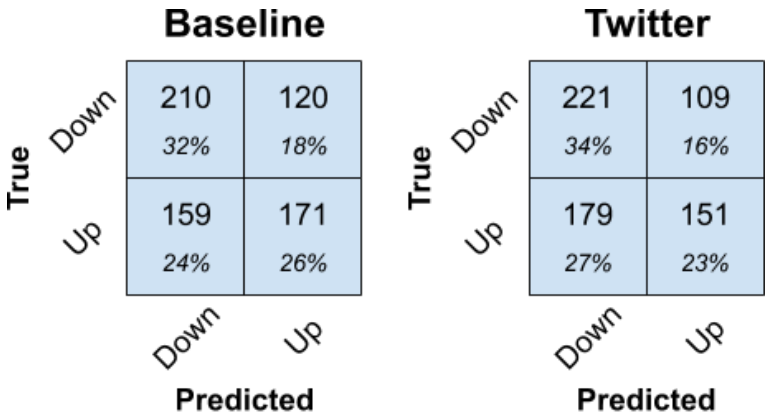

Figure 5.3: Mean confusion matrix for baseline and Twitter architectures with an LSTM time lag of 4

Overall, the baseline networks outperformed the Twitter networks, however, they achieved similar results when there were 20 and 30 LSTM

neurons in the first hidden layer. It is suspected that Twitter networks with 50 LSTM neurons in the first hidden layer dropped off in mean accuracy because there were not enough training data points to learn all of the parameters (network weights), and so these networks underfit. No relationship existed between architecture and standard deviation, although the range of standard deviations across the baseline networks was much larger compared to that of the Twitter networks.

Figure 5.3 shows the mean confusion matrices. Although both the baseline and Twitter networks predicted more false negatives (24% and 27%) than false positives (18% and 16%), the difference between the two numbers for the Twitter neworks is very large. This suggests that the training procedure was suboptimal in some way, such as an insufficient number of epochs, which made these networks underfit.

5.4 LSTM time lag of 5

Table 5.3 shows the difference in mean accuracy and standard deviation between the baseline and Twitter networks for each architecture. Figure A.3 visualises the mean accuracy for each architecture.

Architecture	**1**	**2**	**3**	**4**	**5**	**6**
Δ Mean accuracy	-0.3	2.8	6.4	2.2	3.2	6.7
Δ Standard deviation	2.89	1.39	0.89	-0.35	0.89	1.67
Architecture	**7**	**8**	**9**	**10**	**11**	**12**
Δ Mean accuracy	3.7	3.5	6.9	2.3	2.9	5.6
Δ Standard deviation	2.02	1.79	-0.71	-0.82	-0.59	2.35

Table 5.3: The difference between Twitter and baseline mean accuracies and standard deviations for each architecture, with an LSTM time lag of 5

One of the first things to note is that every Twitter network except the first one vastly outperformed the corresponding baseline network in terms of mean accuracy. At first it may be believed that having a time lag of 5 on sentiment statistics leads to far better LSTM results compared with other time lags. However, a review of the mean accuracies per network suggests that an LSTM time lag of 5 for the baseline networks made them perform extremely poorly, with most failing to achieve an accuracy above 50%. In other words, the baseline networks were no better than flipping a coin to predict between up and down price movements.

In addition, for the Twitter networks it was found that increasing the number of LSTM neurons in the second hidden layer lead to significant improvements in mean accuracy. For the single hidden layer Twitter networks

(architectures 1, 2 and 3), the mean accuracy increased as more LSTM neurons were added. This suggests that high accuracies, similar to those found in LSTM time lags of 3 and 4, may be achieved with a high number of LSTM neurons in the Twitter networks.

A potential reason why the Twitter networks performed much better than the baseline networks is because there was too much information (high dimensional inputs) for the baseline to learn. Although the Twitter networks should experience the same problem, each Twitter network input has a bigger chance of capturing a mostly positive day (a value over 0.5), compared with other LSTM time lags. Because these mostly positive days are quite rare in the sentiment statistics, they may provide predictive power to the LSTM networks.

An important point to highlight is the extremely high standard deviations across all tests, especially with the Twitter networks, which had a high of 6.75% and a minimum of 3.92%. Although this makes these networks inconsistent, all of the results gathered follow the pattern described previously.

