Modelling Stock Volume Using Twitter

Andrew Cropper

Submitted to Oxford University for the degree of

MSc Computer Science

September 2011

Abstract

Stock market prediction is nothing new. For years researchers from various disciplines have been seduced by the appeal of financial gain. Some believe that stock price movements are governed by the random walk hypothesis, which states that stock prices evolve according to a random walk and thus cannot be predicted any more accurately than predicting whether a coin will land heads or tails; others disagree. This thesis investigates whether Twitter can be used to predict stock volume. A Twitter dataset of 40 million tweets is constructed for the period 01 Jan - 31 May 2011. Support vector regression is then used to model and predict next day stock volume for four stocks. The results are disappointing. Tests using features extracted from Twitter fail to beat a two day moving average baseline. This, however, does not mean that prediction using Twitter is impossible, and there is strong potential for future work.

Acknowledgements

I would like to thank Brian Harrington for supervising this project.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Stock market prediction is nothing new. For years researchers from various disciplines have been seduced by the appeal of financial gain. Some believe that stock price movements are governed by the Random Walk Hypothesis (Fama, 1965b), which states that stock prices evolve according to a random walk and thus cannot be predicted any more accurately than predicting whether a coin will land heads or tails; others disagree. This thesis investigates whether Twitter can be used to predict next day stock volume.

1.1 Project Background

Generally, stock price and movement is the focus of both investors and researchers, with volume often overlooked (Sun, 2003). Volume is the number of shares traded over a given period of time, usually a day. Essentially, the higher the volume, the more active the stock.

Volume is used in technical analysis to confirm trends. In general, volume moves with the trend. If prices move in an uptrend, volume usually increases. Likewise, in a downtrend volume usually decreases. Thus, price movements with relatively high volume are viewed as stronger than similar movements with weak volume. In addition, if volume is starting to decrease in an uptrend, it is usually a sign that the uptrend is about to end. Despite volume being subject to distortions, its effectiveness as an indictor of future movements is widely accepted by market analysts (Klinger, 1997).

1.2 Aims and Objectives

This thesis focuses on investigating whether Twitter activity corresponds to volume. There are three hypotheses that this thesis wishes to prove:

Hypothesis 1: Twitter activity for company c correspond to stock volume for company c.

Hypothesis 2: Sudden changes in Twitter activity for company c correspond to sudden changes in stock volume for company c.

Hypothesis 3: By modelling Twitter activity and stock volume for company c, next day predictions on stock volume can be made.

These hypotheses are based on the theory that news drives both Twitter and the stock market. The stock market is driven by new information (Kumar et al., 2010). This information can be a company releasing financial data, a government releasing unemployment statistics, etc. Likewise, Twitter is driven by new information (Asur et al., 2011).

On the 24th of August 2011 Steve Jobs resigned as CEO of Apple. Immediately, #SteveJobs became the top trending topic on Twitter. In addition, four out of the top five trending topics were related to Apple (Mogg, 2011), (Martinelli, 2011). The following day Apple's stock volume increased 50% on the previous day's amount. The same information caused these two events. This thesis investigates this relationship, specifically whether this can be modelled for the purposes of prediction.

An argument to the ideas presented in this thesis is that changes in the stock market cause the Twitter activity, i.e., Twitter reacts to the stock market. This is true. People do tweet in response to stock market movements:

"lmao @ people who sold their Apple stock when Jobs stepped down as CEO. they're back up at $390/share now. u mad?"

However, there are also instances where Twitter activity precedes stock market changes. Chapter four details such an instance. This can be generalised as: "Twitter reacts to information quicker than the markets". It is this premise that drives this thesis. If true, it may be possible to model the relationship as to make short-term predictions on volume.

In this thesis, short-term means a day in advance. Wolfram (2010) found that Twitter could aid stock price predictions up to 15 minutes; above this the accuracy decreased. However, as stock volume is seen as an indicator for trend, short-term predictions, in terms of minutes, are unnecessary. Thus, the focus is on next day predictions.

It is understood that no previous work has attempted to model and predict stock volume using Twitter. Therefore, the findings of this thesis can be considered experimental. The work is intended as an investigation into the concepts, rather than a practical implementation.

The direction of the thesis is as follows: construct Twitter dataset and collect financial data; construct features from Twitter dataset; using support vector regression, model the relationship between Twitter activity and volume; predict next day stock volume.

1.3 Chapter Outlines

This thesis is composed of six chapters. The structure is as follows.

Chapter Two: This chapter introduces the necessary background and theory required for the thesis. Financial markets are introduced and stock volume is explained in detail.

Fundamental financial theory, such as the Efficient Market Hypothesis and the Random Walk Theory, is also covered. Included in this chapter is a literature review. This covers similar work to this thesis. The chapter concludes with a section on social networking and its potential for prediction.

Chapter Three: This chapter documents the methodology for collecting and processing the data required in the thesis. The majority of this chapter is devoted to constructing a Twitter dataset.

Chapter Four: This chapter documents the methodology behind the experiments. This includes evaluation methods, choice of baseline, and how the data collected in chapter three is modelled using support vector regression.

Chapter Five: The results of the experiments conducted are presented in this chapter. The results are analysed and discussed.

Chapter Six: The final chapter gives on overview on the thesis. The results are summarised, and future work is discussed.

Chapter 2

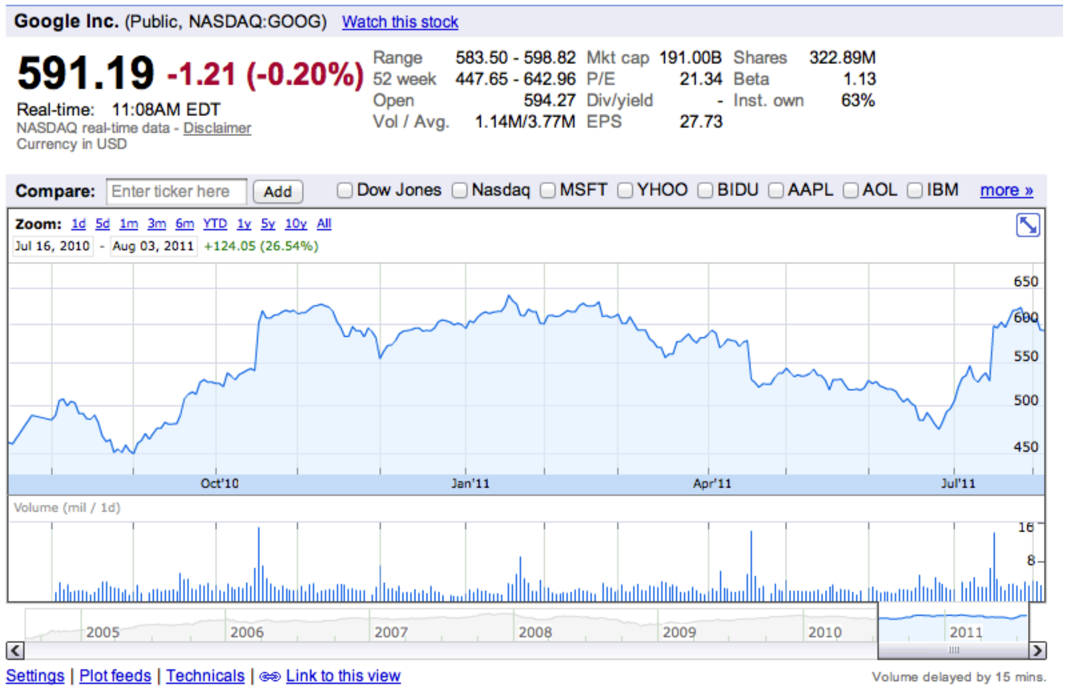
Background and Theory

This chapter details the necessary background and theory for the thesis. First, the chapter introduces financial markets. Stock volume is covered in detail, and evidence is provided supporting the rational behind predicting it. Also covered is an overview of fundamental financial theory. This includes the Efficient Market Hypothesis and the Random Walk Theory. Included in this chapter is a literature review. The purpose of this is to show that previous work has been considered, and that this work is no repeating work already produced. The chapter concludes with a section on social networking and its potential for prediction.

2.1 Financial Markets

A stock market is a public market for the trading of company stock and derivatives at an agreed price. This thesis refers to secondary markets, not primary markets. The primary market is where securities are created (by means of an IPO). The secondary market is where investors trade previously issued securities without the involvement of the issuing companies (Brigham, 1980).

Shares are exchanged via an auction. Sellers state a selling price and buyers state a buying


Google Inc. (Public, NASDAQ:GOOG) Watch this stock

591.19 -1.21 (-0.20%)

Real-time: 11:08AM EDT
NASDAQ real-time data - Disclaimer
Currency in USD

Range	583.50 - 598.82	Mkt cap	191.00B	Shares	322.89M
52 week	447.65 - 642.96	P/E	21.34	Beta	1.13
Open	594.27	Div/yield	-	Inst. own	63%
Vol / Avg.	1.14M/3.77M	EPS	27.73		

Compare: Enter ticker here Add ☐ Dow Jones ☐ Nasdaq ☐ MSFT ☐ YHOO ☐ BIDU ☐ AAPL ☐ AOL ☐ IBM more »

Zoom: 1d 5d 1m 3m 6m YTD 1y 5y 10y All
Jul 16, 2010 - Aug 03, 2011 +124.05 (26.54%)

Settings | Plot feeds | Technicals | Link to this view

Volume delayed by 15 mins.

Figure 2.1: Google Finance

price. When the prices match, the transaction takes place (Wyss, 2000). The participants of these transactions range from individuals, often trading online at home, to large multi-billion dollar investment banks.

Investors (people who buy and sell shares) use various metrics as stock market indicators that help them evaluate a company. Figure 2.1 displays various metrics for Google Inc. (NASDAQ:GOOG), as displayed on Google's own finance site[1]. These metrics include the price, yearly highs and lows, number of shares, etc.

2.1.1 Stock Volume

According to Sun (2003), when investors evaluate a stock, volume often goes unnoticed. The price of the stock is usually their primary concern. Indeed, they are likely to consider yield,

[1]http://www.google.com/finance?q=goog

price-to-earnings ratio, and others before even considering the volume statistic. Despite this, volume has a relationship to price and other aspects of a stock (Sun, 2003).

Volume is a measure of the quantity of shares that change owners for a given stock in a given period. This figure is often given for a trading day. This value varies at anytime on any given day depending on several factors. The major cause of fluctuation is new information about a company (Kumar et al., 2010). This information could be a press release or an earnings announcement. Sun (2003) gives an example of the affect information has on volume. On December 17, 2002, McDonald's Corporation, which had an average trading volume of 7.58 million shares per day, announced a warning and reduction of expected earnings. This new information caused a sudden increase in trading volume, with 35.17 million shares traded that day, about five times the average. It also caused a drop in price of 8%. This change in volume, according to Sun, was due to differences in investor's views of the valuation after incorporating the new information.

Information is anything that causes investors to act, regardless of whether it has any fundamental affect on the underlying valuation of the company. Sometimes information on a company can affect the volume and price of an unrelated company. Such an incident happened between 1996 and 1997 when news releases with information on MCI Communications (Nasdaq:MCIC) led to an increase in volume in Massmutual Corporate Investors (NYSE:MCI) due to ticker confusion. Rashes (2001) found that Massmutual's top volume days between 11/1/1996 and 11/13/1997 all occurred on days when there was merger news on MCI Communications, showing that Massmutual's volume was correlated with MCI Communications' volume, but not those of other telecommunications companies.

Because of the inferences that can be made from volume, the analysis of volume and associated price changes resulting has been of interest to researchers. Karpoff (1987) suggests that volume gives an insight into the structure of financial markets. He argues, "Correlations between volume and price can provide information regarding rate of information flow in the marketplace, the extent that prices reflect public information, the market size, and the existence of short sales and other market constraints". This view is similar to the old Wall Street proverb of "it takes volume to make prices move". Karpoff (1987) calls this proverb 'positive volume-absolute price change correlation'. Numerous empirical studies support this correlation and outlined below.

One of the first analytical studies into the relationship between volume and price changes was conducted by Ying (1966). Ying suggested that volume could predict price changes. Ying analysed a six-year, daily series of price and volume using Standard and Poor's 500 composite index. There were five significant findings regarding the correlation behind price and volume, thus supporting the rational behind predicting volume:

1. A small volume is usually accompanied by a fall in price.
2. A large volume is usually accompanied by a rise in price. A large increase in volume is usually accompanied by either a large rise in price or a large fall in price.
3. A large volume is usually followed by a rise in price.
4. If the volume has been decreasing consecutively for five trading days, then there will be a tendency for the price to fall over the next four trading days.
5. If the volume has been increasing consecutively for a period of five trading days, then there will be a tendency for the price to rise over the next four trading days.

Other works of note include Crouch (1970), who found a correlation for market indices and individual stocks using daily price and volume data; Morgan (1976), who found that the variance of price change was positively related to volume; Jain and Joh (1988) found a similar correlation over one-hour intervals, using data from a market index. In total, Karpoff (1987) lists 17 research papers, with varying time intervals and securities, all supporting the correlation of price change with volume. The conclusion was that the absolute price and volume correlation existed, although the correlation was often weak (Karpoff, 1987).

Overall, volume is a useful metric for investors because it is used to confirm trends. In general, volume moves with the trend. If prices are moving in an upward trend, volume usually increases. Likewise, in a downward trend volume usually decreases. Thus, price movements with relatively high volume viewed as stronger than similar movements with weak volume. In addition, if volume is starting to decrease in an uptrend, it is usually a sign that the upward run is about to end. Despite volume being subject to distortions, its effectiveness as an indictor of future movements is widely accepted by market analysts (Klinger, 1997).

2.2 Financial Theory

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on a financial exchange. Some believe that the stock market is unpredictable. Indeed, stock market prediction research goes against widely accepted theories that state that predicting the price or change of a stock is an impossible task. Others disagree; supporting their arguments with examples of successful investors, such as Warren Buffett, who has made billions through the stock market.

2.2.1 The Random Walk Hypothesis

The Random Walk Hypothesis (RWH) (Fama, 1965b) is a fundamental theory in finance. It states that stock prices evolve according to a random walk and thus cannot be predicted any more accurately than predicting whether a coin with land heads or tails. In this theory, future prices do not depend on past prices; the price movement of a stock is no more predictable than the random selection of successive steps in the positive, negative, or equal direction. If this theory is true, patterns cannot be exploited since trends do not exist.

Lo and MacKinlay (1999) disagree with the RWH and suggest that financial markets are predictable, to some degree. They argue that predictability is not symptom of inefficiency or irrationality but that it is the "the oil that lubricates the gears of capitalism". They also theorise that in the short-run stock prices can gain momentum due to investors 'jumping on the bandwagon'. They see this as several consecutive periods of same direction price movement with a particular stock. Shiller (2001) accounts the Dot-com bubble to the same lemmings affect.

2.2.2 Efficient Market Hypothesis

The Efficient Market Hypothesis (EMH) (Malkiel, 2003a) states that financial markets are 'informationally efficient' and that they immediately reflect the assimilation of all the information available. The theory states that the current stock price reflects all the available information and that future movement is only a consequence of future events. When new information is made available it immediately corrects the value of a stock to its 'correct' value. Therefore, because news is generally unpredictable, prices will follow a random walk pattern and cannot be predicted with more than 50% accuracy. EFH also claims that all available information about the current financial standing of the company is accessible to everyone.

The consequence of this is that the notion of profiting from predicting price movements is difficult and unlikely. The cause of price changes, according to EMH, is the introduction of new information. If a market is said to be 'efficient', i.e., if prices adjust quickly and, on average, without bias to new information, the current prices of securities reflect all available information at any given time. Thus, prices are never too high or too low because they adjust before an investor has time to trade on and profit from a new a piece of information.

This theory is based on the suggestion that market prices incorporate all available information at any time. However, according to the EMH, there are different kinds of information that influence stock values. EMH splits into three forms: weak, semi-strong, and strong.

In weak EMH, only historical information is embedded in the current price. Under this form of EMH, technical analysis, which is the study of historical stock prices to predict future ones, will not be able to generate excess returns in the long-run. However, fundamental analysts, who analyse fundamental financial information, such as company profit, asset values, etc., may still be able to approximate future values. This discrepancy arises because stock prices are widely and easily available to the public. Because of this, the theory claims that one should not be able to profit from information that everyone else knows. This form of EMH is strongly supported by empirical evidence, such as Fama (1965a).

The semi-strong form of EMH suggests that the current price of a stock reflects all publicly available information. This includes not only historical information, such as prices, but also currently public information, such as quarterly reports, dividend announcements, announced merger plans, etc. The semi-strong form of EMH asserts, like the weak form, that one should not be able to profit from information that everyone else knows. Again, there is empirical evidence supporting this form of EMH, such as Fama et al. (1969).

The final, and most controversial, form of EMH is the strong form. The strong form states that the current price of a stock fully incorporates all existing information, both public and private. This differs from semi-strong because in strong no one should be able to profit when trading on information not publicly known at the time. That is to say, the EMH states that even if one is trading with insider information it is still difficult to profit from this information. The reason for this, as claimed in the EMH, is that the market anticipates future developments and therefore the stock already incorporates the information. This form of the EMH is the most disputed and much empirical research contradicts it, such as Jaffe (1974).

The EMH relies on an efficient market. An efficient market is the result of the intense competition among investors to profit off new information. Because of this intense competition, almost all investors spend time and resources to detect mispriced stocks. Therefore, the actual likelihood of being able to detect and exploit such stocks becomes smaller and smaller. Consequently, only a relatively small number of investors profit from the detection of them.

Although the EMH is a cornerstone of modern financial theory, it is controversial and often disputed; its acceptance is not universal (Johnston and Johnston, 2005, p. 26). Believers argue it is pointless to search for undervalued stocks or to try to predict trends in the market through either fundamental or technical analysis.

