Machine Learning for Market Prediction

Contents

1 Introduction

1.1 Motivation

Predicting the market has attracted the attention of a wide variety of people. However, the ability to predict moves within the market has proven to be a challenging task. The efficient market hypothesis suggests that stock prices follow a random walk pattern and so will not be predictable with more than 50% accuracy [1]. Not so long ago, it was believed that once there was new information this news would be factored into the price of a market or stock immediately without delay [2].

However, a large amount of research has challenged the efficient market hypothesis. Areas of Behavioural Economics and the Socionomic Theory of Finance suggest that stock market prices do not follow a random walk [3]. The notion of information being incorporated into the price of a stock immediately is not entirely correct either. Thus concluding traditional techniques of technical or fundamental analysis of stocks would help to determine which stocks to invest in [2]. Where technical analysis is the process of using historical prices to find trends and patterns to help predict future prices. Alternatively, fundamental analysis is a method where you look at macroeconomic factors affecting a company or industry.

Current research suggests that information from microblogging platforms such as twitter or discussion forums could take up to 2 or 3 days to be incorporated into the market[4]. However, for mainstream news it has been seen that the news of today has almost zero impact on the stock the next day which supports some of the claims that the efficient market hypothesis suggests [4]. Knowing this, building a machine learning algorithm to help determine the movement of a market on a day by day basis could prove to be quite useful provided the correct trading strategy is implemented.

Current research into Deep Learning based time series forecasting appears to predict future prices quite accurately. However, upon closer inspection it can be seen that actually it is not very effective. Instead, it typically predicts tomorrows price as the same price today when using approaches like LSTM [5]. Trading algorithms incorporating predictions made from event driven strategies have shown to give more generous returns. The use of microblogging platforms such as Twitter have proven to be highly effective at determining the sentiment of a given topic [6]. With millions of users posting over 140 million tweets per day, information about public feelings are readily available. Since each tweet is only 280 characters long (previously this was 140), this means that each person will give a concise, useful opinion about a topic which could be used to make predictions.

However, the issue with social media and microblogging platforms is the text extracted is usually messy and typically has poor grammar [7]. This makes it hard to determine the sentiment of what a person has written as there are uncommon grammar structures and poor spelling. This will need to be addressed and solved during the project.

Combining microblogging platforms along with price and other indicators to build a machine learning algorithm has the potential to give great returns. By analysing the emotions and opinions of people [8] we could demonstrate the potential Deep Learning and Natural Language Processing has in creating a classifier to predict if a stock will go up or down.

1.2 Related Work

Previous work has shown that an accuracy of 56% for predicting up or down in the market is reported as being a good result [7]. Nguyen et al. [7] used the Yahoo Finance messages. They decided to not choose Twitter as it is more difficult to filter relevant tweets and also the difficultly in collecting them. They then labelled messages with various sentiments to allow them to create a classifier to automatically give each message a sentiment. Unfortunately they achieved a low accuracy of 54.4% on average.

Schumaker and Chen [9] have a 57.1% directional accuracy for predicting the market and were also able to create a 2.06% return in a simulated trading environment. In their research, they estimated the direction of a given stock 20 minutes after a news article was released.

Bollen, Mao & Zeng [3] used mood tracking tools OpinionFinder and Google Profile of Mood States (GPOMS) to measure moods in 6 dimensions (Calm, Alert, Sure, Vital, Kind and Happy). They managed to achieve an accuracy of 86.7% when predicting if the market would go up or down on a given day by implementing a self organising fuzzy neural network. However, the testing period was only between December 1 and December 19 2008 which is quite short, only 15 days of trading (since we exclude weekends). This short period was selected as it was a relatively stable and not very volatile. This questions the effectiveness of their model and how good it really is at making predictions in a more real world scenario.

Vu, Chang, Ha & Collier [10] accuracy's ranging from 75% to 82% when predicting technology companies. They combined a variety of features such as extracting bullish/bearish features from tweets, historical price data and also positive and negative sentiment. They tested all these features separately and combined to see which would yield the highest accuracy. They found that by combining all three features they could yield the best accuracy in determining stock market movement. The methods used in this paper are quite interesting and I will be taking inspiration from them. However, they have suffered from a similar issue where the testing period was also very short, only 13 days of trading which questions the reliability of their results. Their method of combining sentiment with price and bullish or bearish features is one to explore further. In my research I will also be combining sentiment with different metrics.