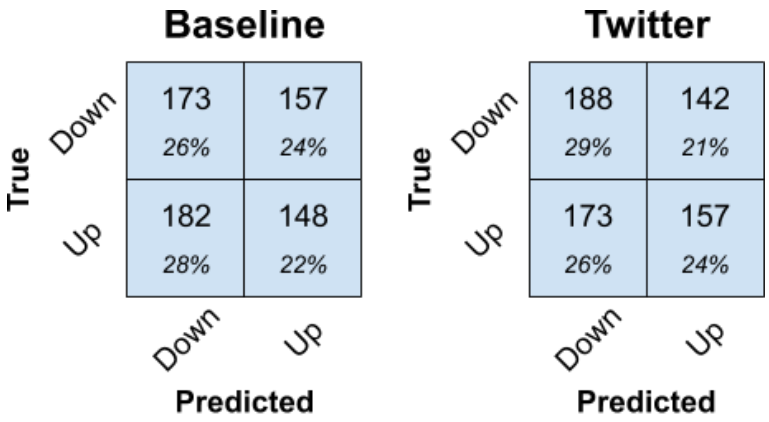

Figure 5.4: Mean confusion matrix for baseline and Twitter architectures with an LSTM time lag of 5

Once again, both the baseline and Twitter confusion matrices predicted more false negatives (28% and 26%) than false positives (24% and 21%), as shown by figure 5.4. The difference between these two numbers for the baseline and Twitter networks are very similar and small, therefore this could be attributed to noise or random chance.

5.5 Overall evaluation

Firstly, attention should be drawn to the naive Bayes sentiment classifier. The accuracy it achieved (79.5%) is very high, as humans typically only

agree on sentiment 70% to 79% of the time, as discussed in the literature review. It was also trained on a Twitter corpus, so the training examples were representative of the inputs the classifier received. However, there was a huge skew in the daily sentiment statistics that were produced, with 88% of the days having over half their tweets classed as negative. Although this could be a true reflection of the tweets gathered, the training of the sentiment classifier could be improved. Many of the gathered tweets contained financial terms, such as 'bearish' and 'bullish', which the Twitter corpus may not have contained. If the classifier was trained on a Twitter corpus that containined tweets relating to finance, then it could be said with much more confidence that the sentiment statistics produced were a true reflection of the gathered tweets.

In addition to this, there was a slight skew in the training set. 122 of the training data points belonged to the positive class 'up', and 139 belonged to the negative class 'down', resulting in a negative skew of 6.5%. Naturally, this makes the LSTM networks predict the negative class more often than the positive class, as the parameters (weights) have been updated more times with the negative class in mind. The test set had 0% skew, with 33 data points for each class, so this could not be blamed for any skew in predictions. When the mean confusion matrices were analysed it was found that in most cases, the networks predicted more false negatives than false positives. It could be suggested that this is a direct result of the skew in training data.

One of the main issues across all the networks is the small quantity of training data. Although a high number of epochs was used to ensure a large number of weight updates, the small number of training data points could lead to the networks not generalising very well, by potentially overfitting. Attempts were made to prevent this, by using a dropout on each LSTM neuron, however overfitting may still have been a problem.

Finally, due to the restriction of the Twitter API, only tweets relating to the stock of Tesla were gathered. The related works discussed in the literature review gathered tweets that were related to the company and their products and services, not just their stock. This provides a much more general sentiment about the company as a whole, and removes the need for the suggested improvement of using a financial Twitter corpus. The related works also gathered orders of magnitude more tweets, allowing them to gain a more detailed insight into public opinion. The gathered tweets in this paper were not geo-location restricted, however, maybe they should have been restricted to the United States as this is where Tesla is based.

6 Conclusion

As demonstrated in this paper, it is achievable to gather Tweets relating to the stock of a company and generate sentiment statistics from this, in the form of the percentage of positive tweets per day. It can be said with confidence that introducing these sentiment statistics along with historical price data as inputs to a long short-term neural network (LSTM), with the task of predicting the direction of price movements within the company's stock, will have an effect on its accuracy. However, this effect will not always be a positive one, and the LSTM hyperparameters such as the number of layers, number of LSTM neurons in each layer and the LSTM time lag will have a large impact.

It can be said that LSTM networks which take historical price data and sentiment statistics as input are less likely to perform poorly, with the sentiment statistics acting as a form of damage limitation. This was shown when the LSTM time lag was 5, where the baseline networks were no better than taking random guesses, but the Twitter networks maintained an acceptable level of performance.