Despite research supporting EMH, there is a substantial amount contradicting it (Malkiel, 2003b). There is much literature supporting the observation that stock prices appear to respond to earnings for about a year after they are announced. The stock prices of companies with positive earnings tend to drift upward, whilst the stock prices of companies with negative earnings tend to drift downward (Ball and Brown, 1968). Ball and Brown (1968) first proposed this idea in their paper, 'An empirical evaluation of accounting income numbers'. Their findings have since been replicated by numerous studies covering different time periods (L Bernard and K Thomas, 1989; Bernard and Thomas, 1990).

2.2.3 Stock Market Prediction

Investors in the stock market generally fall into two categories: fundamental analysts and technical analysts, with the latter incorporating technological methods (Khan and Zuberi, 1999, pg.85). Fundamental analysts believe that the price of a stock and its future price movements derive from the stock's relative data. Fundamentalists evaluate a company's past performance by using information such as earnings, dividends, new products, research, management effectiveness, etc., to determine future forecasts. Although most analysts use fundamental analysis to value stocks, this method of valuation can be used for just about any type of stock (Thomsett, 2000).

Technical analysts ignore the company's fundamentals. Instead, they seek to determine the future price of a stock based solely on its historical statistical data as to identify trends in price and volume. They believe that market timing is key. Technical analysts examine what investors fear or think about a company and whether investors have the means to back up their opinions (Pring, 2002). This approach includes methods that utilise technology, which are becoming increasingly more prevalent. Technological approaches are discussed below.

2.3 Literature Review

There are thousands of papers into stock market prediction. This section focuses on papers which utilise computational analysis for this task. Also in focus is research using electronic forms of information, such as web pages, blogs, etc.

Kim (2003) attempted to forecast the direction of daily price change of a stock using technical indicators as input variables. Their objective was to investigate how effective support vector machines (SVM) were in this task. In addition, they also compared the SVM approach with an Artificial Neural Network (ANN) approach, and a case-based reasoning (CBR) approach. They found that SVM's accuracy was 64.8%, outperforming a back-propagation neural network (58.5%), which was the second most accurate. They highlighted that the prediction performances of SVMs are highly sensitive to the value of upper bound C and the kernel parameters.

In similar research to Kim (2003), Huang et al. (2005) looked into forecasting the weekly movement direction of NIKKEI 225 index using SVM in combination with financial macroeconomic variables of the NIKKEI 225 Index. They also compared SVM's performance, this time with Linear Discriminant Analysis, Quadratic Discriminant Analysis, and Elman Back propagation Neural Networks. Their results show that SVM outperforms the other classification methods, with a hit ratio of 73%, compared with random walk of 50%. The best results, however, were obtained by integrating SVM with the other classification methods, which resulted in a hit ratio of 75%.

In research more inline with the aims of this thesis, Wthrich et al. (1998) conducted some of the first research into using electronic information to predict the stock market. Wthrich et al. developed a system for predicting the opening stock prices based on the contents of the electronic stories downloaded from the Wall Street Journal. Morning version of articles that contained financial analysis and information about what happened on the world's stock, currency, and bond markets were downloaded. From those articles, they attempted to predict the daily closing values of major stock markets in Asia, Europe, and America. Several techniques were tested and compared. These include Rule-Based approaches, the k-NN Algorithm, and Neural Networks. An interesting feature of this work, which few, if any, subsequent papers replicated, was the use of a priori domain knowledge. They used a dictionary consisting of 392 keywords, each considered a typical buzzword capable of influencing the stock market in either direction, which was defined by several experts. The techniques were tested for a three-month period, however none achieved accuracy greater than random chance.

The approach taken by Wthrich et al. (1998) of using online news article as an input source is a popular avenue of research. Gidfalvi and Elkan (2001) also used news articles to predict stock price movements. They focused on predicting the future direction of the price of a stock, whether it would go up, down, or remain unchanged. They looked at over 5,000 financial news articles concerning 12 stocks. Each article in the training set is labelled up, down, or unchanged, according to the movement of the associated stock in a time interval surrounding publication of the article. A Naive Bayesian text classifier is used to predict which movement class an article belongs to, i.e., whether it would go up, down, or remain unchanged. Their results demonstrate that there exists a weak ability to predict the direction of a stock before the market corrects itself to equilibrium. They identified this duration to be a period of twenty-minutes before and twenty minutes after a financial news article was released. The reason, they hypothesise, is that the news articles are reprinted through various media outlets, so it takes time for the information to disseminate. In addition, they suggest that if one could identify the first release of a news article, then there exists a possibility to capitalise on stock price movements before human traders can act.

Fung et al. (2002) also used news articles as an input in an attempt to predict changes in the stock market, but their algorithm differed to the ones used by Wthrich et al. (1998) and Gidfalvi and Elkan (2001). They looked at over 350,000 news articles collected for seven consecutive months. They performed regression analysis of technical data to identify price trends, and then used support vector machine analysis of textual news articles to perform a binary classification of stocks to decide whether the price will rise or drop. The system was evaluated using a market simulation. However, the paper provides only ambiguous results. They say that they were profitable, but no detailed statistical results are provided.

The approach of using news articles remains popular to researchers. Schumaker and Chen (2009) attempted to predict the actual price of stocks twenty minutes after a news article was released. Their input was 9,211 financial news articles and 10,259,042 stock quotes covering the S&P 500 during a five week period. Using text-mining techniques, they transformed the financial news articles into feature representations including bag of words, noun phrases, and named entities. Article and stock quote data was then processed by a SVM derivative, using Sequential Minimal Optimisation in a form of regression. Their results show that a model containing both article terms and stock price at the time of article release had the best performance in closeness to the actual future stock price (MSE 0.04261).

Slightly different to the previously mentioned articles is Mittermayer (2004), who attempted to predict stock price trends for the time immediately after the publication of press releases. This work looks into press releases and not news articles. This is noteworthy because the authors state that due to the Securities Exchange Act 1934 - which guarantees simultaneous public disclosure of ‘material non-public information’ – the potential importance of press releases are the better source of unexpected information compared to news articles. The system retrieves the press articles from PRNewswire and Businesswire immediately after their release. The articles are sorted into predefined categories using a SVM algorithm. There are three categories: Good News, Bad News, and No Movers. Finally, trading strategies are derived from the articles based on their category. Their system performed better than a random trading of securities and yielded an average profit of 0.11% per trade. They found that profits were maximised by buying or shorting stocks and taking profit on them at 1% up movement or 3% down movement.

News articles and press articles are traditional forms of information. Thomas and Sycara (2000) investigated two approaches to stock market prediction using a non-traditional source of information. They downloaded every post in four popular financial discussion boards covering a period of a year. They used two approaches: a maximum-entropy text classification algorithm and a genetic algorithm. Trading rules were constructed based on the trading volumes of the stocks concerned, as well as the number of messages and words posted on the web bulletin boards per day. The authors reported that the profits obtained increased up to 30% by integrating the two approaches rather than using either of them. However, no detailed analysis of the results is provided.

The work of Thomas and Sycara (2000) is pertinent to this thesis because it relied on user generated content. Indeed, user-generated content has drawn more attention of researchers from various disciplines. One of the largest mediums for user generated content is Twitter, which has been used for various research purposes.

Zhang et al. (2009) looked into trying to predict stock market indicators by analysing Twitter posts. They collected a small section of Twitter messages for a six-month period. Their dataset was roughly equivalent to one hundredth of the full volume of all tweets in that period. The messages were analysed to measure collective hope and fear on each day.

They then analysed the correlation between these indices and the stock market indicators. They found that emotional tweet percentage significantly negatively correlated with Dow Jones, NASDAQ and S&P 500, but displayed significant positive correlation to VIX. Their conclusion was that checking twitter for emotional outbursts of any kind gives a predictor of how the stock market will be doing the next day.

Using sentiment was also focus of the work by Bollen et al. (2010), which gained much publicity in the media[2]. Their paper, 'Twitter mood predicts the stock market', details their attempts at predicting the stock market using the general mood (or sentiment) of users on Twitter. They collected 9,853,498 tweets posted by approximately 2.7 million users. Sentiment analysis was applied to the tweets, and then a Granger causality analysis and a Self Organising Fuzzy Neural Network are used to test the hypothesis that public mood states are predictive of changes in the stock market. They claim an accuracy of 87.6% in predicting the daily up and down changes in the closing values.

The accuracy claimed by Bollen et al. (2010), and accuracies in general, must be understood in the correct context as they could ignore market trends. Market trends are tendencies of financial markets to move in a particular direction over time[3]. This is important to understand because in bull and bear markets the markets generally trend upward and downward respectively. A good example of this was India's Bombay Stock Exchange Index, SENSEX. The SENSEX was in a bull market trend for about five years from April 2003 to January 2008 as it increased from 2,900 points to 21,000 points. Therefore, predicting that an index with rise or fall in a bull or bear market is not a 50/50 chance.

Another paper that uses Twitter, and work that is most similar to that of this thesis, is the research by Wolfram (2010), who investigated whether Twitter could be used to model the stock market. Using a Twitter dataset collected by the University of Edinburgh,

[2]http://www.wired.com/wiredscience/2010/10/twitter-crystal-ball/ [accessed: 22-JUL-2011]
[3]http://en.wikipedia.org/wiki/Market_trend [accessed: 22-JUL-2011]

Wolfram extracted features to create a model to predict stock prices. The focus was on several technology stocks over a two-week period. Different feature representations were constructed, which were combined with the stock price to build a regression model using the Support Vector Regression algorithm. The results were promising, with their system able to predict discrete prices close to a strong baseline on average predicting 15 minutes ahead of the actual price, but for longer periods (30 minutes plus) the accuracy becomes unstable.

As the work of Zhang et al. (2009). Bollen et al. (2010), and Wolfram (2010) show, there is an increasing trend in using Twitter as a source of information.

Yi (2009) compared three sources of text from social media and built a predictive model using support vector regression for each text sources to predict a real valued price. Out of the three data sources, Twitter performed the best.

Tayal (2009) explored the two most common online media that the public use to express their opinions and sentiment about products. They compared traditional blogs with microblogs to determine the predicative power on stock prices. They performed sentiment analysis on both and found that micro-blogs consistently outperformed blogs in their predictive accuracy.

2.4 Twitter

Before proceeding, it is useful to introduce Twitter. Twitter describes its self as: "Twitter is a real-time information network that connects you to the latest information about what you find interesting" (Twitter, 2011). It is commonly known as a microblogging service. Twitter users can post tweets (messages), of up to 140 characters. By default, these are publicly available – which is one reason why it is a keen source of information for researchers. Users follow others, or are themselves followed. An interesting aspect of Twitter is that unlike many online social networking sites, such as Facebook, the relationship of following and being followed requires no reciprocation. If you follow someone, you view their tweets on your homepage, known as a timeline.

There are nuances to Twitter. RT stands for retweet, which is when a user tweets what another user has already tweeted, allowing users to spread information of their choice beyond the reach of the original tweet's followers. An '@' sign followed by a username is reply or a mention. Hashtags '#' are also extensively used by users to categorise their message to a certain topic. Twitter analyses these tags to determine trending topics (Kwak et al., 2010). In terms of raw numbers, Twitter claims that there are over 200 million tweets a day and over 200 million active users (Twitter, 2011).

2.4.1 Twitter in Research

Twitter has been used in academic research for various purposes.

Ritterman et al. (2009) looked into using Twitter to predict a swine flu pandemic. Using almost 50 million tweets covering roughly a two-month period, they showed that using Twitter could improve the accuracy of market forecasting models by providing early warnings of external events like the swine flu pandemic.

Tumasjan et al. (2010) investigated whether Twitter activity mirrored the political landscape in the 2009 German federal election. The authors applied sentiment analysis on over 100,000 tweets that had political references. Their results show that Twitter is frequently used for political deliberation and that the mere number of party mentions accurately reflects the election result. In addition, the authors found that the sentiment of the messages, in the case of this investigate positive and negative, corresponds closely to voters political preferences.

Twitter has also been used for earthquake detection. Sakaki et al. (2010) used Twitter as a 'Social Sensor' for real-time earthquake detection. The motivation for their work was to capture and identify earthquake occurrence promptly, as to notify the public. They devised a system to detect a target event. Their system starts by using a classifier of tweets based on features such as the keywords in a tweet, the number of words, and their context. They then produce a probabilistic spatio-temporal model for the target event that can find the centre and the trajectory of the event location. Finally, they consider each Twitter user as a sensor and apply Kalman filtering and particle filtering, which are widely used for location estimation in ubiquitous/pervasive computing. Their system can detect an earthquake with a probability of 96% and notify users by email much faster than the announcements that are broadcast by the Japan Meteorological Agency. In their examples, emails are sent up to eight minutes quicker than the announcements by the Japan Meteorological Agency.

2.5 Chapter Summary

This chapter introduced fundamental financial theory necessary for this thesis. Stock volume, and the rationale for predicting it, was covered. In addition, the literature review demonstrated that there is precedence for this work, including research into stock market prediction using techniques suggested by this thesis. Finally, the chapter introduced Twitter and provided examples of where it has been used successfully in academic research for the purposes of prediction. The following chapter documents the construction of the datasets required for the experiments.

Chapter 3

Constructing Datasets

This chapter documents the construction of datasets required for the thesis. The majority of this chapter concerns building the Twitter dataset. The chapter starts with an explanation of how the raw Twitter data was collected. The processing stage is then explained. The goal of this processing stage is the construction of a number of inverted indices, which are introduced in this chapter. The chapter concludes with a section on how the financial data was collected.

3.1 Twitter Data

To model the relationship between Twitter activity and stock volume, two datasets are required: one of tweets, the other of stock volumes.

3.1.1 Collecting the Twitter Data

There are several ways to obtain Twitter data. There are existing Twitter datasets, such as Kwak et al. (2010), which contains 106 million tweets, and Petrovic et al. (2010), which contains 97 million tweets and was used in the work of Yi (2009) and Wolfram (2010).

Unfortunately, most of these datasets are no longer publicly available. This is due to recent changes in Twitter's terms of service. The changes prohibit the redistribution of datasets collected, including for academic purposes (Watters, 2011). Therefore, this avenue is ignored.

The changes in Twitter's terms of service also state that Twitter will no longer grant white-listed IP addresses. This is significant because a white-listed IP address allows a user to make up to 20,000 requests per hour to the Twitter API. Without a white-listed IP address, the maximum number of requests a user can make per hour is 150 (Watters, 2011). Without a white-listed IP address, it is impractical to collect the volumes of Twitter data required for this thesis. Therefore, this avenue is also ignored.

3.1.2 Web Crawlers

A web crawler is a program that visits web pages in order to retrieve and index the information on them. Web crawlers are often employed by search engines in order to find and index as much of the web as possible. They also have the task of identifying the structure that interconnects them. In web pages, this is usually HTML links. The basic operation of a crawler is as follows: the crawler starts with a seed, basically a URL that is guaranteed to contain links to other pages. The crawler then fetches the web page at that URL, parses it, and extracts both text and links from the page. The text is stored or passed on to another process for indexing, whilst the extracted links are added to a frontier of URLs to visit. This process is repeated continuously (Manning et al., 2008).

This thesis uses a web crawler to crawl Twitter for tweets. This approach is advantageous because it permits the collection of large volumes of data unattainable using a non-whitelisted IP address with the Twitter API. It also allows greater flexibility on what information can be retrieved, i.e., one can go beyond what the API provides.

3.1.3 Building a Web Crawler

There are several ways to implement a web crawler. One option is to use a commercial 'screen scraping' application, such as Mozenda[1]. These allow users to scrape information from websites through a graphical interface. Using the software, the user selects the elements that they wish to scrape from a web age. The software records these elements and automates the crawler task. The applications allow users to select how they wish to navigate away from a page, replacing the logic of how web crawler uses links to navigate between pages. However, most of these applications are commercial, so this method is ignored due to financial constraints.

Another option is to use an open-source crawler, such as Nutch[2] or Scrapy[3]. Nutch, in particular, is a mature crawler which has been used in several commercial search engines (Khare and Cutting, 2004). However, the estimated time required to configure such an open-source crawler exceeds the estimated time required to create a customer crawler, thus, this thesis uses a custom web crawler.

To develop the crawler, the Python[4] programming language is used. Python has many useful features for web crawling. It is especially well regarded for its string manipulating capabilities[5]. Besides Python, two other components are used in the crawler: Beautiful-Soup[6] and MongoDB[7]. BeautifulSoup is a Python HTML/XML parser designed for quick turnaround projects like screen-scraping. It offers useful functionality, such as finding all links with a certain css class. MongoDB is a NoSQL, schema-free, document-oriented database.

The crawler follows the basic algorithm of all crawlers: A seed pair of a username and

[1]http://www.mozenda.com [accessed: 26-JUL-2011]
[2]http://en.wikipedia.org/wiki/Nutch [accessed: 26-JUL-2011]
[3]http://scrapy.org/ [accessed: 26-JUL-2011]
[4]http://www.python.org [accessed: 26-JUL-2011]
[5]http://stackoverflow.com/questions/635155/best-language-for-string-manipulation [accessed: 26-JUL-2011]
[6]http://www.crummy.com/software/BeautifulSoup/ [accessed: 26-JUL-2011]
[7]http://www.mongodb.org/ [accessed: 26-JUL-2011]

a URL are added to a frontier of pages to visit. One of a number of downloader threads then attempts to pop an item of the frontier. If successful, the crawler retrieves the HTML from the URL and adds it along with the username to another frontier of pages to parse. The username is added to another frontier, which represents users visited – done to prevent cycles. Concurrently, a parser thread attempts to pop an item off the to parse frontier. If successful, the parser extracts the user's name, location, and number of followers. This information is extracted because it is potentially useful for feature construction. The user's contacts are also extracted and each contact's username and URL are added to the to visit frontier. A contact is only added if the crawler has not already visited them, i.e., if the username is not in the visited frontier, thus preventing cycles. Next, the user's tweets are extracted. The tweet id, content, and date are stored in a tweets database. Finally, because Twitter uses pagination to display tweets, the parser extracts the URL for the next page of tweets for the current user, which is then added to the to visit frontier.