Weng, Ahmed and Megahed [11] have come up with a data driven approach to be able to predict Apple (AAPL) stock movement to 85% accuracy. Not only do they use stock market data, but they also incorporate technical indicators, number of Wikipedia hits and also Google news data. The key limitation of their project is that they have only examined Apple stock so it is unclear if their methodology would work for other stocks. Another limitation is the exclusion of data from microblogging platforms. It would have been interesting to see the effect of also including this. Regardless of these drawbacks, some of the novel ideas included in their project could be used as inspiration in my own project.

1.3 Aims & Objectives

The aim of this project is to analyse how effective various techniques are in predicting whether a stock market will go up or down. There will be a strong emphasis on the sentiment analysis of micro-blogging platforms such as twitter.

The key objectives for this project are:

1. Research previous methods for predicting financial markets using machine learning and determine which methods are most effective. This will be used as a benchmark to compare with my new algorithm.

2. Retrieve or scrape a large data set of tweets and historical stock data for specific companies. This data can be used to train and test a machine learning algorithm.

3. Sentiment analysis algorithm that will either output a sentiment of positive / negative or a numerical value between -1 and 1. This should be tested against other methods of sentiment analysis and the most accurate method should be selected to be used for the project.

4. Implement a machine learning algorithm to determine whether a stock is likely to go up or down with the stocks sentiment as one of the inputs. We will compare our accuracy to other research to see how well we perform compared to other techniques.

2 Design & Implementation

As this is a more research focused project, there is less emphasis of producing a design for a final product. However, design choices have to been made early on in order to plan accordingly for what needs to be built. I've chosen Python as my main language. Python is the most popular language of choice when it comes to Machine Learning [12]. This means there is a large community that can provide support and there are many useful libraries that can help with developing prototypes fast.

I'll also be using and testing a large variety of libraries, such as VADER, GetOldTweets3, numpy and tensorflow, which will be discussed in their relevant sections. Standard libraries to make coding and experimenting easier are those such as numpy, pandas and tensorflow. However, I will be justifying my choices when it comes between using VADER or TextBlob, for example, or perhaps TwitterAPI and GetOldTweets3. This is important as it allows for a design that has been well reasoned and thought out.

2.1 Data

2.1.1 Data selection

In order to determine which tweets we need to keep or query for, we need to decide which keywords to focus on. As suggested in previous research [10], Google Ads Keyword Planner Tool [13] can be used to come up with a list of relevant keywords related to a given company. Table 1 shows some examples of major keywords related to a given company.

Company	Keywords
Apple	Airpods, iPhone XR, Apple Watch, iPhone 7, iPhone, iTunes
Microsoft	Office 365, Onedrive, Window 10, Surface Pro, Xbox
Amazon	Amazon Prime, Jeff Bezos, Alexa, Kindle, Amazon Music
Netflix	Stranger Things, Bird Box, Netflix Movies
Google	Gmail, Google Chrome, Google Forms, Android, Google Pixel

Table 1: Keywords for each company from Google Ads Platform

We can use this list of words to filter out tweets by ensuring that at least one of the keywords in mentioned in the tweet. This can be done by creating a more complex query to twitter which will be covered later.

2.1.2 Tweets

Having a large collection of historical tweets is important as we will need to use it as input for sentiment analysis. The output sentiment will be used as a feature for a machine learning algorithm later.

One option is to use the TwitterAPI [14]. However, one major limitation is that on the free plan, we can only scrape tweets from up to 7 days ago. This would be alright when using our final model live. However, we need larger number of historical tweets to be able to train and test our model effectively, which would allow us to produce more reliable results than some previous research [3, 10].

The alternative is to use an open source library called GetOldTweets3 [15]. With this library I have created a small script in python which allows me to scrape tweets. GetOldTweets3 works by accessing tweets from the front-end of Twitter. While this is likely to be slower than using the twitter API it provides an effective method to get historical tweet data.