However, when LSTM networks that take only historical price data perform relatively well, there are no consistent and obvious improvements when sentiment statistics are added to the inputs. This is consistent with the findings of previous works such as [48], which found that incorporating positive/negative sentiment statistics into inputs had no effect on accuracy when predicting the direction of the price changes.

When the LSTM time lag was set to 3, the Twitter networks achieved high accuracies consistently, whereas the accuracy of the baseline networks varied with no obvious patterns being shown. The LSTM time lag of 4 recorded the highest mean accuracy for both the baseline (60.9%) and Twitter (60.3%) networks. However, the mean accuracies of the Twitter networks with this time lag were very varied and inconsistent.

Standard deviations were relatively high across many of the networks tested, which suggests that those networks were underfitting. This is a result of the low quantity of data points available in training. A suggestion for future work could be to gathered multiple years worth of daily stock price and sentiment data, rather than the 11 months used here.

If a Twitter LSTM network was intended to be used to make investment

decisions, then it is suggested that one of the high accuracy networks (such as the single hidden layer architectures) with a time lag of 3 is used. However, if sentiment inputs are not a requirement then it may be a better option to use a baseline network with a time lag of 4, as these were quite consistent at achieving high accuracies, and the effort of producing sentiment statistics is not required.

All the relationships between results that have been discussed show correlation, but not necessarily causality. Given a different company to focus on (instead of Tesla), the results and conclusions made may have been different. As such any future work should aim to run this experiment multiple times for different stocks. One requirement is that these stocks should be volatile, with an almost equal number of up movements in price from day to day as down movements, which minimises skew.

If the experiment was to be repeated again then a much larger quantity of tweets per day should be gathered, by using a more expansive search query. Tweets relating to the company in general rather than just the company's stock should be the aim of the search. If this is not possible then a more suitable corpus should be used to train the sentiment classifier, such as a financial Twitter corpus. This should increase the likelihood that the sentiment statistics are a true reflection of the gathered tweets.

A Raw Results

A.1 LSTM time lag of 3

Architecture	**1**	**2**	**3**	**4**	**5**	**6**
Mean accuracy	58.9	58.8	55.6	52.7	51.2	53.3
Standard deviation	4.19	2.24	2.86	1.72	2.45	4.27
Architecture	**7**	**8**	**9**	**10**	**11**	**12**
Mean accuracy	53.9	56.5	56.1	55.8	56.8	57.7
Standard deviation	2.39	2.68	1.89	3.56	2.97	4.31

Table A.1: Mean accuracy and standard deviation for each baseline architecture with an LSTM time lag of 3

Architecture	**1**	**2**	**3**	**4**	**5**	**6**
Mean accuracy	58.9	58.8	58.2	55.2	56.1	56.2
Standard deviation	3.53	2.12	2.78	2.05	3.19	3.74
Architecture	**7**	**8**	**9**	**10**	**11**	**12**
Mean accuracy	54.7	55.5	57.0	53.8	53.2	52.3
Standard deviation	4.49	3.21	2.28	4.36	3.60	4.42

Table A.2: Mean accuracy and standard deviation for each Twitter architecture with an LSTM time lag of 3