The developer crawled ran for approximately three weeks. It collected 57,782,207 tweets posted by 573,020 users over a period of four years. An overview of the distribution of the tweets is provided in figure 3.1. Hardware limitations, namely a lack of hard-drive space and insufficient processing power, prevented the crawler from running for longer. According to Twitter (2011), there are 200 million tweets per day, so the data collected only represents a fraction of the overall tweets. However, for this thesis, the data is considered sufficient.

3.2 Raw Dataset Cleanup

The raw data retrieved from Twitter contains a substantial amount of noise, i.e., anything irrelevant for the task in hand, such as common words, non-English tweets (as this thesis is only concerned with English tweets), etc. It is useful to remove this noise because it would not only interfere with the regression models, but also add computation time. To remove

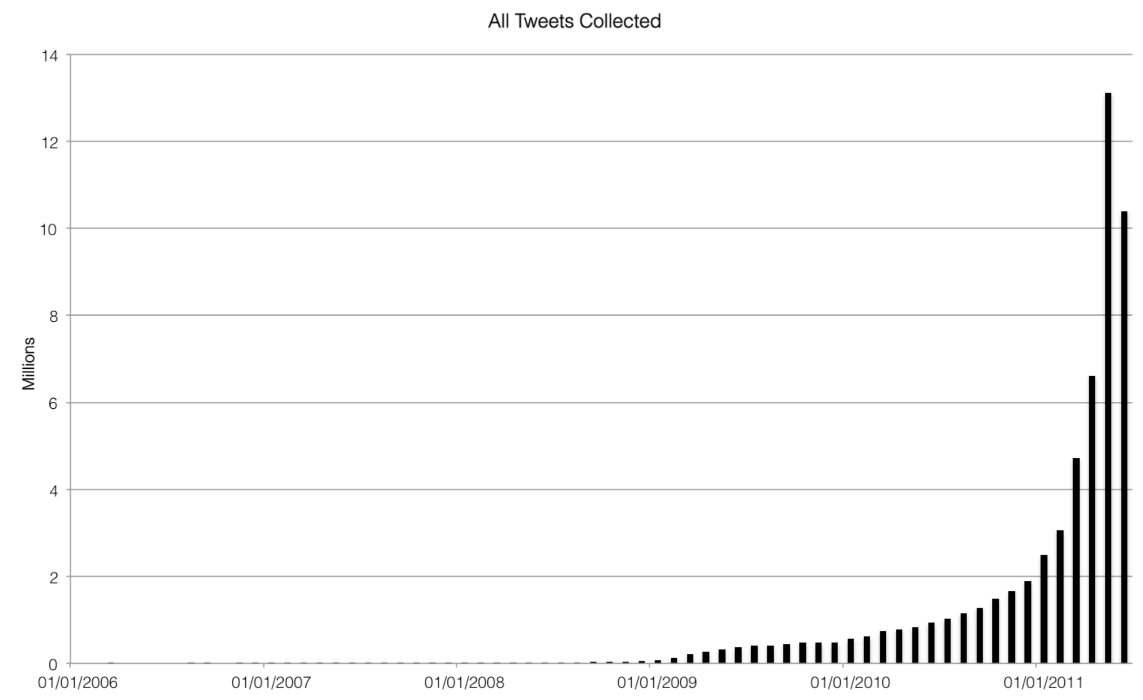


Figure 3.1: All Tweets Collected

noise, the dataset goes through a stage of pre-processing. The aim of this pre-processing is to reduce the dataset to a manageable size and into a format for efficient lookup.

The first step is to reduce the dataset to a specific date range. This date range should contain the mosts and have a consistent distribution throughout. As displayed in figure 3.1, most tweets collected are from 2011, so tweets before 2011 are removed. Figure 3.2 shows a monthly breakdown of reduced dataset.

Having reduced the dataset, the next step is to parse each tweet to build a model for feature selection.

3.3 The Standard Boolean Model

The Standard Boolean model is a common and simple model for retrieving documents in which a document either matches a query or does not. It is based on classical Set theory

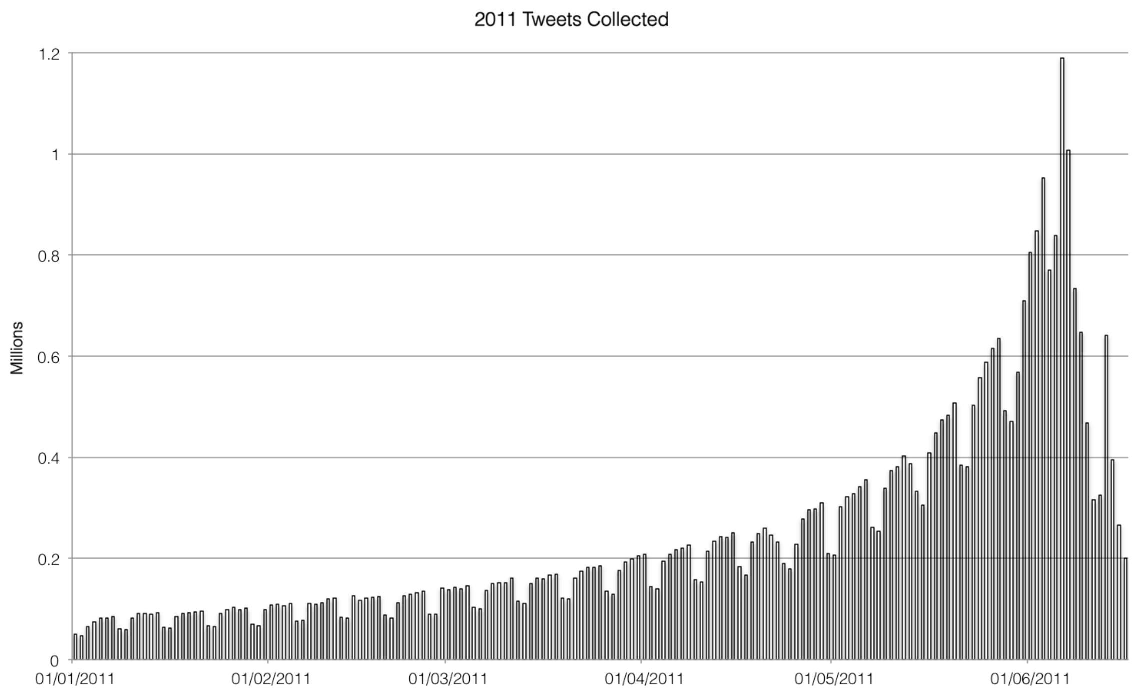


Figure 3.2: 2011 Tweets

in that both the query and the documents are conceived as sets of terms. This model disregards grammar, word order, and the frequency of terms (Manning et al., 2008). It can be demonstrated with an example, consider the following query q_1 and documents d_1 and d_2.

q_1 = I fancy a toffee apple.

d_1 = My favourite flavour of toffee is apple.

d_2 = The toffee store on Apple Street sells toffee apples.

The Boolean model considers both the query and document as sets. Thus, ignoring grammar and case sensitivity, the query and documents would be represented as the following sets.

$$q_1 = (\text{i, fancy, a, toffee, apple})$$

$$d_1 = (\text{my, favourite, flavour, of, toffee, is, apple})$$

$$d_2 = (\text{the, toffee, store, on, apple, street, sells, apples})$$

Queries in the Boolean model are basically set operations. To perform a query that tries to find documents matching q_1, a linear scan is performed where each document is considered in turn and documents are matched if they contain the query terms. In the above example, both d_1 and d_2 would be matched and their relevance be deemed equal. A linear scan using a modern computer on a modest collection of less than one million words is practical, however, for larger collections or more complex queries a more efficient method is required (Manning et al., 2008).

For faster lookup, it is beneficial to index the documents in advance. One technique is to create a binary term-document incidence matrix. In the matrix, there is vector for each document and an element for each term in the vocabulary. If the term is in the document, then the element has a 1; otherwise a 0 (Manning et al., 2008). An example is displayed in figure 3.3. To perform a query, a bitwise AND operation is performed on the query term vectors to identify matched documents.

Constructing a term-document incidence matrix for all 40 million tweets and all words is impractical. The matrix is likely to be sparse, as most documents do not contain most words in a vocabulary. A more efficient representation is to record only the things that do occur, i.e., the 1 positions. This is called an inverted index.

$$\begin{array}{ccccc}
 & d_1 & d_2 & \dots & d_n \\
t_1 & td_{11} & td_{12} & \dots & td_{1n} \\
t_2 & td_{21} & td_{22} & \dots & td_{2n} \\
\vdots & \vdots & \vdots & & \vdots \\
t_n & td_{n1} & td_{n2} & \dots & td_{nn}
\end{array}$$

Figure 3.3: Term Document Incidence Matrix

3.4 Inverted Indices

In an inverted index there is a dictionary of terms and for each term there is a list of documents containing the term. Each item in the list is called a posting and the list is called a postings list. Postings can include extraneous information, such as the frequency of the term, or the position of the term within the document (Manning et al., 2008). The following is an example of an inverted index, where t_i is a term and d_i is a document containing the term.

$$t_1 = (d_1)$$

$$t_2 = (d_1, d_2, d_4)$$

$$t_3 = (d_3, d_4)$$

Other models besides the Boolean model were considered. N-gram models, which consider a sequence of n words, were investigated, but were disregarded because of the storage and computational requirements. In addition, a bag of words model was also considered. In this model, document term frequencies are maintained, which are useful in many weighting schemes, such as term frequency inverse document frequency. However, the Boolean model is preferred to the bag of words model because of the length of Twitter messages.

	Mean	Median	Mode
Words	20	21	28
Unique Words	18	29	24

Table 3.1: Average Lengths of Tweets

Tweets are limited to 140 characters. This limits what a user can say in terms of ideas and actual words. In a sample of 500,000 tweets, it was calculated that the average number of words used was 21 and the average number of unique words was 19. The averages are displayed in table 3.1. The averages are case insensitive and include stop words. Because, on average, few words are repeated in a tweet, the value of maintaining term document counts was minimal, thus, the standard Boolean model is considered sufficient.

3.4.1 Constructing Inverted Indices

Having chosen a model, the next task is to construct the inverted file indices. This task is complicated because of the temporal element of the thesis, i.e., matching Twitter activity and stock volume on specific days. A number of alternative techniques were investigated. Most using multidimensional access methods such as K-D trees[8]. However, the inverted indices method was chosen due to relative ease of construction, compared with that of K-D trees.

Having decided on inverted indices, the exact implementation is to be decided. One option is to create an index for terms and documents, and one another for dates and documents. However, this requires multiple indices to represent the same thing, thus, is simpler approach is taken. The date is embedded into the key of the term, thus requiring one index to represent the same thing. Overall, four indices are constructed, listed below:

Index 1: - Dates and words to tweets: used to identify tweets mentioning a word on a

[8]http://en.wikipedia.org/wiki/K-d_tree [accessed: 28-JUL-2011]

That was a long day's filming. My 100 Greatest Gadgets for C4. Shan't giveaway No 1, but it's not what you think. Nor that. Nor even that.

Figure 3.4: Raw Tweet

Ooh, and don't forget that today is Towel Day `<ahref="http://t.co/xTv9uIo"class="tweet-urlweb"rel="nofollow"target="_blank">http://t.co/xTv9uIo</a> <ahref="search?q=%23towelday"title="#towelday"class="tweet-urlhashtag"rel="nofollow">#towelday</a>`

Figure 3.5: Raw Tweet with a Hashtag

particular date.

Index 2: - Dates and hashtags to tweets: used to identify tweets that hashtag a word on a particular date.

Index 3: - Dates and words to users: used to identify users mentioning a word on a particular date.

Index 4: - Dates and hashtags to users: used to identify users mentioning a hashtag on a particular date.

The latter two are seen as potentially useful because by indexing users, rather than tweets, single anomalies or exaggerations can be diminished. For example, a single user may repeatedly post the same tweet non-stop throughout the day.

Although the four indices require different algorithms for construction, the general steps are the same. The following section details the steps taken to construct index 1, dates and words to tweets.

The first step is to remove noise from the tweets. One major form of noise in tweets is the HTML code used by Twitter to differentiate tweet types. Tweets can be categorised into four types: normal tweets, tweets with hashtags, direct tweets, and retweets. Figures 3.4, 3.5, 3.7, and 3.6 show raw examples of normal, hashtag, direct, and retweets respectively.

Check her out! RT `@<aclass="tweet-urlusername"href="/RachelPlatten"rel="nofollow">RachelPlatten</a>Hinyc!MyAlbumreleaseshowistonightatlepoissonrouge!7:30pm,Ihopetoseeyou!!Pleaseretweet:)`

Figure 3.6: Raw Tweet that is a Retweet

```
@<aclass="tweet-urlusername"href="KazimAliPoet"rel="nofollow">
KazimAliPoet</a>sogreatmeetingyou.Readingitontheplane:)
```

Figure 3.7: Raw Tweet that is a Direct Tweet

Recognising the different types of tweets, especially direct tweets and retweets, may be useful for modelling stock volume. For instance, the frequency and distribution that a tweet is retweeted may have a relationship to a company's stock volume. However, this is ignored in this thesis, and it left for investigation in future work. Thus, when a tweet is parsed, its content is stripped of HTML code. To remove HTML from the content of tweets the clean_html function provided by the Natural Language Toolkit[9] (NLTK) is used. The NLTK is a suite of libraries for symbolic and statistical natural language processing for the Python programming language and is used in other parts of the index construction.

Note: when constructing the hashtag indices, HTML is not removed because, as displayed in figure 3.5, hashtags are identified by the CSS class 'tweet-url hashtag'. Hashtags are extracted using a regular expression. In addition, when constructing the hashtag indices, no tokenisation takes place. The hashtags identified with the regular expression are just checked to whether they are words.

The next step in the index construction is tokenisation. Tokenisation is the process of chopping up a character sequence, most often a sentence into pieces, called tokens (Manning et al., 2008). To perform tokenisation, the NLTK 'wordpunct_tokenize(text)' function is used. As part of the tokenisation step, tokens are made lower case and tokens that are not alphabetic are removed, i.e., a token is deemed a word if and only if all characters in the

[9] http://www.nltk.org [accessed: 29-JUL-2011]

Input: Off to Lord's for a day's cricket. First day off for what seems months.
Output: (a, cricket, day, first, for, lord, months, off, s, seems, to, what)

Figure 3.8: An example of tokenisation

token are alphabetic and there is at least one character. Finally, duplicates are removed. They are removed because the standard Boolean model is used, so frequencies are irrelevant. Figure 3.8 show an example of tokenisation.

After tokenisation, the parser checks each word to see whether it is English, and non-English words are removed. This is a limitation of the thesis; considering different languages is left for future work. Again, there were several methods to detect the language, such as Google's Language Detection API[10], or using an n-gram-based language identifier (Carter et al., 2011). Both of these approaches are deemed overly complex for this task. A simpler technique is to check each word with an English dictionary. This approach, however, has a disadvantage in that although a dictionary would capture most of the tokens, some tokens would be excluded. For instance, stock symbols, such as APPL (Apple Inc) would likely be excluded. Additionally, other terms such as 'IPod' or 'GMail' may be missing in dictionaries. For this work, a simple, although naive, technique is used. Every letter in a token is checked to see whether it is in the set string.ascii_lowercase, provided by Python. If not, the token is ignored. This technique may result in some non-English words being included, but this is preferred to excluding English words. The decision can be seen as a trade-off of recall vs. precision. Note: this technique may sill exclude some words, such as cafè, however, such instances are deemed acceptable given the size of the dataset.

Also at this stage common words, known as stop words, are removed. They are removed because, due to their high frequency in most documents, they are of little value in identifying relevant documents. Stop words are removed by checking each token with a list of stop words provided by NLTK. In addition, tokens with only a single character are removed.

[10]http://code.google.com/apis/ajax/playground/#language_detect [accessed: 29-JUL-2011]

The remaining tokens are deemed words and are added to the inverted index. Redis[11], an open-source, in-memory, persistent, key-value data store, it used to store the indices. This was chosen over MongoDB because in a small test it performed better.

3.5 Financial Data

In the preliminary stages of this thesis, the aim was to model stock volume on an intra-day basis. This, it was reasoned, would take advantage of the almost instant nature of Twitter, i.e., people tweeting as soon as an event happens. However, following the literature review it became evident that volume is seen as in indicator of trend, thus predicting daily intervals is adequate. Another obstacle with intra-day prediction is the lack of free intra-day volume data. Crawling for the data was considered, but hardware limitations, namely a lack of hard drive space, prevented this.

3.6 Chapter Summary

This chapter documented the methodology for collecting and processing the data required for the thesis. A web crawler collected over 60 million tweets, which was reduced to 40 million tweets following a stage of pre-processing. The dataset was then used to create four inverted file indices. Overall, this chapter laid the foundations for the rest of the thesis. The data collected and processed in this chapter is used in the following chapter to model the relationship between Twitter activity and stock volume.

[11]http://redis.io [accessed: 26-JUL-2011]

Chapter 4

Modelling the Market

The previous chapter described the construction of the Twitter dataset. This chapter explains how the dataset is used in support vector regression to model the relationship between Twitter activity and stock volume. This chapter includes sections on evaluation methods, baselines, and validation methods. In addition, support vector machines and their regression counterparts are detailed. The chapter concludes by stating the methodology for the experiments in the following chapter.