I have also included a variety of parameters such a date range, language of tweets, query to find tweets and the number of tweets to get per day. The tweet ID, tweet text, hashtags, number of retweets and favorites, link to tweet and the date are outputted to a csv file in order of date. Keeping data on the number of retweets and favorites could be used as a weighting factor to see which tweets have a bigger impact on people and as a consequence the stock market. Figure 1 shows an example query that could be sent to twitter.

```
query = "(@apple OR mac OR apple OR iphone)"
date_start = datetime.date(2019, 11, 20)
date_until = datetime.date(2019, 11, 12)
num_of_tweets = 50000
tweetCriteria = got.manager.TweetCriteria().setQuerySearch(query)\
                                           .setSince(str(date_start))\
                                           .setUntil(str(date_until))\
                                           .setMaxTweets(num_of_tweets)\
                                           .setEmoji("unicode")\
                                           .setLang("en")
```

Figure 1: Code extract showing how to get tweets based on a number of parameters

The code in Figure 1 allows you to search for as many tweets as you want between the date_start and date_until variable. I have modified the above example to loop through each day within a given time frame and select a number of tweets to download per day. This allows me to have better control over how many tweets I want per day instead of downloading every single tweet which may be more time consuming and result in diminishing returns where there is little to no accuracy improvement from scraping more tweets.

An important part of the code above is the query variable. As mentioned previously we want tweets that contain relevant keywords of a particular company. To come up with a list of keywords, we can use the Google Ads Keyword tool to help us with that as mentioned previously. The query section is where we would input any relevant keywords. We can use the OR operator to ensure that at least one of our keywords is mentioned. Twitter allows you to make this more complex and we can test this from the front end by using the advanced search feature.

2.1.3 Daily Up or Down

We will need market data for each given company. With this data we can then calculate if the stock went up or down on a given day. Yahoo Finance [16] gives us the ability to download all historical stock data for a given company. Table 2 shows an example of the data and the format we get it in from Yahoo Finance:

Date	Open	High	Low	Close	Adj. Close	Volume
27/11/2018	171.509995	174.770004	170.880005	174.240005	171.660797	41387400
28/11/2018	176.729996	181.289993	174.929993	180.940002	178.261627	46062500
29/11/2018	182.660004	182.800003	177.699997	179.550003	176.892197	41770000
30/11/2018	180.289993	180.330002	177.029999	178.580002	175.936554	39531500
03/12/2018	184.460007	184.940002	181.210007	184.820007	182.084183	40802500

Table 2: Stock data for a given company provided by Yahoo Finance

Figure 2 shows some pseudo code on how each day will be labelled. If the difference between the close price and open price is positive then the stock went up on that given day. If the difference is negative then the price went down.

```
difference = close_price - open_price

if(difference > 0):
    label = UP

elif(difference < 0):
    label = DOWN
```

Figure 2: Code extract showing how to label each days stock movement

There is a rare possibility that the price could stay exactly the same. However, this is very unlikely but could be easily handled by simply ignoring it. Alternatively, we could extend our labelling by introducing a 3rd class when the price does not change or does not change significantly we could label this as being the same. The range at which we decide if something has not changed much may need to be attuned to each individual stock based on their volatility and what is considered a large change. However, this is something that could be experimented with at a later date and is not a key objective of this project.

2.2 Sentiment Analysis

An important feature that will be used for the machine learning part of this project will be the sentiment of tweets about a given topic.

TextBlob [17] is a library for natural language processing. It has a sentiment analysis tool that allows you to classify sentences. TextBlob has two inbuilt methods of sentiment analysis. The first is using the pattern library and the second is using a Naive Bayes classifier.

Surprisingly, from Table 3 we can see that the Naive Bayes classifier has a lower accuracy compared to the pattern implementation. However, this could easily be explained by the fact that the TextBlob Naive Bayes algorithm is trained on a set of movie reviews. Since we are analysing tweets the type of language used is different and results in the classifier not being as accurate.

The VADER [18] library is an alternative to TextBlob. Both VADER and TextBlob are based on the nltk python library. However, from Table 3 we can see that VADER has the highest accuracy. This is because VADER has been specifically attuned to microblog-like contexts such as twitter.