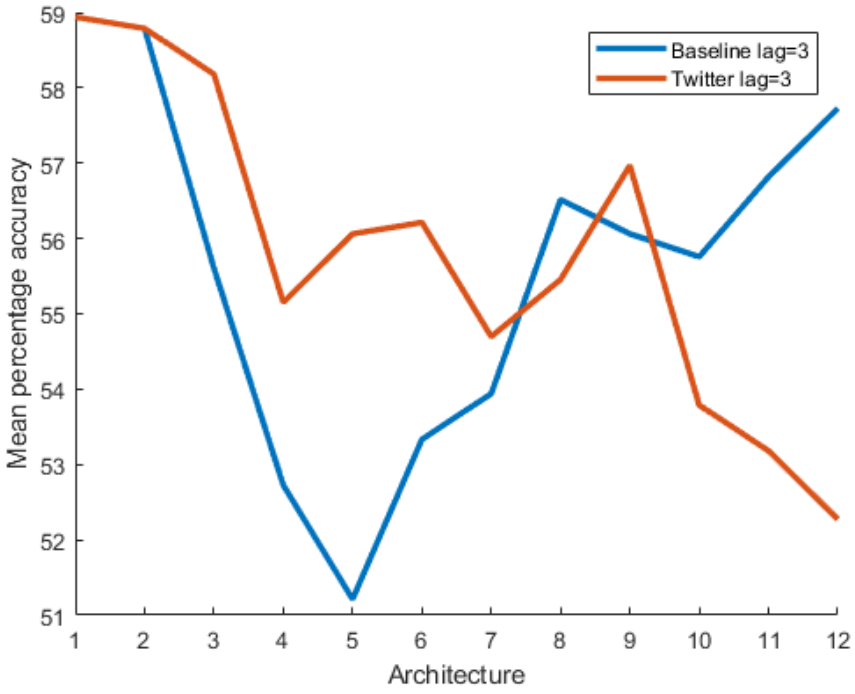

Figure A.1: Mean accuracy for each architecture with an LSTM time lag of 3

A.2 LSTM time lag of 4

Architecture	**1**	**2**	**3**	**4**	**5**	**6**
Mean accuracy	56.7	56.7	57.1	55.9	56.8	60.9
Standard deviation	1.78	3.29	4.52	5.55	4.75	2.84
Architecture	**7**	**8**	**9**	**10**	**11**	**12**
Mean accuracy	57.4	58.8	59.2	57.7	56.5	57.9
Standard deviation	4.65	3.70	2.72	2.98	4.46	2.93

Table A.3: Mean accuracy and standard deviation for each baseline architecture with an LSTM time lag of 4

Architecture	1	2	3	4	5	6
Mean accuracy	53.5	54.5	53.0	58.0	59.7	60.3
Standard deviation	2.48	3.50	2.86	3.12	3.73	3.10
Architecture	**7**	**8**	**9**	**10**	**11**	**12**
Mean accuracy	57.0	57.7	58.5	54.4	55.5	54.5
Standard deviation	3.73	3.46	4.41	3.94	2.96	4.29

Table A.4: Mean accuracy and standard deviation for each Twitter architecture with an LSTM time lag of 4

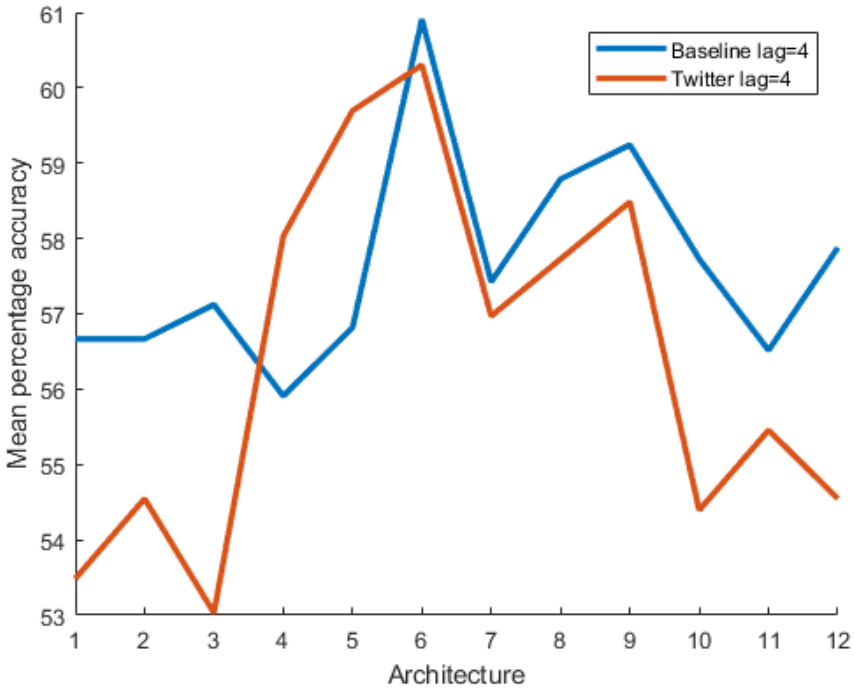

Figure A.2: Mean accuracy for each architecture with an LSTM time lag of 4

A.3 LSTM time lag of 5

Architecture	1	2	3	4	5	6
Mean accuracy	50.9	49.8	50.6	47.3	48.0	49.2
Standard deviation	3.86	3.07	3.29	4.27	3.78	4.24
Architecture	**7**	**8**	**9**	**10**	**11**	**12**
Mean accuracy	46.5	48.8	46.4	48.2	48.9	47.7
Standard deviation	2.58	3.01	4.80	4.99	4.74	3.66