4.1 Stock Selection

Four stocks are investigated:

1. Apple Inc. (AAPL)
2. Google Inc. (GOOG)
3. Vodafone GRP (VOD)
4. Burberry Group plc (BRBY)

The choice of this number of stocks is largely arbitrary, however one reason for keeping the number relatively small (compared with the total number of stocks in the world), is that training support vector machines is computationally intensive. Thus, four is a manageable size.

The choice of stocks, however, is not arbitrary. APPL and GOOG are chosen due to their popularity in Twitter. In 2010 APPL and GOOG took up six out of the top ten trending topics of year[1], as displayed below.

1. Apple iPad
2. Google Android
3. Apple iOS
4. Apple iPhone
5. Call of Duty Black Ops
6. New Twitter
7. HTC
8. RockMelt
9. MacBook Air
10. Google Instant

The reasons for choosing these two popular companies is that tweet activity will be high and distributed throughout the dataset, not concentrated on a few major events. This should provide a stable baseline of Twitter activity, so that the SVM can learn to identify spikes and trends.

In addition to these popular stocks, VOD and BRBY are also chosen. VOD is chosen because it is one of the most traded stocks on the FTSE[2] with an average volume approximately four times higher than that of APPL and nearly 50 times higher than that of GOOG.

BRBY, however, is chosen because of prior knowledge about the relationship between BRBY's Twitter Activity and BRBY's stock volume. In the preliminary stages of this

[1]http://yearinreview.twitter.com/trends/ [accessed:26-AUG-2011]

[2]http://www.hl.co.uk/shares/stock-market-summary/ftse-100/top-volume [accessed:01-AUG-2011]

thesis, the relationship between Twitter activity and stock volume was explored. To do this, the now retired Google Real-Time search was used. This service allowed users to view historical Twitter activity. Using this site, a number of instances were found where a significant increase in Twitter activity preceded a significant increase in stock volume. One such instance involved BRBY.

From 01 Jan 2011, Twitter activity for BRBY is stable until Monday 21 Feb. Then there is a 7000% increase in Twitter activity relating to BRBY. The Twitter activity returns to its previous levels by Thursday 24 Feb. Meanwhile, BRBY's stock volume remained stable until Thursday 24 Feb, when it increased 9000% on the previous day. The volume returned to its previous levels the following day. This sudden increase in activity can be accounted to BRBY's impressive performance at the London Fashion Week (Kirka and Katz, 2011).

Although the original findings cannot be displayed, because Google Real-time search has been retried, the findings can be displayed using the data collected in chapter three. Figure 4.1 displays the BRBY's stock volume and the number of Tweets mentioning BRBY. Figure 4.2 shows the specific period under examination. Note that in the figures, the Twitter activity displayed is the number of tweets that mention BRBY. The figures are also normalised, which is explained later in this chapter.

Instances such as this may be rare, but do occur; similar instances were found for other companies.

With the exception of BRBY, the stocks have stable volumes. The mean daily percentage change in volume from 01 Jan - 31 May for APPL, GOOG, and VOD is 9.1%, 8.1%, and 9.6% respectively. In comparison, BRBY has a mean daily percentage change of 140%.

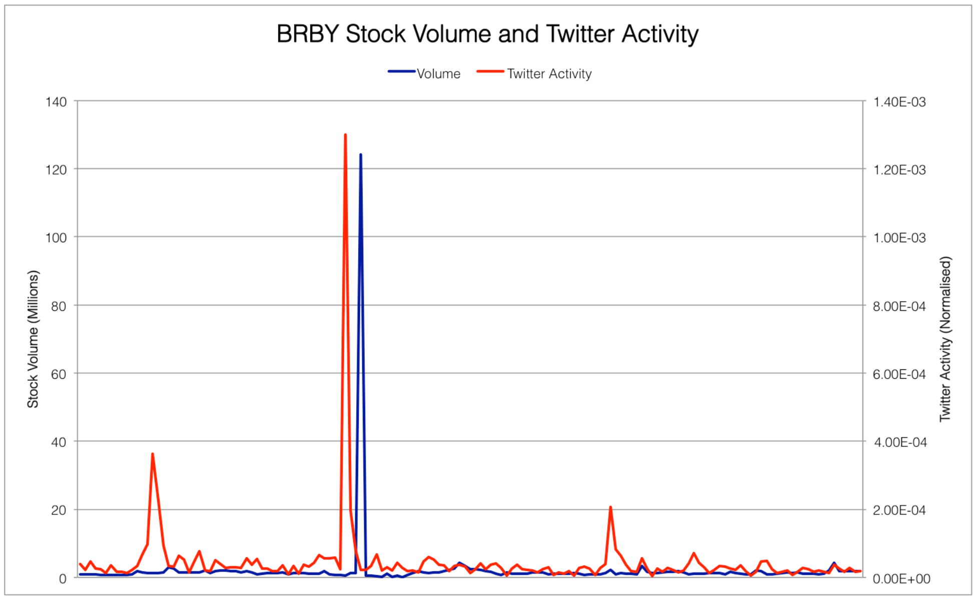


Figure 4.1: BRBY Stock Volume and Twitter Activity - 2011

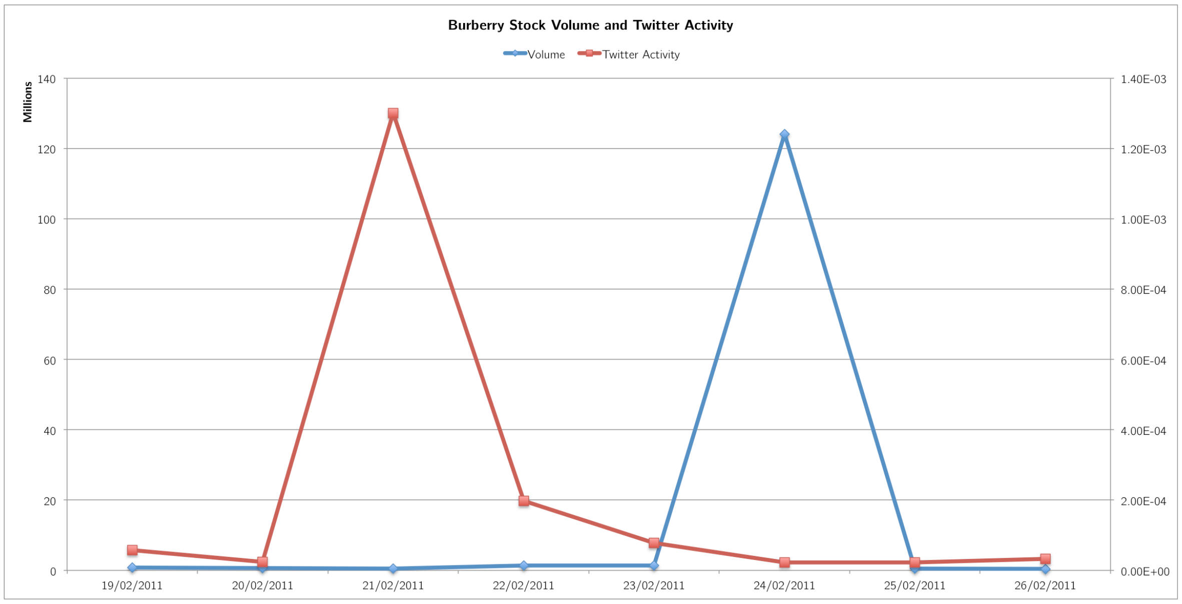


Figure 4.2: BRBY Stock Volume and Twitter Activity - 19 26 Feb 2011

4.2 Query Expansion

From the inverted indices constructed in chapter three, features can be extracted. The base feature is the number of tweets on a day mentioning a company name. This feature alone is naive and simplistic. Often a tweet refers to a company without explicitly stating the company's name. For example, the following tweet refers to Apple without stating Apple:

"Got an iPhone 4 woooo I'm in the cool group now ;)"

It is useful to identify such tweets. However, using the Boolean mode, described in chapter three, if a document does not contain exact terms that are in the query, then that document will not be retrieved.

Query expansion aims to reduce this query/document mismatch by expanding queries using words or phrases with a similar meaning or some other statistical relation to the set of relevant documents. The general objective is to increase recall. A query expansion algorithm may use a number of different techniques to expand a query, these include, stemming, synonyms, acronyms, etc. In this thesis, query expansion is used to find tweets related to a company that do not explicitly mention the company's name. The approach is based on the work by Wolfram (2010).

4.2.1 Google Sets

Google Sets is a prediction service that groups similar words. It is used in this thesis for query expansion. For each company, its name and stock symbol is entered into Google Sets. The top five results for each stock are displayed in table 4.1. These results are used in a query score feature described later in this chapter.

APPL	GOOG	VOD	BRBY
appl	google	vodafone	burberry
apple	goog	vod	gucci
mac	chrome	vedia	chanel
ipod	android	vimpelcom	zara
macbook	gmail	valups	nutella

Table 4.1: Google Sets Query Expansion Results

4.3 Evaluation Methods

Before explaining how Twitter activity and stock volume are modelled, it is important to state how predictions are evaluated. Measures of forecast accuracy can be placed into one of two sets, those that are 'scale dependent' and those that are not (Hyndman and Koehler, 2006). The root mean squared error (RMSE) is considered the standard measure in time series forecasting (Wang and Bovik, 2009). The RMSE, however, is not perfect. Thus, predictions are evaluated using two methods: one a scale dependent measure, RMSE, the other a scale independent measure, mean absolute percentage error (MAPE). The primary reason for evaluating results using MAPE, in addition to RMSE, is that the results are easier to interpret. Each method is explained below, but basically MAPE provides a percentage error, understandable to most people, whereas RMSE provides an output that is relative to the scale it is taken on, which can become extremely large and difficult to interpret.

4.3.1 Root Mean Squared Error

The RMSE is a measure of the differences between values predicted by a model and the actual values. It is based on the mean squared error (MSE), however, because the RMSE is measured in the same units as the data, rather than in squared units, it is usually more representative of the size of a 'typical' error. The RMSE and the MSE have been the dominant quantitative performance metric for more than 50 years (Wang and Bovik, 2009). Given two finite sized vectors x, y of length n, such that x = $x_1, x_2, \ldots, x_n$, and y = $y_1, y_2, \ldots, y_n$, where n is the number of elements in the vectors, the RMSE of the two vectors is:

$$RMSE(x, y) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2}$$

In the above example, x represents the vector that is considered true and y the vector of predictions.

An advantage of the RMSE is its simplicity. It is parameter free, inexpensive to compute, and memory-less (Wang and Bovik, 2009). There are, however, several disadvantages. The measure depends on the units in which the variable is measured. This can make it difficult to compare systems. Indeed, even the same system may perform different at different time-periods simply because of something like inflation. In addition, because the RMSE is scale variant, there is no absolute criterion for a ‘good’ value. Another disadvantage is that because of the squaring process it is sensitive to the occasional large error. This makes it sensitive to stock market crashes and similar occurrences.

4.3.2 Mean Absolute Percentage Error

The mean absolute percentage error (MAPE) is a scale independent measure of accuracy. It is useful for purposes of reporting because it is expressed in generic percentage terms, which will make some kind of sense even to someone who has no idea what constitutes a ‘big’ error. A disadvantage of this measure is that it can only be computed with data that is guaranteed to be positive. Given two finite sized vectors x, y of length n, such that x = $x_1, x_2, \ldots, x_n$, and y = $y_1, y_2, \ldots, y_n$, where n is the number of elements in the vectors, the RMSE of the two vectors is:

$$MAPE(x, y) = \frac{1}{n} \sum_{i=1}^{n} |\frac{x_i - y_i}{x_i}|$$

Most textbooks recommend the use of the MAPE and it was the primary measure in the M-competition – a competition for comparing forecasting methods (Hyndman and Koehler, 2006).

4.4 Baseline

The experiments in this thesis are evaluated against the actual stock volume of a company. Matching this value is the best-case scenario. However, it is also useful to compare the predictions to a baseline. This allows the predictions to be compared to something more attainable.

4.4.1 Moving Average

In statistics, a moving average is used to analyse a sequence of data points by creating a series of averages of different subsets of the full data set. It works as follows: given a time series sequence of data points that is of a finite size, a subset is created from the first n items. The moving average is obtained by taking the average of the first subset. The subset is then shifted forward, as to create a new subset. This process is repeated over the entire data series. The usual effect of a moving average is that short term fluctuations are smoothed out, leaving long-term trends or cycles. The bias towards short-term fluctuations and long term-trends can be adjusted by changing the moving average size. There are various moving averages, such as weighted and exponential. In these weights are given to more recent events, as to diminish the influence of older data. In this thesis, the simple moving average is used

Stock	MAPE	RMSE
APPL	0.227	6497221
GOOG	0.231	1381626
VOD	0.241	30604842
BRBY	2.030	12374632

Table 4.2: Two Day Moving Average

as a baseline.

4.4.2 Simple Moving Average

A simple moving average is the series of unweighted averages in a subset of time-series data points. The subset progresses over the entire data series. For each subset, the average of all values is taken. This produces a set of numbers representing the final moving average. It is important to state that the simple moving average is a not measure of prediction, but a measure of the current trend. Mathematically, the SMA can be described as, where n is the moving average number:

$$MA(t) = \sum_{i=1}^{n} \frac{x_t + t_{t-1} + \cdots + x_{t-n}}{n}$$

A SMA average was chosen for the baseline due to its simplicity.

4.4.3 Baseline Construction

Having chosen the SMA as the baseline, the next step is to choose the window size. To do this, the SMA for two, three, and four days were calculated for the stock volumes of APPL, GOOG, LAD, and BRBY for the period of 01 Jan - 31 May. Tables 4.2, 4.3, and 4.4 display the results for two, three, and four days respectively.

Stock	MAPE	RMSE
APPL	0.258	7174737
GOOG	0.265	1518551
VOD	0.258	31157256
BRBY	2.3328	11729338

Table 4.3: Three Day Moving Average

Stock	MAPE	RMSE
APPL	0.279	7737101
GOOG	0.291	1616953
VOD	0.269	31566141
BRBY	2.739	11404821

Table 4.4: Four Day Moving Average

With the exception of BRBY, the two-day SMA achieves the best accuracy. The poor accuracy for BRBY is as expected, as it is known to have the significant peak of activity towards the end of February. Thus, the two day SMA is chosen as the baseline the experiments.

4.5 Support Vector Machines

Having stated the evaluation methods and baseline, support vector machines (SVM) are now introduced. SVMs are a form of supervised learning, which is a branch of statistical machine learning that learns to map inputs to outputs following a period of training, i.e., it infers a function from training data. In supervised learning, the training data consists of a series of pairs each containing input objects, which are usually D dimension vectors whose components are referred to as features or attributes, and a desired output, known as labels. There are two sub genres to supervised learning: classification and regression. In classification the output variables are discrete, such as a class label. In regression the output is continuous, such as a real number. Using the scenario of predicting a stock value, an example of classification is predicting whether the stock will rise or fall. An example of regression is predicting the actual stock value. The learning algorithm produces an inferred function, which is called a classifier (if discrete) or a regression function (if continuous), after training. The inferred function is used to predict the outputs of future unseen inputs (Parrella, 2007).

SVM are a vector space based supervised machine learning technique. They are generally used for binary classification problems. The transition to regression is a small step but contains a significant difference in loss function. The goal of SVM is to find a decision boundary between two classes that is maximally far from any point in the training data. This boundary is known a maximum margin decision hyperplane. It is called maximum (also referred to as optimal) since there is only one that can maximise the margin. In the simplest case, the maximum margin is the longest distance that separates the two closest and linearly separable points from two opposing classes. Such points are known as the support vectors (Manning et al., 2008). Figure 4.3 shows an illustration of a SVM.

4.5.1 Linear SVM

Given a set of L training points, each containing an input x_i with D features and y_i = -1 or +1, the training data is said to be of the form:

$$(x_i, y_i) \mid i = 1, \cdots, l, x_i \in \mathbb{R}^D, y_i \in \{-1, 1\}$$

Assuming that the training data is linearly separable, i.e., it can be completely separated by a single line, the SVM tries to find the maximum margin hyperplane that divides the points having y_i = -1 from those having y_i = +1. A hyperplane can be written as the set of points satisfying:

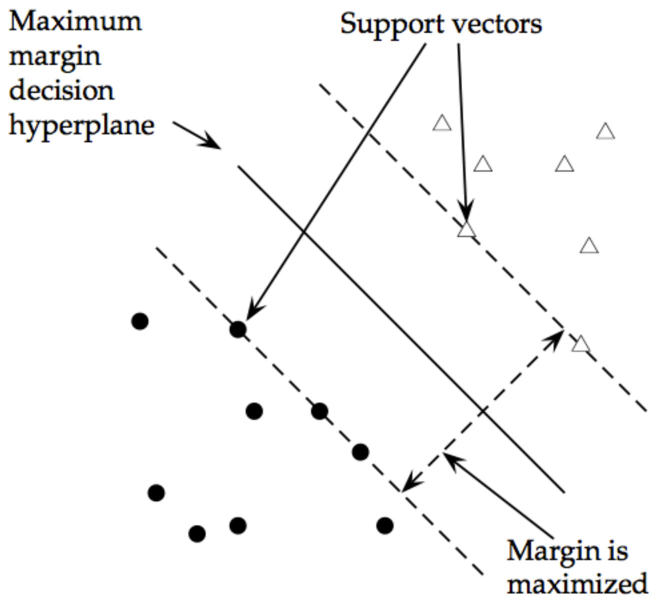


Figure 4.3: Support Vector Machine Example (Manning et al., 2008)

$$w \cdot x + b = 0$$

Here, $\cdot$ denotes the dot product and w is normal to the hyperplane. The parameter $\frac{b}{||w||}$ is the perpendicular distance from the hyperplane to the origin. The SVM tires to choose values for w and b as to maximise the margin, i.e., the distance between the parallel hyperplanes that are as far apart as possible whilst still separating the data. These hyperplanes can be described by the equations:

$$x_i \cdot w + b \geq +1 \; for \; y_i = +1$$

$$x_i \cdot w + b \leq -1 \; for \; y_i = -1$$

Which can be combined into:

$$y_i(x_i \cdot w + b) \geq 1 \; i = 1 \cdots l$$

The SVM then solves following optimisation problem, known as Quadratic Optimisation Problem:

$$\min_{w} \frac{1}{2} w^T \cdot w$$
$$s.t. \; y_i(w^T \cdot x_i + b) \geq 1 \; i \cdots l$$

4.5.2 Soft Margin

If there exists no hyperplane that can cleanly split two classes, for example due to mislabelled or aberrational training data, the Soft Margin method chooses a hyperplane that splits the points as cleanly as possible, whilst still maximising the distance to the nearest cleanly split points. This method introduces two variables ξ and C, which define the limit of tolerance. The quadratic optimisation problem is then increased by a function which penalises non-zero ξ_i, and the optimisation becomes a trade off between a large margin, and a small error penalty. If the penalty function is linear, the optimisation problem becomes:

$$\min_{w,\epsilon,b} w^T \cdot w + C \sum_{i=1}^{l} \xi i$$
$$s.t. \; y_i(w^T \cdot x_i + b) \geq 1 - \xi, \; \xi_i \geq 0 \; i = 1$$

The C parameter controls the trade off between errors of the SVM on training data and margin maximisation. Whilst a small value for C will increase the number of training errors, a large C will lead to a hard-margin. Finding the right value for ξ and C is considered complicated. Generally, the most commonly used technique is to find them by trial and error (Parrella, 2007). These parameters are discussed later in this chapter.