Sentiment Tool	Accuracy	Precision	Recall	F1
TextBlob Pattern	62.26%	0.56	0.64	0.60
TextBlob NaiveBayes	54.65%	0.57	0.54	0.56
VADER	64.99%	0.62	0.66	0.64
VADER (Ignore Neutral)	77.48%	0.93	0.75	0.83

Table 3: Comparison of Sentiment Analysis Libraries

In order to improve the classification accuracy of VADER we can ignore when it classifies tweets as neutral. From this we can see the Accuracy increase to 77.48% and also the F1 Classification Accuracy has increased to 0.83. Neutral tweets can be defined as tweets which receive a score between -0.5 and 0.5. Tweets with a score above 0.5 are positive and below -0.5 are considered negative.

The data set used to test each of these sentiment libraries has a collection of 1.6 million tweets containing 800,000 positive tweets and 800,000 negative tweets [19]. It is worth noting that this data set was not labelled by humans. Therefore, the accuracy's shown in Table 3 are likely to be lower than what the true accuracy would be in a real world situation. This does not mean that the testing of them is completely useless. We can still compare each of the algorithms to one another to help decide which is the best to continue forward with. In this case VADER. Experiment shown in the paper discussing VADER suggests that the F1 accuracy on tweets is 0.96 which is very high [18].

Before settling with VADER it is also worth mentioning online APIs that can be used for sentiment analysis. There exists sentiment analysis tools by IBM, Google and AWS. The one major drawback of these tools is that they are limited by the number of API calls you can send. Afterwards, you need to pay to continue using their API. In addition to this since their algorithm is a black box it does not allow us to easily extend or improve it ourselves. However, it is still worth comparing to see if it generates better results compared to VADER. Experiments online [20] compare these 3 API sentiment tools against each other. I have used the same data set [21] as the experiment uses and have compared it to how well the VADER sentiment classification performs.

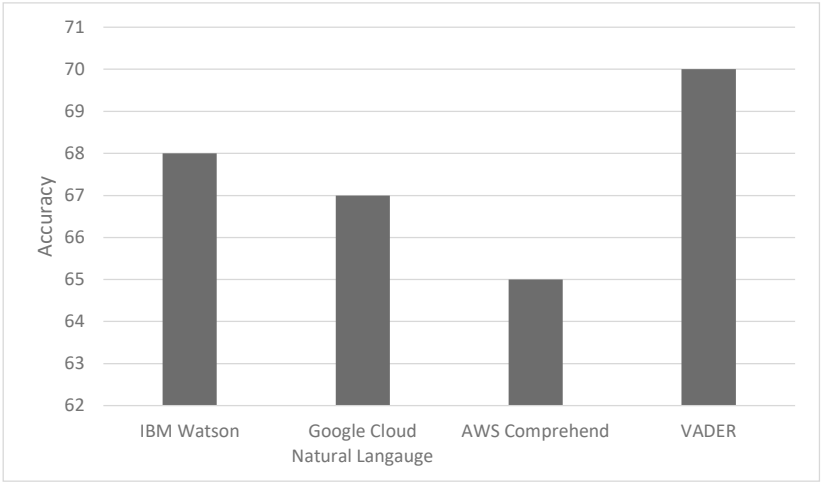

Figure 3: Accuracy of Sentiment Decisions

After classifying the data into three categories, positive, neutral and negative, VADER gets an accuracy of 70%. According to this test this has beaten the capability of IBM Watson, Google Cloud and also AWS. After a comparing VADER to all of the other options available, it is clear that we will be using this going forward. In addition to this, VADER is also open source and is quite easily extendable.

2.3 Preprocessing

Before data is sent to VADER for sentiment analysis, we need to preprocess some of the tweets before handing them over. VADER is lexicon based. This means that it will check the tweet and see if any words in the tweet are in the lexicon. As it is lexicon based, we do not need to worry about removing things like usernames or links within the tweets since VADER will essentially ignore them anyways.

However, there are certain issues that do need to be addressed. The first is the ability to handle emojis. VADER has a lexicon for emojis and will score them a sentiment. However, I found that certain issues occurred where emojis were not being recognised.