Table A.5: Mean accuracy and standard deviation for each baseline architecture with an LSTM time lag of 5

Architecture	1	2	3	4	5	6
Mean accuracy	50.6	52.6	57.0	49.5	51.2	55.9
Standard deviation	6.75	4.46	4.18	3.92	4.67	5.91
Architecture	**7**	**8**	**9**	**10**	**11**	**12**
Mean accuracy	50.2	52.3	53.3	50.5	51.8	53.3
Standard deviation	4.60	4.80	4.09	4.17	4.15	6.01

Table A.6: Mean accuracy and standard deviation for each Twitter architecture with an LSTM time lag of 5

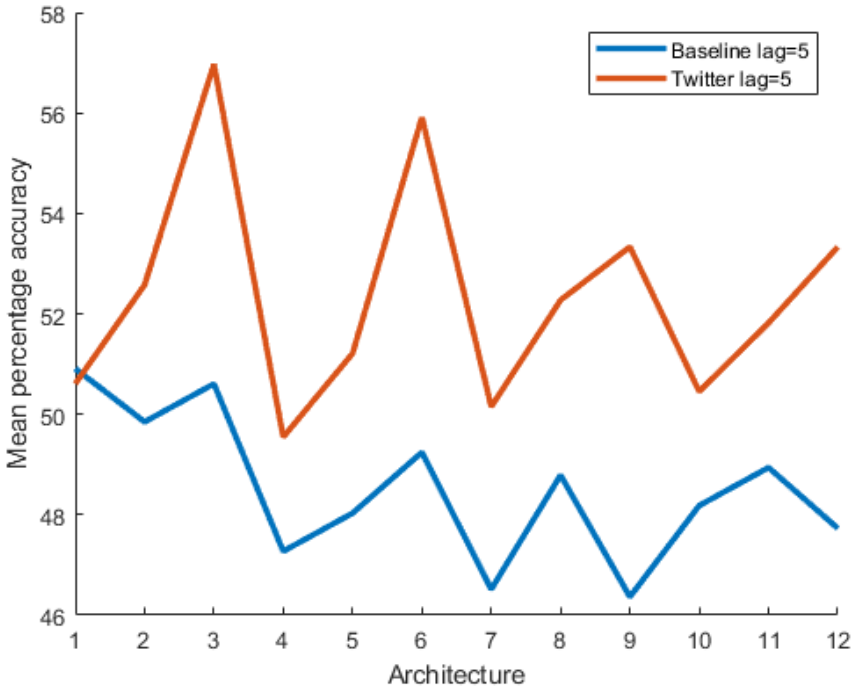

Figure A.3: Mean accuracy for each architecture with an LSTM time lag of 5

Bibliography

[1] (). Triennial central bank survey of foreign exchange and otc derivatives markets in 2016, Bank for International Settlements, [Online]. Available: https://www.bis.org/publ/rpfx16.htm. [Accessed: 22-Jan-2019].

[2] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012, p. 1.

[3] B. Pang, L. Lee *et al.*, 'Opinion mining and sentiment analysis', *Foundations and Trends in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.

[4] A. Timmermann and C. W. Granger, 'Efficient market hypothesis and forecasting', *International Journal of forecasting*, vol. 20, no. 1, pp. 15–27, 2004.

[5] Y. S. Abu-Mostafa and A. F. Atiya, 'Introduction to financial forecasting', *Applied Intelligence*, vol. 6, no. 3, pp. 205–213, 1996.

[6] P. J. Darwen, 'Questioning the efficient markets hypothesis: Big data evidence of non-random stock prices', in *Big Data Analysis (ICBDA), 2018 IEEE 3rd International Conference on*, IEEE, 2018, pp. 201–205.

[7] (). Stock, Business Dictionary, [Online]. Available: http://www.businessdictionary.com/definition/stock.html. [Accessed: 23-Jan-2019].

[8] P. Asquith and D. W. Mullins Jr, 'Equity issues and offering dilution', *Journal of financial economics*, vol. 15, no. 1-2, pp. 61–89, 1986.