4.5.3 Non-Linear SVM

To handle non-linear classifiers, SVM employ kernels to map points to a higher dimensional space, referred to as 'the kernel trick'. Rather than compute the transformation in terms of dot product, it is substituted for another function, called the kernel function given as $K(x_i, x_j) = \phi(x_i^T) \cdot \phi(x_j))$, which allows for mapping into a higher or infinite dimensional feature space. The two most commonly used families of kernels are polynomial kernels and radial basis functions (Manning et al., 2008). Kernels are covered in detail later in this chapter.

4.5.4 Support Vector Regression

The previous section detailed SVM concerned with classification. For this thesis, modelling and predicting continuous data is the goal, thus regression is required. Support Vector Regression (SVR) is based on SVM. There are, however, some differences. Because the output is now a real number, there are infinite possibilities. The quadratic optimisation problem becomes:

$$\min_{w} \frac{1}{2} w^T \cdot w$$

$$s.t. \begin{cases} y_i - (w^T \cdot \phi(x) + b) \leq \epsilon \\ (w^T \cdot \phi(x) + b) - y_i \leq \epsilon \end{cases}$$

In addition, there is a bound added in order to set the tolerance on the number of errors that can be committed:

$$\min_{w} \frac{1}{2} w^T \cdot w + C \sum_{i=1}^{l} (\xi_i + \xi_i^*)$$

$$s.t. \begin{cases} y_i - (w^T \cdot \phi(x) + b) \leq \epsilon + \xi_i \\ (w^T \cdot \phi(x) + b) - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0,\ i = 1, \cdots, l \end{cases}$$

This thesis uses SVR to model twitter activity and stock volume as to predict next day volume.

4.6 Support Vector Regression Implementation

Scikits-learn[3] is used to implement SVR. Scikits-learn interfaces with the libsvm library (Chang and Lin, 2011). The original libsvm library was also considered, but the sci-kits implementation is preferred due to its interoperability with other areas of code used in the

[3]http://scikit-learn.sourceforge.net/stable/

thesis, namely the use of NumPy[4] during chapter three.

4.6.1 Feature Construction

The first task to perform SVR is to choose the features required to train the SVM. For SVR, three base features are chosen.

1. Word Count: The number of tweets on day d that company c is mentioned in
2. Hash Count: The number of tweets on day d that company c is hash tagged in.
3. Query Score: The number of tweets on day d that mention any of the query expansion terms for company c.

In addition to these three, stock volume is also used as a feature. These features and their variations form the basis for all the experiments in chapter five. To use these features, they first have to be transformed into suitable format.

4.6.2 Financial Data Interpolation

The stock chosen for evaluation are traded on either the NASAQ or the FTSE. Neither operate on weekends or bank holidays. Thus, there are gaps in the dataset for the stock volume. The gaps could remain, and the SVR could model only weekday data. Alternatively, the missing values could be linearly interpolated. This second option is used in this thesis, because it allows the SVR to capture potentially significant Twitter activity occurring on weekends.

4.6.3 Twitter Data Normalisation

There are more tweets towards to end of the dataset than at the start. This is displayed in figure 3.2. This means that features towards the end of the date range will have higher

[4]http://numpy.scipy.org/ [accessed: 28-AUG-2011]

values than those near the start. For example, let N_1 be the number of tweets on day 1 and let N_2 be the number of tweets on day 2. Let $N_1 = 100$, and $N_2 = 1000$. Let cf_1 be the number of tweets that mention company c on day 1 and let cf_2 be the number of tweets that mention company c on day 2. Let cf_1 be 11 and let cf_2 be 100. If only the raw cf values are used in the feature vector, it would appear that on day 2 there is more twitter activity for company c compared to day 1, as $cf_2 > cf_1$. However, this may be just because the crawler only collected a sample of tweets. To counteract this, the Twitter features are normalised. This is done by dividing cf_i by N_i. This approach is based on the term-frequency element of the term-frequency inverse-document-frequency weighting scheme (Manning et al., 2008).

4.6.4 Transforming Data

Scikits-learn requires that training data be represented as a vector of real numbers. Thus, the training data was in the following format, where x is a vector of features, and each feature vector is of length m, and y is a vector of responses (or labels), and n is the size of the training data:

$$x = ((f_{1,1}, f_{1,j+1}, \ldots, f_{1,m}), (f_{i,1}, f_{i,j+1}, \ldots, f_{i,m}), \ldots, (f_{n,1}, f_{n,j+1}, \ldots, f_{n,m}))$$
$$y = (l_1, l_i, \ldots, l_n)$$

4.6.5 Scaling

The feature vectors also require scaling. The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation, because the kernel values usually depend on the inner products of feature vectors, large values may cause numerical problems

(Hsu et al., 2010). Feature vectors are scaled into the range of $[-1{:}1]$ using scikits-learn.

4.6.6 Kernels

Having prepared the data, the next step is to select the SVM kernel. One of the main attributes of SVMs is the use of kernel functions to extend the class of decision functions to the non-linear case. This is done by mapping the data from the input space into a high dimensional feature space by a function. This is commonly referred to as "the kernel trick". There are four common kernels used in SVMs, here, γ, r, and d are the kernel parameters:

Linear: $K(x_i, x_j) = x_i^T x_j$

Radial Basis Function (RBF): $K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2), \gamma > 0$

Polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$

Sigmoid: $K(x_i, x_j) = tanh(\gamma x_i^T x_j + r)$

The Linear kernel is the simplest kernel function. It is given by the inner product (x, y) plus an optional constant C. An advantage of the linear kernel is that there is only one parameter to tune.

The Radial Basis Function (RBF) is equivalent to mapping the data into an infinite dimensional Hilbert space (Manning et al., 2008). In addition to the C paramour, the adjustable parameter σ plays a major role in the performance of the kernel. If overestimated, the kernel will behave almost linearly and the higher-dimensional projection will start to lose its non-linear power. If underestimated, the function will lack regularisation and the decision boundary will be highly sensitive to noise in training data.

The Sigmoid Kernel comes from the Neural Networks field, where the bipolar sigmoid function is often used as an activation function for artificial neurons. Besides the C parameter, there is an adjustable slope alpha parameter. A common value for alpha is 1/N, where N is the data dimension (Lin and Lin, 2003).

A polynomial kernel is used to model feature conjunctions (up to the order of the polynomial). The γ parameter decides the degree of polynomial and determines the flexibility of the resulting SVM in fitting the data (Manning et al., 2008).

4.6.7 Kernel Selection

The effectiveness of SVM depends on the selection of kernel, the kernel's parameters, and soft margin parameter C (Hsu et al., 2010). For the experiments, there are two options for the selection of the kernel and parameters: use the same kernel and parameters for all stocks, or optimise the kernel and parameters for each stock. Using the same kernel and parameters for all stocks was investigated, however the predictions made using standardised parameters were poor, on average an order of two worse than the 2SMA baseline. Thus, it was decided to optimise each individual stock. Each stock is trained using its own SVM and its own features, so it is logical that each SVM requires custom parameters.

Grid Search and Cross validation

To reduce complexity and computing time, the same kernel is used for each stock and just the parameters are optimised. Optimal parameters for SVMs can be found using grid-search and cross-validation. A grid search is the act of trying every possible combination of parameters in turn. Cross-validation is a way of measuring the predictive performance of a model by testing the model on data not used in training. By combining these two methods, optimal SVM parameters can be found.

K-fold cross validation is a method of cross validation that can be used for SVM parameter selection. In K fold cross validation, the training set is divided into k subsets. With each iteration i the SVM is trained on k-1 subsets, where the k_i is removed. The SVM is then tested on the i subset. The average error across all k trials is computed. An advantage of this method is that every data point in the dataset is tested exactly once and is used in training k-1 times. A disadvantage of this method is that the training algorithm has to be rerun from scratch k times.

10 fold cross validation in conjunction with grid search is used to find the most suitable kernel for the experiments. In the following tests, values $10^{-5}, 10^{-4}, \ldots, 10^{5}$ were tested for the C parameter. For non-linear kernels, values $10^{-5}, 10^{-4}, \ldots, 10^{5}$ were also tested for the γ parameter.

Test 1 - 30 days using volume

The first test uses a subset of the data set, 01 Jan - 30 Jan. A single feature is used: the previous day's volume. Thus, the SVM is attempting to predict the stock volume for day$_i$, using the stock volume for day$_{i-1}$. The training set for this test is in the following format, where x is a feature vector and y is the output / label vector.

$$x = ((vol_1), (vol_{i+1}), \ldots, (vol_{n-1}))$$

$$y = ((vol_2, vol_{i+2}, \ldots, vol_n))$$

The results of this initial experiment for the Linear, RBF, and Sigmoid kernels are displayed in table 4.5. The polynomial kernel was also tested, however, for unknown reasons the computation times for this kernel were considerably larger than other kernels. As a result, the polynomial kernel was ignored for this and future tests.

The test is inconclusive. The difference between the kernels is nominal. Thus, a second test was conducted.

Kernel	APPL	GOOG	VOD	BRBY
Linear	0.337	0.340	0.177	0.326
RBF	0.339	0.358	0.176	0.336
Sigmoid	0.335	0.358	0.177	0.328

Table 4.5: Kernel Test 1 - 30 day sample - using Volume feature

Kernel	APPL	GOOG	VOD	BRBY
Linear	0.335	0.323	0.176	0.286
RBF	0.339	0.357	0.176	0.341
Sigmoid	0.335	0.356	0.176	0.307

Table 4.6: Kernel Test 2 - 30 day sample - Volume and Moving Average

Test 2 - 30 days using volume and moving average

The second test is similar to the first, but an extra feature, the two day moving average, is added. The training data for this test is in the form:

$$x = ((vol_1, ma_1), (vol_{i+1}, ma_{i+1}), \ldots, (vol_{n-1}, ma_{n-1}))$$
$$y = ((vol_2, vol_{i+2}, \ldots, vol_n))$$

The results of this test are displayed in table 4.6.

Again, there is little separating the kernels. The linear kernel performed slightly better than the other two, but the difference is nominal. Thus, a third test is to be conducted.

Test 3 - 60 days using volume and moving average

The purpose of the third test is to see whether extending the sample training set makes any difference. The test duration is doubled to 60 days. The results of this test are displayed in table 4.7.

Again, the test shows that little separates the kernels. The linear kernel is chosen for

Kernel	APPL	GOOG	VOD	BRBY
Linear	0.304	0.265	0.329	0.751
RBF	0.306	0.278	0.331	0.662
Sigmoid	0.303	0.260	0.329	0.749

Table 4.7: Kernel Test 3 - 60 day sample - Volume and Moving Average

future experiments. There are two reasons for this choice. Firstly, the linear kernel performed nominally better than the other two in the first and second tests. Secondly, computationally the linear kernel is quicker, compared to this the other kernel. However, no statistics were recorded to prove this. The computation speed is important because of the choice of cross validation, discussed below.

Parameter Optimisation and Validation

In the above tests, each stock is trained and tested using every possible combination of parameters using 10 fold cross validation and the best results are presented. However, in the experiments in the following chapter the above method is unsuitable. Using parameters optimised to the actual output is unrealistic, because if attempting to predict tomorrow's stock volume the output value is unknown. The parameters should be optimised without knowledge of the final output.

Leave-one-out cross validation (LOOXV) is K-fold cross validation where K is equal to N, here N is number of data points in the dataset. This means that for every data point in the training set, the rest of the dataset is used for training and that single item is used for testing. The average error is computed and used to evaluate the model. This method has the benefit that the SVM is trained on as much of the data as possible. The downside is that it is expensive to compute.

The experiments presented in the following chapter use LOOXV to predict each value in the dataset. This ensures that each value is trained using as large a training set as possible.

To find the optimal parameters to train the model, 10 fold cross validation is used on the remaining training set, i.e., the dataset excluding the value to be predicted.

4.7 Experiment Setup

Before proceeding to the experiments, it is important to define the test conditions:

- All tests use the same dataset. Tests are conducted on the period 03 Jan - 31 May 2011. The SVM is trained on the period 03 Jan - 30 May and tested on the period 04 Jan - 31 May.

- There are no missing values in the training data. Missing financial data is linearly interpolated, as discussed above.

- Features extracted from Twitter are normalised. All features are scaled.

- Predictions are made using leave one out cross validation. The results are displayed using the MAPE and RMSE evaluation methods.

- The SVM uses a linear kernel. ϵ is set to 0.1 and the C parameter is optimised for each prediction using 10 fold cross validation. 10 fold cross validation is conducted using every value in the training set excluding the value to be predicted.

Each test follows the following steps:

1. Financial and Twitter data is loaded.
2. Selected features are constructed from the dataset.
3. Features are scaled.
4. Predictions are made for each day using leave one out cross validation.

Step four can be further expanded:

1. For i = 1 to N, where N is the size of the dataset.
2. Let (x_i, y_i) be the ith record.
3. Temporarily remove (x_i, y_i) from the data set
4. Use 10-fold cross validation on the data set to find the optimal parameters
5. Train SVM using the optimal parameters on the remaining N-1 data points.
6. Predict value p_i for d_i.
7. Repeat steps 2 through 5 for d_i, $i = 1, \ldots, n$
8. Compute MAPE and RMSE from $p_i, \ldots, p_n$

4.8 Chapter Summary

This chapter explained the theory behind the experiments presented in the following chapter. An evaluation method and a baseline were defined. In addition, the test conditions were clearly defined.

Chapter 5

Results and Discussion

This chapter documents the experiments conducted as part of this thesis. After establishing a SVM baseline, a series of tests are introduced, conducted, and their results are analysed. The purpose of these tests is to establish whether Twitter activity can predict stock volume a day in advance. The chapter concludes with analysis of and investigation into the results.

5.1 Tests

5.1.1 Test 1 - SVM Baseline Using Previous Day's Volume

The first step in the experiments is to establish a baseline using the SVM. Although a two day moving average baseline (2MA) was established in the previous chapter, it is beneficial to establish a baseline using the SVM for numerous reasons. Fundamentally, it demonstrates that the SVM is performing as expected and that the test methodology is competent. In addition, it establishes the potential performance of the SVM. Finally, it adds another benchmark to compare the results of later experiments.

To establish this second baseline, the preceding day's stock volume (vol) is used as a single feature. Thus, the SVM is attempting to predict the stock volume for day$_i$, using the

	APPL	GOOG	VOD	BRBY
Prediction				
MAPE RMSE	0.176 5601884	0.177 1158401	0.236 27535083	1.015 10477476
2MA baseline				
MAPE RMSE	0.227 6497221	0.231 1381626	0.241 30604842	2.030 12374632

Table 5.1: Test 1 - Prediction using preceding day's volume

stock volume for day_{i1}. The training set for this test is in the following format, where x is a feature vector and y is the label vector.

$$x = ((vol_1), (vol_{i+1}), \ldots, (vol_{n-1}))$$

$$y = ((vol_2, vol_{i+2}, \ldots, vol_n))$$

The results of test 1 are displayed in table 5.1. The results are encouraging. The predictions made for each stock improve on the 2MA baseline. Although still relatively poor, the prediction for BRBY is a considerable improvement on the 2MA baseline.

Figures 5.1, 5.2 and 5.3, show line charts displaying the predictions for APPL, GOOD, and VOD respectively. The predictions for both APPL and GOOG are particularly impressive. The line chart displaying the predictions for VOD is less impressive. The predictions for BRBY, displayed in figure 5.4, are, as expected, dominated by the volume spike.

Overall, the results of this test show that there is potential to predict the following day's stock volume and that the SVM is capable of performing this prediction.