Input	Sentiment (-1 to 1)
“😀”	0.7184
“😀😀😀”	0.0
“😀 😀 😀”	0.9517

Table 4: Sentiment scores of various emoji inputs

From Table 4 you can see that if there is any other text or emoji next to another emoji then VADER does not give it a sentiment score. To fix this we will need to preprocess tweets by adding a space either side of every emoji. As you can also see from Table 4 when we have 3 emojis in a row the sentiment score reflects that. In this case 3 smiley faces has a higher sentiment than one smiley face.

```
from emoji import UNICODE_EMOJI

def is_emoji(s):
    return s in UNICODE_EMOJI

def add_space(text):
    return ''.join(' ' + char if is_emoji(char) else char for char in text).strip()

text = add_space("Loving my new iPhone! 😀😀")
print(text)
```

Figure 4: Code showing how to add spaces between emojis

The code extract in Figure 4 shows how to separate emojis appropriately from each other. This allows VADER to give a more correct sentiment score to the tweet.

Another issue with some tweets is the exaggeration of certain words where some letters may be repeated for this effect. This raises an issue where those words are not part of VADERs lexicon. This is illustrated in Table 5.

Input	Sentiment (-1 to 1)
“The new iPhone is so cooooooool”	0.0
“The new iPhone is so cool”	0.4572

Table 5: Exaggerated words by repeated characters

The word cool when exaggeated is not part of the lexicon of VADER. There are various techniques and methods we could use to solve this. Previous research has addressed this problem by first reducing

the number of multiple characters down to only 3 repeats [10]. The next step is to apply a method to normalise tweets by constructing a lexicon of variants of known words [22]. This could be one possible solution to the problem. However, an already existing spell checking tool, such as textblob, might also be a viable option. This will need to be tested by experiement to determine which is the best moving forward.

Another issue with VADER is that it does not automatically recognise words with a "#" in front of them. This could lead to a more inaccurate sentiment score due to certain words being missed out. We can see from Table 6 when the hashtag symbol is not removed then we do not get a sentiment score. However, once we process the tweet to remove the hashtag symbol it increases to a positive sentiment as expected.

Input	Sentiment (-1 to 1)
"The new iPhone is #Amazing"	0.0
"The new iPhone is Amazing"	0.5859

Table 6: VADER not recognising and treating hashtags appropriately

We can easily remove hashtags by some simplistic code involving a regular expression. We find all hashtags in the tweet and remove them. We do not need to worry about the following case: "#Amazing#Fantastic" where 2 hashtags are next to each other. This is because twitter requires the user to separate the hashtag with a space. Figure 5 shows how we can process each tweet to remove the hashtag symbol which would result in the desired output.

```
import re

tweet = "The new iPhone is #Amazing"
processed_tweet = re.sub("#", "", tweet)
print(processed_tweet)
```

Figure 5: Code extract showing simple regular expression to replace hashtags with a space

2.4 Machine Learning

We will be using a supervised machine learning model for this part of the project. The output will either be a prediction to buy or sell. We will label our data as suggested in the previous section with either buy or sell. Sentiment will be one of the inputs for the model.

There are a variety of different machine learning methods we could use. As part of this project, I would like to compare a variety of different machine learning techniques and pick a model that works the best.

An important part that will need to be tested and experimented with is the various inputs we could use. For the sentiment, we could simply average the sentiment for the day and have that as an individual input. Additionally, it could be worth testing 2 days of sentiment as it has been suggested this is the maximum amount of time before microblogging data is no longer a relevant factor in the market [4].

A variety of additional features such as price, highest daily price, lowest daily price, volume and other technical indicators could be used to train a model. However, we would only want to select those features that improve the classification accuracy of the model. Having partially irrelevant or irrelevant features will slow down the time it takes to train and likely reduce the accuracy of our model.

3 Progress

3.1 Project Management

I've approached this project in an agile manner. Although this is a more research focused project, agile can still be applied here. Working in 2 week sprints, each sprint I decide what tasks I want to complete with the consultation of my supervisor. This method has been effective and has allowed me to stay on track to what I want to achieve by the end of this term.

Figure 6 shows my initial plan that I thought I would follow. So far I have found and used various data sets of tweets labelled with sentiment. These data sets were used to benchmark various algorithms for sentiment classification against each other to help make a more informed decision about which method of sentiment analysis would be most appropriate for this project. In addition to this I have also created code that will allow me to scrape historical tweets with very specific parameters. These tweets will have their sentiment classified and then the sentiments will be averaged each day and then given as an input for machine learning. In addition to this, I have also explored some methods for preprocessing my data. However, this will be a key focus for next term.