[9] G. Gidofalvi and C. Elkan, 'Using news articles to predict stock price movements', *Department of Computer Science and Engineering, University of California, San Diego*, 2001.

[10] J. S. Abarbanell and B. J. Bushee, 'Abnormal returns to a fundamental analysis strategy', *Accounting Review*, pp. 19–45, 1998.

[11] A. W. Lo, H. Mamaysky and J. Wang, 'Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation', *The journal of finance*, vol. 55, no. 4, pp. 1705–1765, 2000.

[12] L. Menkhoff and M. P. Taylor, 'The obstinate passion of foreign exchange professionals: Technical analysis', *Journal of Economic Literature*, vol. 45, no. 4, pp. 936–972, 2007.

[13] K. Gurney, *An introduction to neural networks*. CRC press, 2014.

[14] H. Landahl, W. S. McCulloch and W. Pitts, 'A statistical consequence of the logical calculus of nervous nets', *Bulletin of Mathematical Biology*, vol. 5, no. 4, pp. 135–137, 1943.

[15] F. Rosenblatt, 'The perceptron: A probabilistic model for information storage and organization in the brain.', *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[16] B. Karlik and A. V. Olgac, 'Performance analysis of various activation functions in generalized mlp architectures of neural networks', *International Journal of Artificial Intelligence and Expert Systems*, vol. 1, no. 4, pp. 111–122, 2011.

[17] D. F. Specht, 'Probabilistic neural networks', *Neural networks*, vol. 3, no. 1, pp. 109–118, 1990.

[18] R. Stengel, 'Introduction to neural networks!', 2017.

[19] X. Glorot, A. Bordes and Y. Bengio, 'Deep sparse rectifier neural networks', in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[20] G. K. Jha, 'Artificial neural networks and its applications', *IARI, New Delhi*, 2007.

[21] J. J. Hopfield, 'Neural networks and physical systems with emergent collective computational abilities', *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[22] J.-S. Zhang and X.-C. Xiao, 'Predicting chaotic time series using recurrent neural network', *Chinese Physics Letters*, vol. 17, no. 2, p. 88, 2000.

[23] Y. Bengio, P. Simard and P. Frasconi, 'Learning long-term dependencies with gradient descent is difficult', *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[24] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, 'Empirical evaluation of gated recurrent neural networks on sequence modeling', *arXiv:1412.3555*, 2014.

[25] S. Hochreiter and J. Schmidhuber, 'Long short-term memory', *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] T. G. Dietterich, 'Ensemble methods in machine learning', in *International workshop on multiple classifier systems*, Springer, 2000, pp. 1–15.

[27] B. Muller, J. Reinhardt and M. T. Strickland, *Neural networks: an introduction*. Springer Science & Business Media, 2012.

[28] P. Werbos, 'Beyond regression : New tools for prediction and analysis in the behavioral sciences', Jan. 1974.

[29] D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning representations by back-propagating errors', *nature*, vol. 323, no. 6088, p. 533, 1986.

[30] M. Marvin and P. Seymour, *Perceptrons*. MIT Press, 1969.

[31] L. Bottou, 'Stochastic gradient learning in neural networks', *Proceedings of Neuro-Nimes*, vol. 91, no. 8, p. 12, 1991.

[32] J. Duchi, E. Hazan and Y. Singer, 'Adaptive subgradient methods for online learning and stochastic optimization', *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[33] G. Hinton, N. Srivastava and K. Swersky, 'Neural networks for machine learning lecture 6a overview of mini-batch gradient descent', *Cited on*, vol. 14, 2012.

[34] D. P. Kingma and J. Ba, 'Adam: A method for stochastic optimization', *ICLR*, 2014.

[35] G. G. Chowdhury, 'Natural language processing', *Annual review of information science and technology*, vol. 37, no. 1, pp. 51–89, 2003.

[36] C. D. Manning, C. D. Manning and H. Schutze, *Foundations of statistical natural language processing*. MIT press, 1999.

[37] T. Winograd, 'Procedures as a representation for data in a computer program for understanding natural language', Massachusetts Inst of Tech Cambridge Project Mac, Tech. Rep., 1971.

[38] R. C. Schank and R. P. Abelson, *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press, 2013.