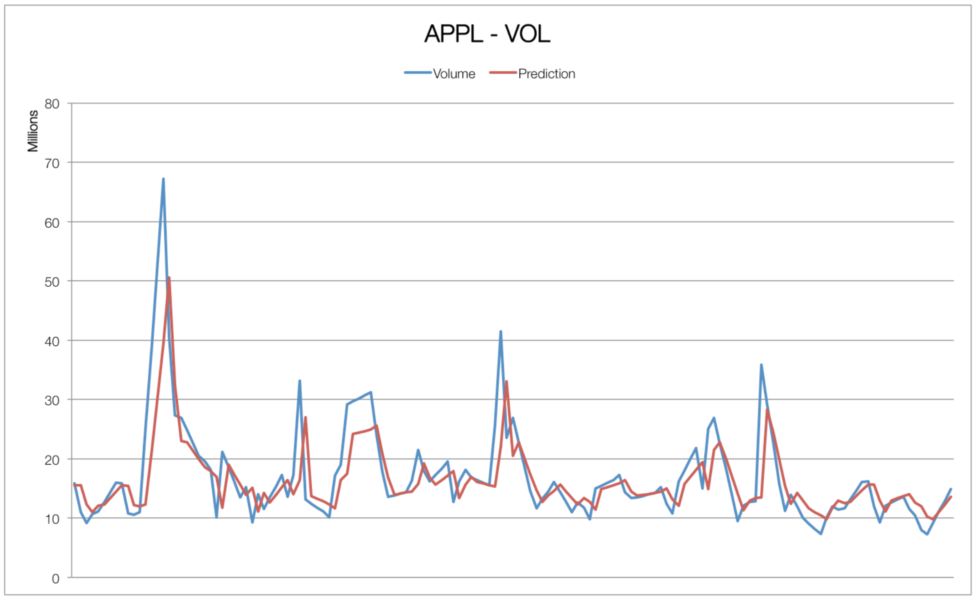


Figure 5.1: Test 1 - APPL VOL

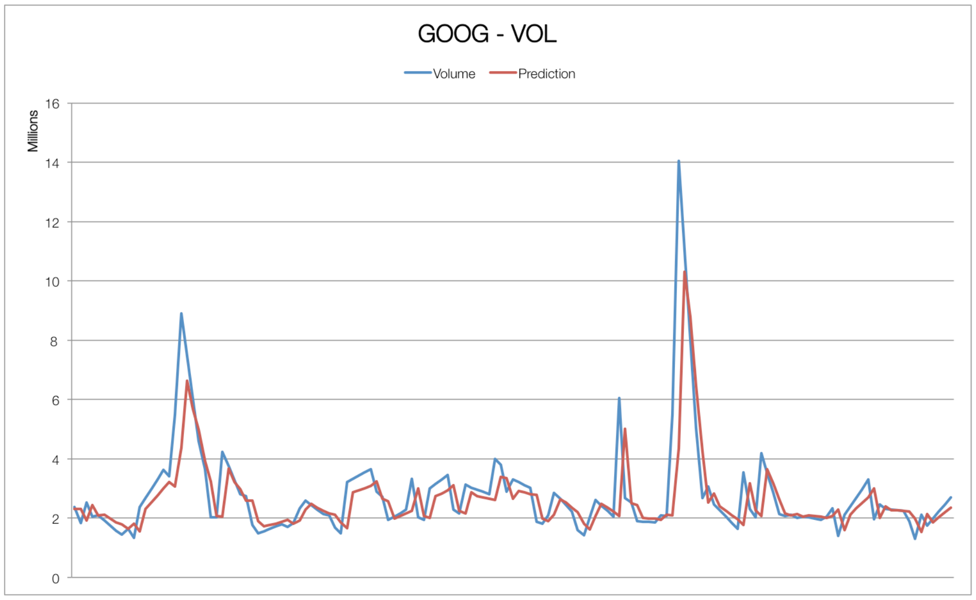


Figure 5.2: Test 1 - GOOG VOL

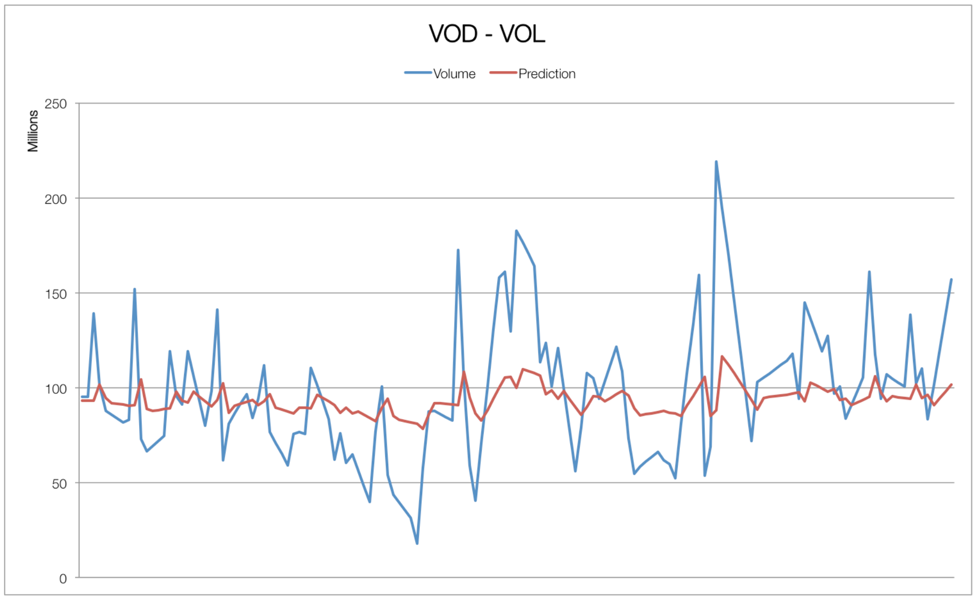


Figure 5.3: Test 1 - VOD VOL

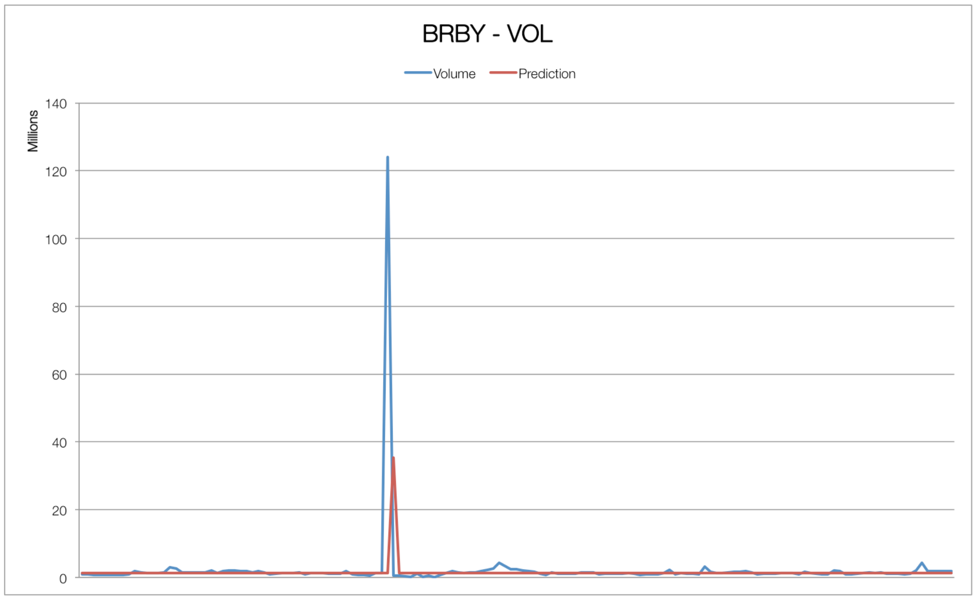


Figure 5.4: Test 1 - BRBY VOL

5.1.2 Test 2 - Word Count, Hash Count, Query Score

After establishing the SVM volume baseline in test 1, the purpose of the second test is to investigate whether Twitter activity alone can predict stock volume.

There are three tests to be conducted. The first uses the single feature WC, which is the number of tweets on day_i that mention a company name. The second test is similar but uses the feature HC, which is the number of tweets on day_i that mentioned a company in a hash tag. The final test uses the single feature QS, which is the number of tweets on day_i that mentioned any of the words in the expanded query for a company. The training set for the test is of the following form, where f is one of the before mentioned features.

$$
\begin{aligned}
x &= ((f_1), (f_{i+1}), \ldots, (f_{n-1})) \\
y &= ((vol_2, vol_{i+2}, \ldots, vol_n))
\end{aligned}
$$

The results of this test are displayed in table 5.2. The results are disappointing. In terms of MAPE and RMSE, the predictions are less accurate than both the 2MA baseline and the SVM volume baseline established in test 1, except for the BRBY prediction, which beat the 2MA baseline. The results are particularity discouraging when viewed on a line chart. Figure 5.5 displays the predictions of each test for APPL. The SVM roughly makes the same prediction every day, regardless of the different feature values. The results are frustrating because if the raw Twitter activity for APPL is viewed on a similar line chart, it is not a straight line and there is much variation throughout the time period. Figure 5.6 displays this. The same applies to line charts for GOOG, VOD, and BRBY. These charts are omitted for brevity.

Overall, this test shows that single Twitter features alone are incapable of predicting the following day's stock volume. Consequently, subsequent tests focus on two things: combining

	APPL	GOOG	VOD	BRBY
WC:				
MAPE RMSE	0.274 8642021	0.260 1742235	0.331 34654733	0.598 10083778
HC:				
MAPE RMSE	0.264 8649416	0.254 1717906	0.301 33605387	0.597 10083769
QS:				
MAPE RMSE	0.274 8712307	0.265 1739501	0.326 34435465	0.598 10083812
VOL baseline:				
MAPE RMSE	0.176 5601884	0.177 1158401	0.236 27535083	1.015 10477476
2MA baselne:				
MAPE RMSE	0.227 6497221	0.231 1381626	0.241 30604842	2.030 12374632

Table 5.2: Test 2 - Word Count, Hash Count, and Query Score

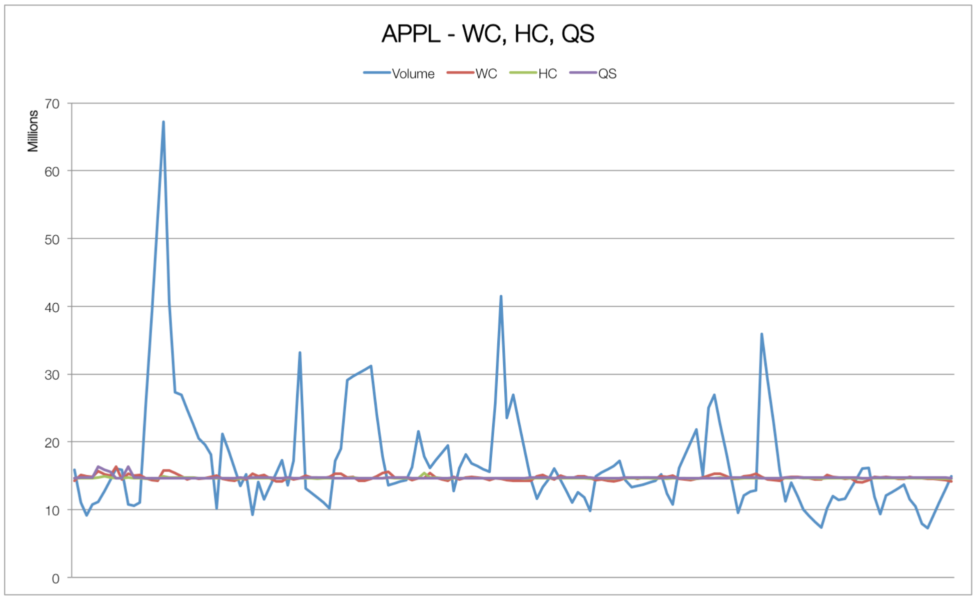


Figure 5.5: Test 2 - Word Count, Hash Count, Query Score

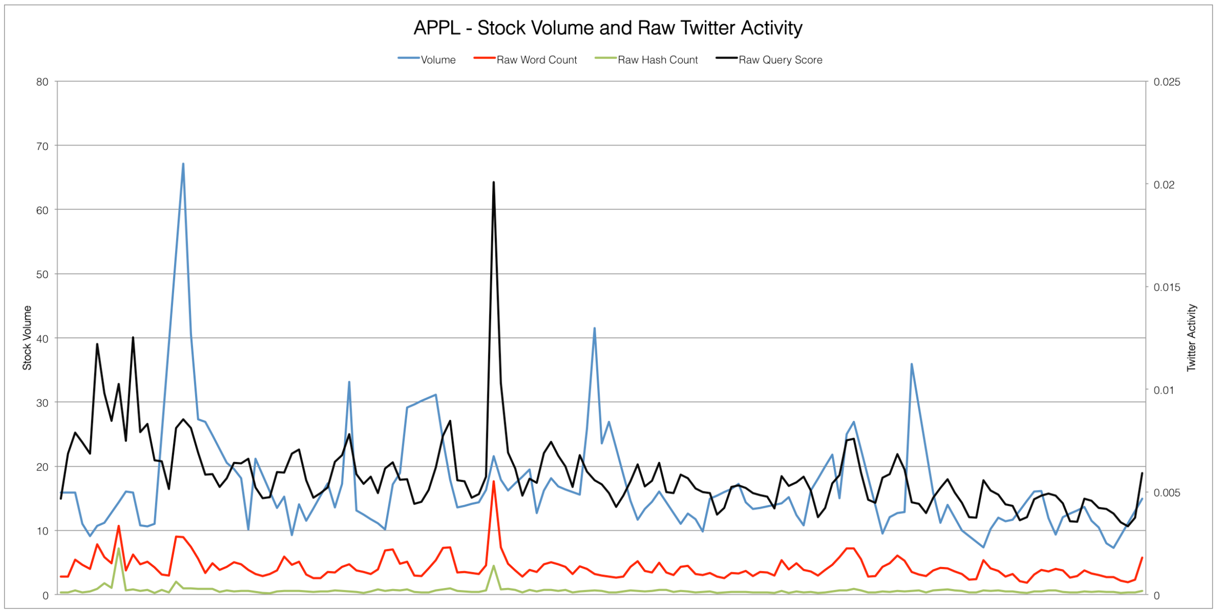


Figure 5.6: APPL Stock volume and Raw Twitter Activity

Twitter features with each other, and combining Twitter features with stock features, such as previous day's volumes.

5.1.3 Test 3 - Word Count and Hash Count and Query Score Combined

Following the poor results obtained in test 2, the purpose of test 3 is to see whether combining the Twitter improves the predictive performance. In this test, the WC, HC, and QS features are all used together. Thus, the training set for this test is as followed:

$$x = ((wc_1, hc_i, qs_i), (wc_{i+1}, hc_{i+1}, qs_{i+1}), \ldots, (wc_{n-1}, hc_{n-1}, qs_{n-1}))$$
$$y = ((vol_2, vol_{i+1}, \ldots, vol_n))$$

The results for test 3 are displayed in table 5.3. Again, the results were disappointing.

	APPL	GOOG	VOD	BRBY
Prediction:				
MAPE RMSE	0.275 8710121	0.254 1717906	0.319 33827061	0.598 10083823
VOL baseline:				
MAPE RMSE	0.176 5601884	0.177 1158401	0.236 27535083	1.015 10477476
2MA baseline:				
MAPE RMSE	0.227 6497221	0.231 1381626	0.241 30604842	2.030 12374632

Table 5.3: Test 3 - Word Count, Hash Count, and Query Score Combined

The difference between the results in test 2 and test 3 are small.

Test 3 shows that combining the three Twitter features has negligible effect on performance. Both tests 2 and 3 show that there is little, if any, potential for predicting stock volume using Twitter features alone. Thus, subsequent tests focus on combining the Twitter features with features embedded in the stock, namely previous stock volumes.

5.1.4 Test 4 - Volume and Twitter Features

In tests 2 and 3, single features extracted from Twitter are used, i.e., no knowledge about stock volume was included in the training model. The purpose of this test is to see whether the SVM baseline established in test 1 can be improved on by adding Twitter features. There are three tests. In each one, volume is combined with one of WC, HC, or QS. Thus, the training set was in the following form, where f is one of WC, HC, or QS.

$$x = ((vol_1, f_1), (vol_{i+1}, f_{i+1}), \ldots, (vol_{n-1}, f_{n-1}))$$
$$y = ((vol_2, vol_{i+2}, \ldots, vol_n))$$

	APPL	GOOG	VOD	BRBY
MAPE RMSE	0.174 5581249	0.178 1144768	0.228 27079416	1.026 10497190
MAPE RMSE	0.179 5622453	0.175 1138790	0.227 26768131	1.025 10496383
MAPE RMSE	0.176 5587158	0.172 1151304	0.211 26534799	1.026 10496950
VOL baseline:				
MAPE RMSE	0.176 5601884	0.177 1158401	0.236 27535083	1.015 10477476
2MA baseline:				
MAPE RMSE	0.227 6497221	0.231 1381626	0.241 30604842	2.030 12374632

Table 5.4: Test 4 - Volume and Count, Hash Count, and Query Score

The results of test 4 are displayed in table 5.4. Again, the results are disappointing. Combining Twitter features with stock volume makes little difference in terms of predictive accuracy. Adding the WC feature improved the accuracy for APPL and VOD, but worsened the accuracy for GOOD and BRBY. Adding the HC and QS improved the accuracy for GOOG and VOD, but worsened it for APPL and BRBY.