One of the key risks with a research project is that there is always room to find improvements. Often when engrossed in testing a new idea or finding a new type of algorithm one can often end up spending too much time out of curiosity to test it. I often found that while implementing a sentiment analysis algorithm I would find new data sets a compare all of them again. This runs the risk of not completing the key objectives in time. In order to mitigate this risk it is important to keep a visual of the Gantt chart available at all times. Maintaining a to-do list also ensures that I meet my key objectives on time.

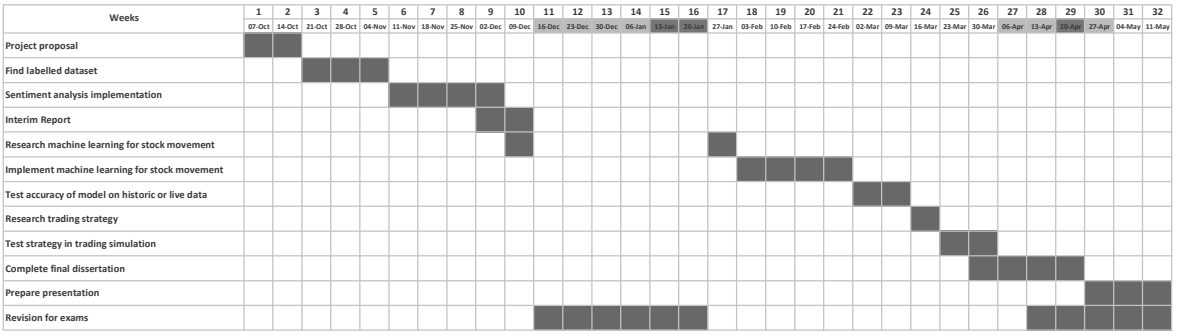

Figure 6: Initial Gantt chart

As the initial Gantt chart was produced during the premature stages of the project, a variety of changes have been applied. As the project has been developed I have put more emphasis on the sentiment analysis part as an input for the machine learning model. I have also put more emphasis onto the overall process of collecting data. This involves the collection of tweet data itself.

As a result of this, I decided early on that the creating of a trading simulation will no be within the scope of the project. This means I can focus of improving the accuracy of predicting either up or down for a given stock. In the revised plan I have dedicated 6 weeks to scrape and process the data to later be implemented by a machine learning algorithm. I have kept the amount of time to implement the machine learning algorithm the same (4 weeks) as I will likely be experimenting with different types of inputs and may inevitably attempt to try different types of data. Whilst 4 weeks may seem quite high, this allows me to mitigate the inevitable risk of wanting to experiment with different inputs, models and data.

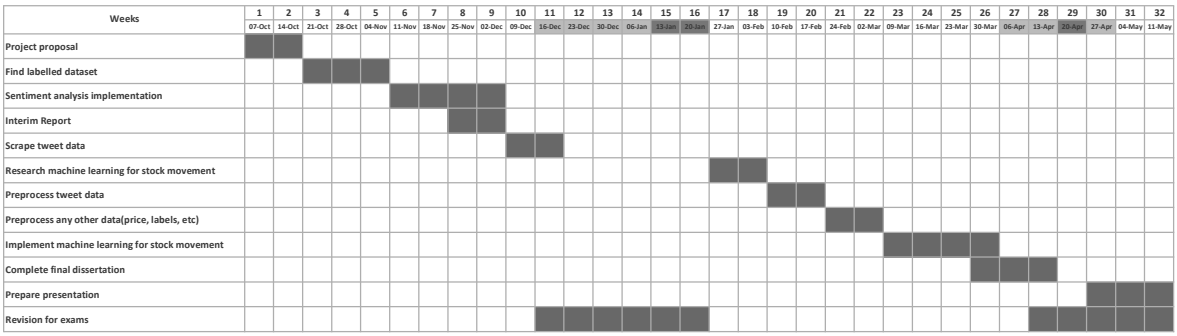

Figure 7: Revised Gantt chart

3.2 Contributions & Reflections

So far this term I have been able to carry out research to a high quality. This has meant that I have been able to create a true focus of the project and understand exactly what I want to achieve by the end of it. As I will be taking fewer credits next semester I will be able to focus even more time towards this project.