[39] B. Liu, 'Sentiment analysis and opinion mining', *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[40] T. Mikolov, K. Chen, G. Corrado and J. Dean, 'Efficient estimation of word representations in vector space', *arXiv preprint arXiv:1301.3781*, 2013.

[41] B. Pang, L. Lee and S. Vaithyanathan, 'Thumbs up?: Sentiment classification using machine learning techniques', in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, Association for Computational Linguistics, 2002, pp. 79–86.

[42] N. A. Diakopoulos and D. A. Shamma, 'Characterizing debate performance via aggregated twitter sentiment', in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2010, pp. 1195–1198.

[43] (). On social sentiment and sentiment analysis, brnrd.me, [Online]. Available: https://brnrd.me/social-sentiment-sentiment-analysis/. [Accessed: 12-Apr-2019].

[44] A. Pak and P. Paroubek, 'Twitter as a corpus for sentiment analysis and opinion mining.', in *LREc*, vol. 10, 2010, pp. 1320–1326.

[45] (). Twitter reveals its daily active user numbers for the first time, The Washington Post, [Online]. Available: https://www.washingtonpost.com/technology/2019/02/07/twitter-reveals-its-daily-active-user-numbers-first-time. [Accessed: 08-Apr-2019].

[46] (). Number of tweets per day?, David Sayce, [Online]. Available: https://www.dsayce.com/social-media/tweets-day. [Accessed: 08-Apr-2019].

[47] H. Kwak, C. Lee, H. Park and S. Moon, 'What is twitter, a social network or a news media?', in *Proceedings of the 19th international conference on World wide web*, AcM, 2010, pp. 591–600.

[48] J. Bollen, H. Mao and X. Zeng, 'Twitter mood predicts the stock market', *Journal of computational science*, vol. 2, no. 1, pp. 1–8, 2011.

[49] D. McNair, M. Lorr and Droppleman, 'Manual for the profile of mood states (poms)', *San Diego: Educational and Industrial Testing Service*, 1971.

[50] A. Mittal and A. Goel, 'Stock prediction using twitter sentiment analysis', *Standford University, CS229*, vol. 15, 2012.

[51] T. Rao and S. Srivastava, 'Tweetsmart: Hedging in markets through twitter', in *2012 Third International Conference on Emerging Applications of Information Technology*, IEEE, 2012, pp. 193–196.

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

[53] L. Bing, K. C. Chan and C. Ou, 'Public sentiment analysis in twitter data for prediction of a company's stock price movements', in *2014 IEEE 11th International Conference on e-Business Engineering*, IEEE, 2014, pp. 232–239.

[54] D. M. Nelson, A. C. Pereira and R. A. de Oliveira, 'Stock market's price movement prediction with lstm neural networks', in *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2017, pp. 1419–1426.

[55] J. Yao and C. L. Tan, 'A case study on using neural networks to perform technical forecasting of forex', *Neurocomputing*, vol. 34, no. 1-4, pp. 79–98, 2000.

[56] F. Provost, 'Machine learning from imbalanced data sets 101', in *Proceedings of the AAAI 2000 workshop on imbalanced data sets*, AAAI Press, vol. 68, 2000, pp. 1–3.

[57] (). Snap: Network datasets: 476 million twitter tweets, Stanford University, [Online]. Available: https://snap.stanford.edu/data/twitter7.html. [Accessed: 20-Mar-2019].

[58] (). Pricing - twitter developers, Twitter, [Online]. Available: https://developer.twitter.com/en/pricing/search-fullarchive. [Accessed: 20-Mar-2019].

[59] (). Tesla inc. (tsla), Yahoo Finance, [Online]. Available: https://finance.yahoo.com/quote/TSLA/history. [Accessed: 21-Mar-2019].

[60] H. Cheng, P.-N. Tan, J. Gao and J. Scripps, 'Multistep-ahead time series prediction', in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2006, pp. 765–774.

[61] P. J. Brockwell, R. A. Davis and M. V. Calder, *Introduction to time series and forecasting*. Springer, 2002, vol. 2.

[62] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy and P. T. P. Tang, 'On large-batch training for deep learning: Generalization gap and sharp minima', *arXiv:1609.04836*, 2016.

[63] N. Srivastava, A. Krizhevsky, I. Sutskever, R. Salakhutdinov and G. Hinton, 'Dropout: A simple way to prevent neural networks from overfitting', *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.