5.1.5 Test 5 - Past Values

It is clear that Twitter activity on the day preceding the day to be predicted has little if any potential to predict stock volume. The purpose of test 5 is to explore whether including previous day's features could aid prediction performance. The theory behind this test is to include more of the time-series element of the data. For this test, the previous $j + 1$ days features are included. All three Twitter features, WC, HC, and QS are used. In addition, the previous day's volumes are also included in the feature vector. Because this test uses previous day's features, the size of the training set needs to be adjusted, because d_1 has no preceding days. Thus, the test starts at d_j. The specifics of the training data are omitted

	APPL	GOOG	VOD	BRBY
VOL baseline:				
MAPE RMSE	0.176 5601884	0.177 1158401	0.236 27535083	1.015 10477476
j = 1				
MAPE RMSE	0.178 5628102	0.169 1121443	0.230 27074797	0.597 10083722
j = 2				
MAPE RMSE	0.176 5687635	0.173 1136699	0.241 27997071	0.609 10152010
j = 3				
MAPE RMSE	0.180 5701694	0.175 1138317	0.241 28354901	0.609 10152010
2MA baseline:				
MAPE RMSE	0.227 6497221	0.231 1381626	0.241 30604842	2.030 12374632

Table 5.5: Test 5 - Previous Features

for brevity. The results for test 5 are displayed in table 5.5.

The results show that including the previous day's features generally improves performance. The best performance is obtained when j = 1, that is when the previous two day's features are included. This improves the predictive performance for GOOG, VOD, and significantly so for BRBY, but performance decreases for APPL. Although this test shows an improvement in predictive accuracy, the improvements are negligible, with the exception of BRBY, where the accuracy improved considerably.

5.1.6 Test 6 - Feature Changes

By now, it is clear that beating the SVM baseline obtained in test 1 is difficult. In this test, a slightly different approach is taken. In test 5, the preceding day's metrics were included in the training set, i.e., the previous day's stock volumes and previous day's Twitter metrics were included in the feature vector. This test aims to continue with the idea of embedding time-series information into the feature vector, however, the purpose of this test is to capture

	APPL	GOOG	VOD	BRBY
VOL baseline:				
MAPE RMSE	0.176 5601884	0.177 1158401	0.236 27535083	1.015 10477476
j = 1				
MAPE RMSE	0.174 5699450	0.171 1129461	0.248 28318410	1.003 10496661
j = 2				
MAPE RMSE	0.177 5711873	0.172 1160732	0.240 28096340	1.014 10515861
j = 3				
MAPE RMSE	0.177 5623564	0.179 1170237	0.233 27527911	1.014 10515249
2MA baseline:				
MAPE RMSE	0.227 6497221	0.231 1381626	0.241 30604842	2.030 12374632

Table 5.6: Test 6 - Volume previous volume changes

the notion of trends. The aim is to enable the SVM to identify trends in Twitter metrics and stock volume. For this test, the previous $j+1$ days features are included. However, in this test the features are the changes in metrics from one day to another. Using volume as an example, the feature vector would consist of:

$$x = (vol_2 - vol_1), (vol_{2+i} - vol_{2+i-1-j}), \ldots, (vol_{n-1} - vol_{n-1-j})$$

All three Twitter metrics, WC, HC, and QS are used in this test, in addition to volume and previous $j+1$ day's volumes. Again, since this test uses previous day's metrics, the size of the training set is adjusted accordingly. The results of this test are displayed in table 5.6.

The results show that there is little change from the results obtained in test 5.

	APPL	GOOG	VOD	BRBY
Prediction:				
MAPE RMSE	0.174 5672886	0.176 1128379	0.238 27258805	0.602 10151631
VOL baseline:				
MAPE RMSE	0.176 5601884	0.177 1158401	0.236 27535083	1.015 10477476
2MA baseline:				
MAPE RMSE	0.227 6497221	0.231 1381626	0.241 30604842	2.030 12374632

Table 5.7: Test 7 - 24 Features

5.1.7 Test 7 - All But The Kitchen Sink

By now, it is evident that is it difficult to significantly beat the baseline obtained in test 1. As a final test, all the training features of previous tests are included. In terms of previous day's metrics, the previous three days values are included. The purpose of this test is to see whether training the SMV with substantially more features improves performance. In total, there are 24 features used. The following lists all the features used, where d is the day preceding the day to be predicted:

- VOL d$_i$, VOL d$_{i-1}$, VOL d$_{i-2}$
- WC d$_i$, WC d$_{i-1}$, WC d$_{i-2}$
- HC d$_i$, HC d$_{i-1}$, HC d$_{i-2}$
- QS d$_i$, QS d$_{i-1}$, QS d$_{i-2}$
- VOL Change d$_i$, VOL Change d$_{i-1}$, VOL Change d$_{i-2}$
- WC Change d$_i$, WC Change d$_{i-1}$, WC Change d$_{i-2}$
- HC Change d$_i$, HC Change d$_{i-1}$, HC Change d$_{i-2}$
- QS Change d$_i$, QS Change d$_{i-1}$, QS Change d$_{i-2}$

The results are displayed in table 5.7.

No improvements are made on the baseline. The only difference is that the SVM took longer to train.

5.2 Results Investigation

After obtaining a strong baseline in test 1, subsequent tests failed to improve on performance. The purpose of this section is to investigate the reasons.

5.2.1 SVM C Parameter

The first focus of investigation is the SVM and specifically the choice of kernel and C parameter. To investigate this, a simple test is to be conducted that focuses solely on the C parameter. The SVM is to be trained to learn the identity function, i.e., it will be trained on a training set where each training instance (x, y), x and y are the same value. The training data is in the following form:

$$x = ((vol_1), (vol_{i+1}), \ldots, (vol_{n-1}))$$
$$y = ((vol_1), (vol_{i+1}), \ldots, (vol_{n-1}))$$

The results of this test are displayed in table 5.8.

The results show that C parameter has an absolute effect on performance. With a C ¡ 10^5, the SVM fails to learn the identity function for any of the stocks and the MAPE and RMSE scores for all stock are high, generally higher than results obtained in tests 1 - 7. When C = 10^5, the SVM learns the identity function for APPL and GOOG, but not VOD or BRBY. In addition, although not quantitatively measured, with each increment to the power of C, the computation time increases. When training the SVM with C = 10^5 for APPL and GOOG

	APPL	GOOG	VOD	BRBY
C = 10^{-3} :				
MAPE	0.269	0.253	0.337	0.600
RMSE	8769711	1731537	34925604	10151909
C = 10^{-2} :				
MAPE	0.269	0.253	0.337	0.600
RMSE	8769711	1731537	34925604	10151909
C = 10^{-1} :				
MAPE	0.269	0.253	0.337	0.600
RMSE	8769711	1731537	34925604	10151909
C = 10^{-0} :				
MAPE	0.269	0.253	0.337	0.600
RMSE	8769711	1731537	34925604	10151909
C = 10^{1} :				
MAPE	0.269	0.253	0.337	0.600
RMSE	8769619	1731460	34925493	10151903
C = 10^{2} :				
P MAPE	0.269	0.253	0.337	0.600
P RMSE	87696198791	1730767	34924494	10151846
C = 10^{3} :				
MAPE	0.269	0.252	0.337	0.600
RMSE	8760511	1723836	34914495	10151277
C = 10^{4} :				
MAPE	0.266	0.241	0.336	0.599
RMSE	8677714	1654523	34814507	10145584
C = 10^{4} :				
MAPE	0.00	0.000	0.327	0.589
RMSE	7	9	33814634	10088662

Table 5.8: Identify Function Test

the computation time took significantly longer than in any previous tests. Whereas the tests 1 - 7 took on average less than five minutes to compute, training the SVM when C = 10^5 just for APPL took over 20 minutes. The results of this test show that, using the training set used, SVMS are highly temperamental to the kernel parameters. This agrees with Kim (2003), who highlighted that the prediction performances of SVMs are highly sensitive to the value of upper bound C and the kernel parameters.

5.2.2 Is There Any Correlation At All?

According to tests 1 - 7, Twitter activity is not a strong feature for predicting stock volume a day in advance using SVR. This section investigates whether there is any correlation at all between Twitter activity and stock volume, i.e., is there any potential at all.

Another test is to be conducted. The purpose is to see whether Twitter activity on day_i can predict stock volume on day_i. Three test are to be conducted: one using WC, one using HC, and the final one using QS. Thus, the training set is of the following form, where f is one of the preceding features.

$$x = ((f_i), (f_{i+1}), \ldots, (f_{n-1}))$$
$$y = ((vol_i, vol_i, \ldots, vol_{n-1}))$$

Table 5.9 displays the results. Again, the results were poor and confirmed the findings in tests 1 - 7.

Note that, in addition to all the previous tests in this chapter, the tests were repeated again using non-interpolated financial data, to see whether that made a difference. However, the results were the same – the results are omitted for brevity.

	APPL	GOOG	VOD	BRBY
WC:				
MAPE RMSE	0.269 8769588	0.253 1731574	0.337 34925564	0.600 10151909
HC:				
MAPE RMSE	0.269 8769707	0.253 1731548	0.337 34925566	0.600 10151909
QS:				
MAPE RMSE	0.269 8769649	0.253 1731569	0.337 34925558	0.600 10151909

Table 5.9: Model Twitter Activity with same day volume

5.3 Results Analysis

The results obtained in the tests presented in this chapter suggest that it is not possible to predict the following day's stock volume using features extracted from Twitter using SVR. This section investigates this.

The first step is to visualise the relationship between Twitter activity and stock volume. Figures 5.7, 5.8, 5.9, and 5.10 display line charts showing the relationship between stock volume and Twitter activity for APPL, GOOG, VOD, and BRBY respectively. The Twitter activity displayed in the charts is the word count feature, i.e., the number of tweets mentioning the company name.

By viewing the graphs, there appears to be a disparity between the different stocks. Visually, the relationship between Twitter activity and stock volume for APPL is stronger than the other stocks. To investigate this further, the stocks were plotted on a scatter graph. Figures 5.11, 5.12, 5.13 and 5.14 display scatter graphs for APPL, GOOG, VOD, and BRBY respectively.

By viewing the scatter graphs, there appears to be no relationship between the values. To investigate this further, i.e., beyond just looking, the Pearson correlation coefficient is used to measures the linear relationship between two datasets.