During this semester I have managed to focus of researching and experimenting various techniques and produce experimental code to test my theories. So far my contributions have included creating a method for scraping large amounts of historical tweets. I've also analysed the effectiveness of various sentiment analysis methods. I have also proposed the idea of combining twitter sentiment with other technical indicators for a machine learning algorithm, inspired by previous research with the aim to improve on these.

Next semester I aim to utilise my experimental code (such as tweet scraping, sentiment analysis and price labelling) and collect real data using the methods described and to preprocess using the techniques I have implemented this term to then create a machine learning model to predict daily stock movements.

I am very happy with the initial stages of my project as it has allowed me to understand which direction to take the project. I am excited to continue implementing my findings and see the outcome of my project.

4 Bibliography

References

[1] B. Qian and K. Rasheed, "Stock market prediction with multiple classifiers," *Applied Intelligence*, vol. 26, no. 1, pp. 25–33, 2007.

[2] B. G. Malkiel, "The efficient market hypothesis and its critics," *Journal of economic perspectives*, vol. 17, no. 1, pp. 59–82, 2003.

[3] 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.

[4] W. Zhang and S. Skiena, "Trading strategies to exploit blog and news sentiment," in *Fourth international aAAI conference on weblogs and social media*, 2010.

[5] M. Roondiwala, H. Patel, and S. Varma, "Predicting stock prices using lstm," *International Journal of Science and Research (IJSR)*, vol. 6, no. 4, pp. 1754–1756, 2017.

[6] 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)*, pp. 1345–1350, IEEE, 2016.

[7] T. H. Nguyen, K. Shirai, and J. Velcin, "Sentiment analysis on social media for stock movement prediction," *Expert Systems with Applications*, vol. 42, no. 24, pp. 9603–9611, 2015.

[8] 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.

[9] R. P. Schumaker and H. Chen, "Textual analysis of stock market prediction using breaking financial news: The azfin text system," *ACM Transactions on Information Systems (TOIS)*, vol. 27, no. 2, p. 12, 2009.

[10] T. T. Vu, S. Chang, Q. T. Ha, and N. Collier, "An experiment in integrating sentiment features for tech stock prediction in twitter," in *Proceedings of the workshop on information extraction and entity analytics on social media data*, pp. 23–38, 2012.

[11] B. Weng, M. A. Ahmed, and F. M. Megahed, "Stock market one-day ahead movement prediction using disparate data sources," *Expert Systems with Applications*, vol. 79, pp. 153–163, 2017.

[12] T. Elliot, "The state of the octoverse: machine learning - the github blog." `https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning/`, 2019. Accessed: 2019-11-24 [online].

[13] Anon, "Keyword planner: Choose the right keywords." `https://ads.google.com/intl/en_uk/home/tools/keyword-planner/`, 2019. Accessed: 2019-11-27 [online].

[14] Anon, "Twitter api." `https://developer.twitter.com/`, 2019. Accessed: 2019-11-24 [online].

[15] H. Jefferson and Mottl, "Getoldtweets3." `https://github.com/Mottl/GetOldTweets3/`, 2019. Accessed: 2019-11-24 [online].

[16] Anon, "Yahoo finance – stock market live, quotes, business & finance news." `https://uk.finance.yahoo.com/`, 2019. Accessed: 2019-11-27 [online].

[17] Anon, "Textblob: Simplified text processing." `https://textblob.readthedocs.io/`, 2019. Accessed: 2019-11-26 [online].

[18] C. J. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Eighth international AAAI conference on weblogs and social media*, 2014.

[19] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N Project Report, Stanford*, vol. 1, no. 12, p. 2009, 2009.

[20] Anon, "Sentiment analysis in the cloud with google cloud natural language, aws comprehend, & ibm watson." `https://www.deducive.com/blog/2018/6/02/using-r-for-sentiment-analysis-with-aws-comprehend-google-cloud-natural-language-ibm-watson/`, 2018. Accessed: 2019-11-29 [online].

[21] R. Maestre, "Sentiwordnet-bc." `https://github.com/rmaestre/Sentiwordnet-BC/blob/master/test/testdata.manual.2009.06.14.csv`, 2013. Accessed: 2019-11-29 [online].

[22] B. Han, P. Cook, and T. Baldwin, "Automatically constructing a normalisation dictionary for microblogs," in *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pp. 421–432, Association for Computational Linguistics, 2012.