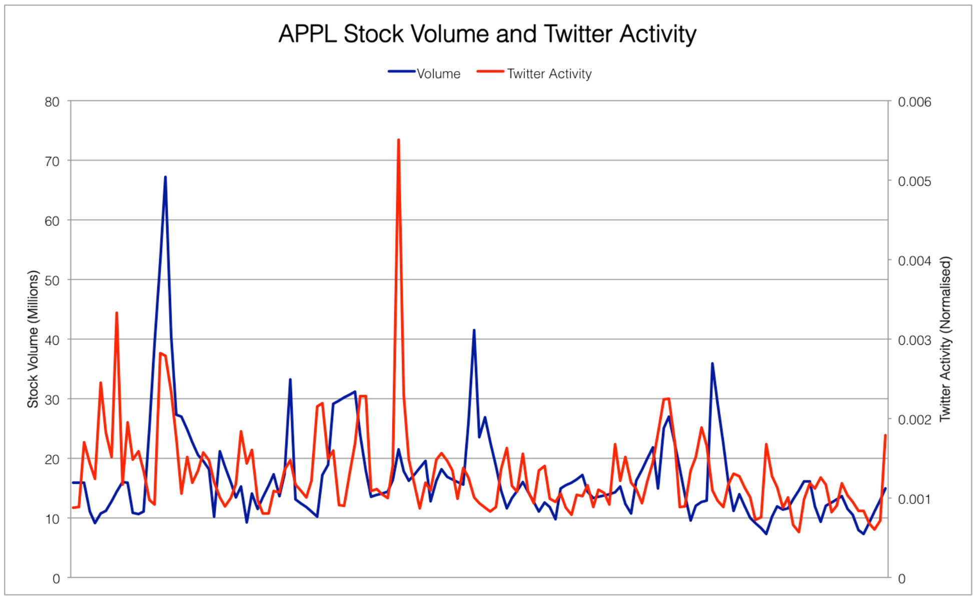


Figure 5.7: APPL Stock volume and Raw Twitter Activity

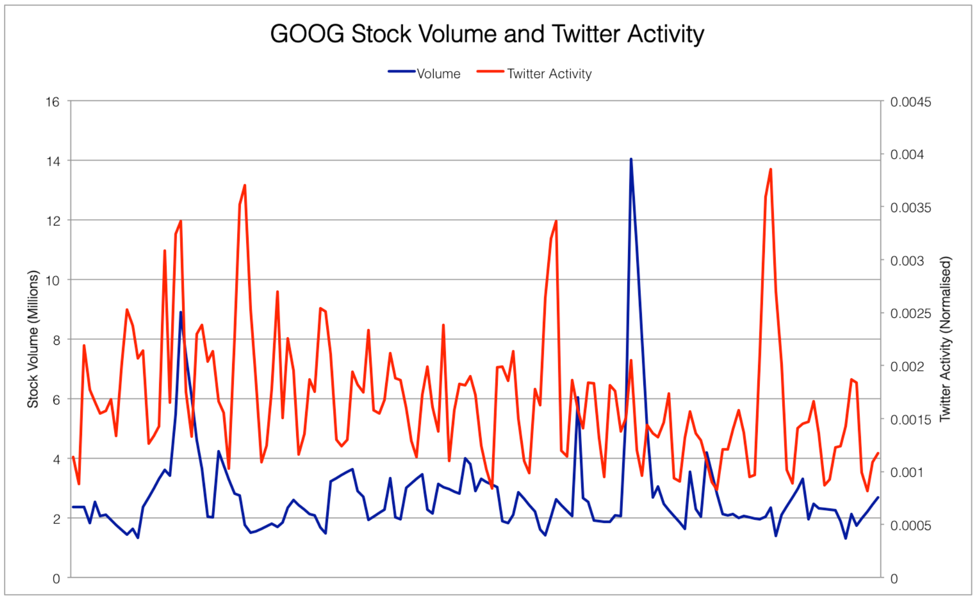


Figure 5.8: GOOG Stock volume and Raw Twitter Activity

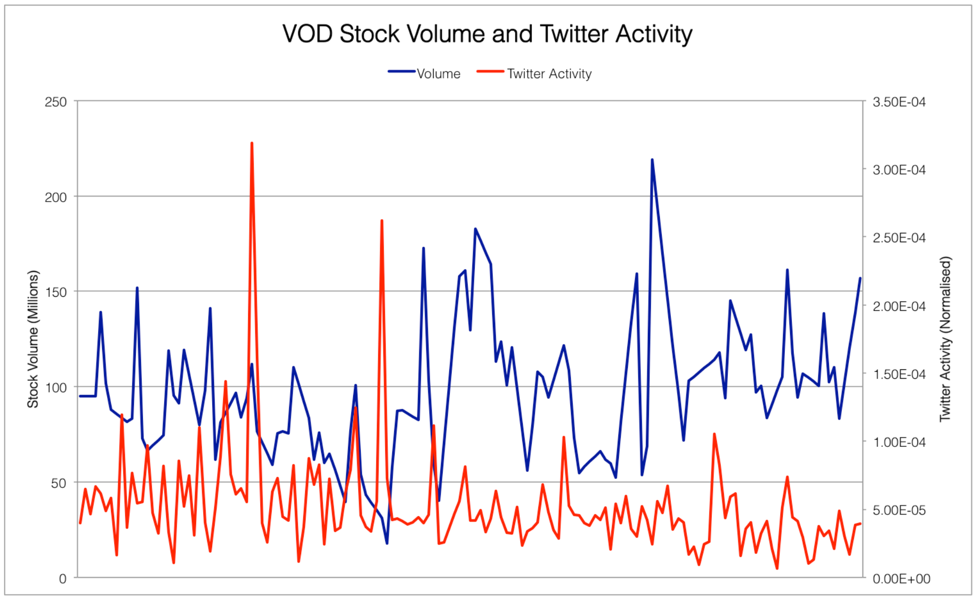


Figure 5.9: VOD Stock volume and Raw Twitter Activity

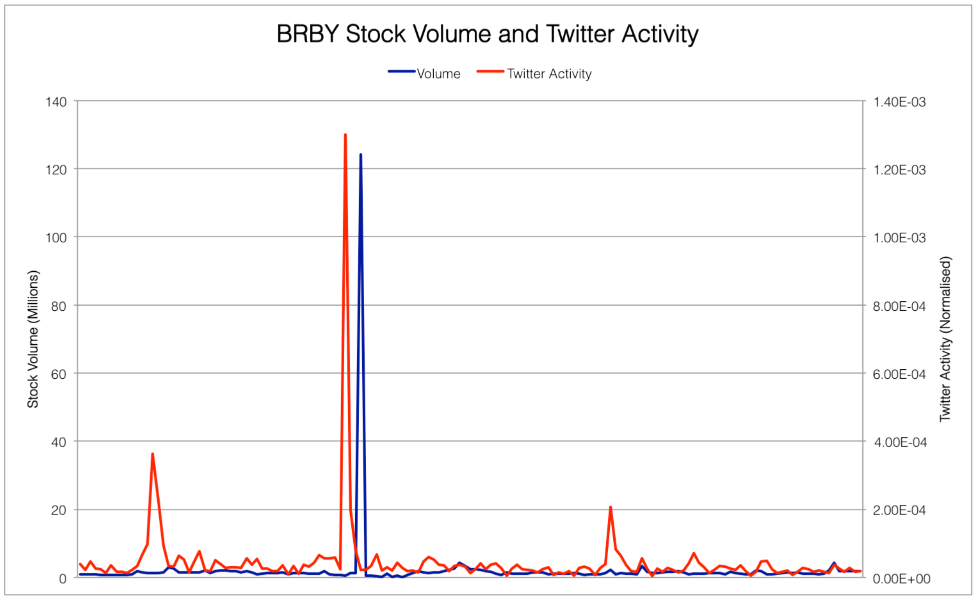


Figure 5.10: BRBY Stock volume and Raw Twitter Activity

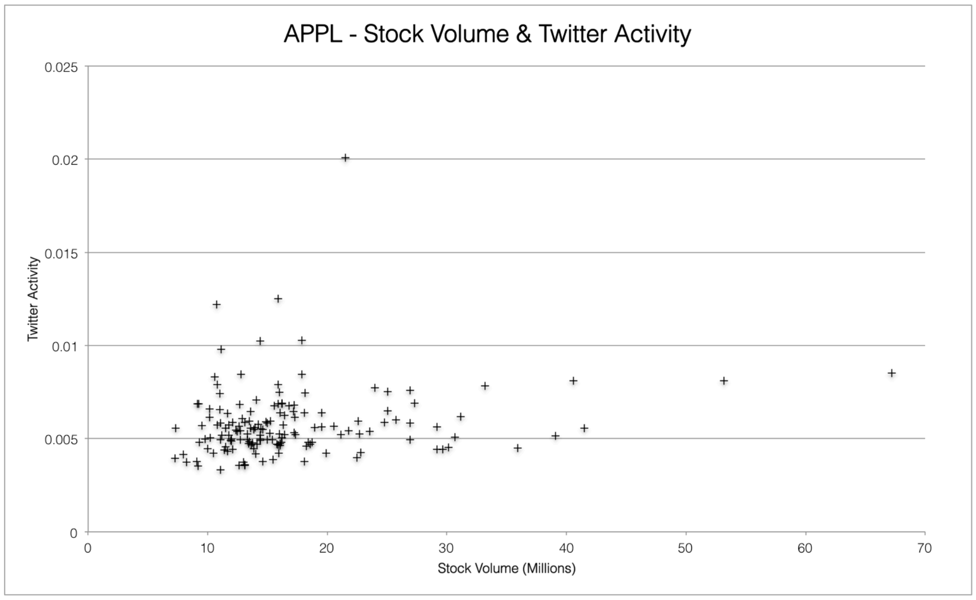


Figure 5.11: APPL Stock volume and Raw Twitter Activity

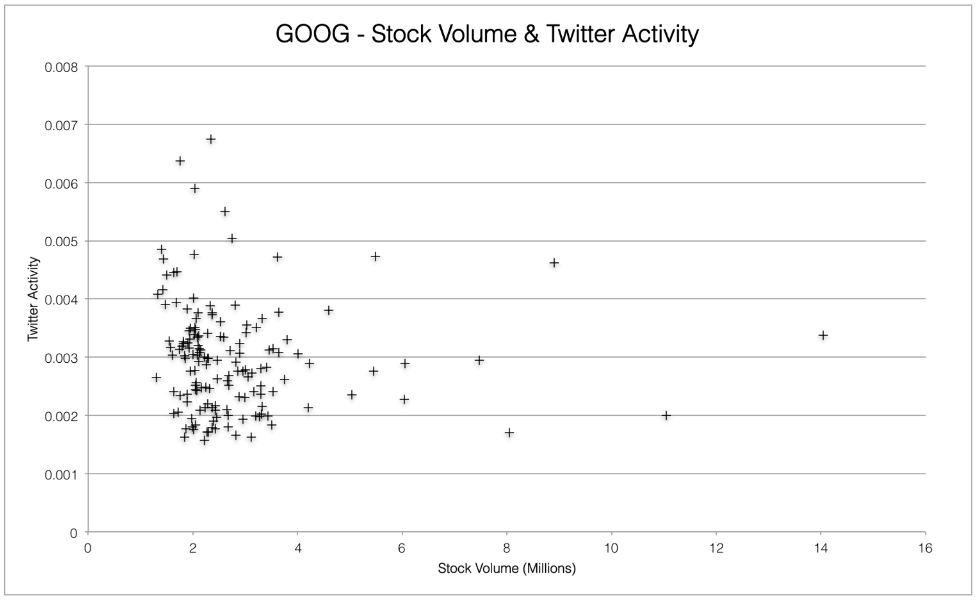


Figure 5.12: GOOG Stock volume and Raw Twitter Activity

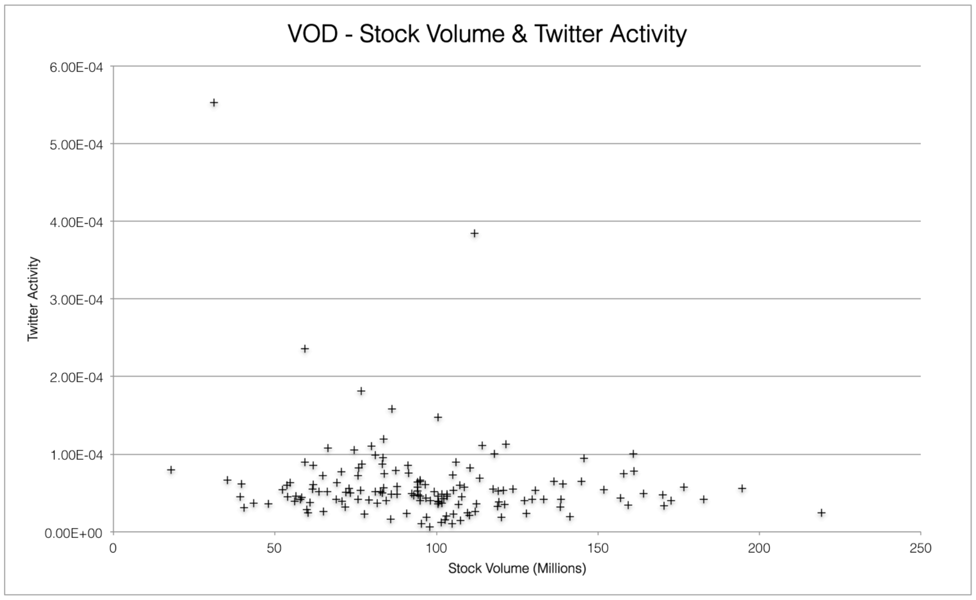


Figure 5.13: VOD Stock volume and Raw Twitter Activity

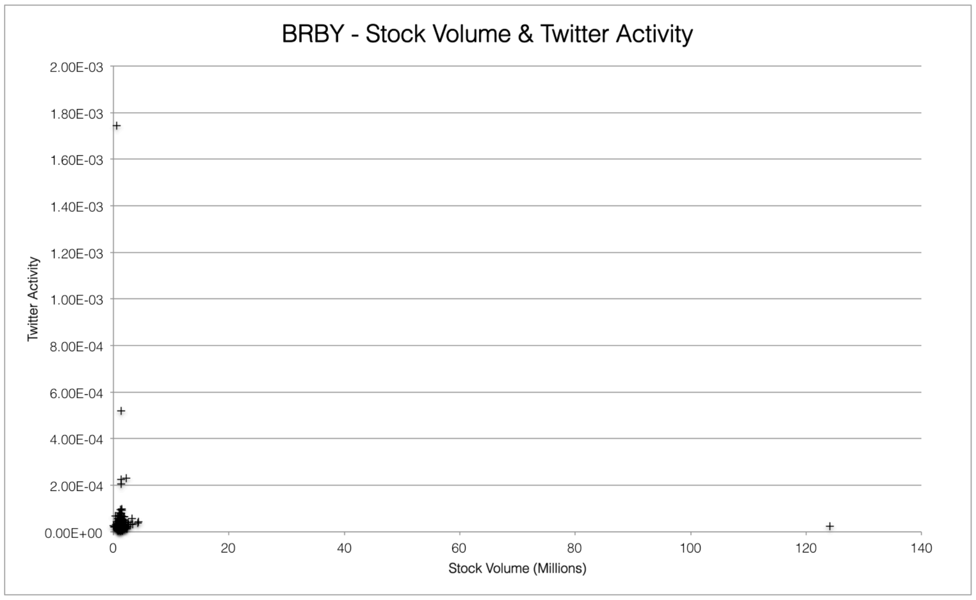


Figure 5.14: BRBY Stock volume and Raw Twitter Activity

APPL	GOOG	VOD	BRBY
0.3015	0.058	-0.142	-0.026

Table 5.10: Pearson Coefficients

The Pearson correlation coefficient measures the linear relationship between two datasets. The coefficient varies between 1 and +1 with 0 implying no correlation. Correlations of 1 or +1 imply an exact linear relationship. Positive correlations imply that as x increases, so does y. Negative correlations imply that as x increases, y decreases.

The Pearson correlation coefficients for the stocks are displayed in table 5.10.

The Pearson Coefficients confirm what the scatter plots suggested visually, that is that there is no strong correlation. APPL has the highest correlation, which agrees with what could be visually inferred on the line charts. However, none of the coefficients are high. Thus, it is understandable why the SVM struggles to create a hyperplane that achieves good accuracy. This, however, does not mean that Twitter cannot help to predict stock volume. This is elaborated further in chapter six.

5.4 Chapter Summary

This chapter presented several experiments to investigate whether it is possible to model and predict stock volume using Twitter. The first test established a strong SVM baseline using volume alone. However, subsequent tests failed to improve on this baseline. In general, the results in this chapter suggest that is not possible to use Twitter and SVR to model and predict stock volume.

Chapter 6

Conclusions and Further Work

This thesis investigated whether Twitter could be used to model and predict stock volume. A Twitter dataset of 60 million tweets posted by 500k users was constructed. After filtering, this was reduced to 40 million tweets covering the period 01 Jan - 31 May 2011. Support vector regression was used to model and predict stock volume. To test the model, four stocks were investigated: Apple (APPL), Google (GOOG), Vodafone (VOD), and Burberry (BRBY). Several experiments were initially conducted, with further experiments used in the analysis stage. A two day moving average was used as the baseline. Results were evaluated using the mean absolute percentage error and root mean squared error methods. Leave one out cross validation was used for the experiments, with 10 fold cross validation used to optimise the SVM parameters. The experiments focused on three features extracted from Twitter: word count, hash count, and query score. Each of these corresponded to the number of tweets mentioning a company, or a related term, on a day.

The first test established a SVM baseline using the previous day's stock volume. This baseline comfortably beat the two day moving average baseline for every stock. The next tests focused on whether features extracted from Twitter alone could model stock volume. The results were conclusive: Twitter features alone are insufficient to predict stock volume.

No test beat the SVM nor the two day moving average baselines. Subsequent tests focused on various combinations of features, including historical volumes. Although some experiments showed improvements, notably when adding past values to the feature vector, the improvements were negligible.

Overall, the results suggest that there is little potential to predict stock volume a day in advance using SVR and the features used in this thesis.

6.1 Limitations and Future Work

It is understood that no previous work has attempted to model and predict stock volume using Twitter. Therefore, the findings of this thesis can be considered exploratory.

For future work, one area to explore is the prediction period. This thesis focused on next day prediction. Future work could consider shorter intervals. Wolfram (2010), found that Twitter could aid stock price predictions of up to 15 minutes; above this the accuracy decreased. The theory behind predicting shorter intervals reflects the idea that Twitter responds to new information quicker than the stock market.

Alternatives to SVR could also be investigated. When training the SVM, equal precedence is given to all training instances. Past events are of equal relevance to recent events. In future work, it may be beneficial to weight the training data, as to give precedence to recent events. This was considered during the thesis. Indeed, in the early stages of this thesis, time series analysis was investigated in depth, specifically vector autoregression and autoregressive moving average models. A benefit of these models is that they model the temporal aspect of the data. Eventually, however, the SVR approach was chose due to its usage in previous works, such as (Wolfram, 2010), and (Yi, 2009), and also due to the knowledge gained in the literature, particularly the efficient market hypothesis.

The main hypothesis of the thesis is that Twitter activity corresponds to stock volume.

This work suggests that, in general, this is false. However, this is not absolute, and Twitter still has potential to predict stock volume.

This work relied on purely quantitive Twitter features based on the number of tweets. This limits what can be captured from Twitter. Future work could consider different features. For example, consider the scenario of APPL releasing an update for the IPhone. People are likely to tweet about this, causing an increase in activity. However, the impact of this update on the stock market is likely to be minimal, the event can be considered insignificant. Now, consider the scenario presented in the introduction, where Steve Jobs resigns as the CEO of APPL. This resulted in an increase in Twitter activity, with APPL trending in the top four out of five Twitter topics(Mogg, 2011), (Martinelli, 2011). In addition, APPL's stock volume increased by 50%. By only counting tweets, there is no way to disambiguate between these events. For future work, it would be beneficial to investigate the significance of Twitter activity.

There are several approaches to this. One is to perform lexical analysis on the tweets. This may be useful to identify particular words in tweets that have some correlation to changes in the stock market, similar to the work of Wthrich et al. (1998), who used priori domain knowledge to aid predictions.

Another approach it to focus on Twitter users. For example, consider users Alice and Bob. Let Alice be an APPL fanatic, who frequently tweets about APPL. Let Bob be Alice's opposite, i.e., has little interest in APPL, and infrequently tweets about them. If Alice's tweets about APPL, it can be considered 'normal'. However, if Bob tweets about APPL, it would suggest something 'abnormal' and therefore possibly of increased significance. By capturing this on a mass scale, it may be possible to identify significant events, which could be used to aid prediction.

One final suggestion for future work is to measure user influence in Twitter, and then use this in modelling. This, however, is more complex that simply counting the number of followers that a user has, as the work of Cha et al. (2010) notes. One potential benefit of this is to capture the herd behaviour of some investors. Herd behaviour, also known as lemmings affect, describes how individuals in a group can act together without planned direction. Shiller (2001) accounts the cause of the the Dot-com bubble to this lemmings behaviour.

6.1.1 Problems Encountered

The goal was to use several more features extracted from Twitter during the experiments, such as unique users tweeting etc. However, several problems during the thesis prevented this. The construction of the datasets took considerably longer than expected. Although the implementation of the crawler was trivial, the running of the crawler was problematic. Several methods for storing the Tweets were tested before eventually deciding upon MongoDB. This step, however, was relatively straightforward compared to the pre-processing stage. This stage was problematic and time consuming. The computation time required to parse the dataset exceeded expectations. In addition, it took several development iterations until an efficient method of processing and storage was established. Even then, it took several days to process the dataset. In hindsight, the thesis either should have focused on a smaller date range, e.g. 30 days instead of 150, or used better hardware to compute the data quicker.

6.2 Concluding Remarks

This thesis investigated whether Twitter could be used to predict next day stock volume. It is understood that no previous has attempted to model and predict stock volume using Twitter. This work failed to predict next day stock volume. This, however, does not mean that it is impossible, and there is strong potential for future work.

Bibliography

Sitaram Asur, Bernardo A. Huberman, Gábor Szabó, and Chunyan Wang. Trends in social media : Persistence and decay. *CoRR*, abs/1102.1402, 2011.

R Ball and P Brown. An empirical evaluation of accounting income numbers. *Journal of Accounting Research*, 6(2):159–178, 1968.

Victor L Bernard and Jacob K Thomas. Evidence that stock prices do not fully reflect the implications of current earnings for future earnings. *Journal of Accounting and Economics*, 13(4):305–340, 1990.

J. Bollen, H. Mao, and X.-J. Zeng. Twitter mood predicts the stock market. *ArXiv e-prints*, 2010.

Eugene F. Brigham. *Fundamentals of financial management.* Dryden Press, Hinsdale, Ill., 2. ed edition, 1980.

Simon Carter, Manos Tsagkias, and Wouter Weerkamp. Semi-supervised priors for microblog language identification, February 2011.

Meeyoung Cha, Hamed Haddadi, Fabricio Benevenuto, and K P Gummadi. *Measuring User Influence in Twitter: The Million Follower Fallacy.* The AAAI Press, 2010.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines.

ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Robert L. Crouch. The volume of transactions and price changes on the new york stock exchange. *Financial Analysts Journal*, 26(4):pp. 104–109, 1970.

E F Fama, L Fisher, M C Jensen, and R Roll. The adjustment of stock prices to new information. *International Economic Review*, 10(1):1–21, 1969.

Eugene F. Fama. The Behavior of Stock-Market Prices. *The Journal of Business*, 38(1): 34–105, 1965a.

Eugene F. Fama. Random walks in stock-market prices. *Financial Analysts Journal*, 21: 55–59, 1965b.

Gabriel Pui Cheong Fung, Jeffrey Xu Yu, and Wai Lam. News Sensitive Stock Trend Prediction. *Advances in Knowledge Discovery and Data Mining : 6th Pacific-Asia Conference, PAKDD 2002, Taipel, Taiwan, May 6-8, 2002. Proceedings*, 2336, 2002.

Gyz Gidfalvi and C Elkan. Using news articles to predict stock price movements. *Department of Computer Science and Engineering University of California San Diego*, 2001.

Chih-Wei Hsu, Chih-Chung Chang, and Ch Lin. A practical guide to support vector classification. *Bioinformatics*, 1(1):1–16, 2010.

Wei Huang, Yoshiteru Nakamori, and Shou-Yang Wang. Forecasting stock market movement direction with support vector machine. *Comput. Oper. Res.*, 32:2513–2522, October 2005. ISSN 0305-0548.

Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, pages 679–688, 2006.

Jeffrey F Jaffe. Special information and insider trading. *Journal of Business*, 47(3):410–428, 1974.

Prem C. Jain and Gun-Ho Joh. The dependence between hourly prices and trading volume. *The Journal of Financial and Quantitative Analysis*, 23(3):pp. 269–283, 1988.

D.C. Johnston and D. Johnston. *Introduction to oil company financial analysis*. PennWell Corp., 2005.

Jonathan M. Karpoff. The relation between price changes and trading volume: A survey. *The Journal of Financial and Quantitative Analysis*, 22(1):pp. 109–126, 1987.

A. Khan and V. Zuberi. *Stock Investing for Everyone: Tools for Investing Like the Pros*. Wiley investment series. Wiley, 1999.

Rohit Khare and Doug Cutting. Nutch: A flexible and scalable open-source web search engine, 2004.

K Kim. Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1-2):307–319, 2003.

Danica Kirka and Gregory Katz. Burberry seizes the moment at london fashion week, 2011. URL `http://seattletimes.nwsource.com/html/nationworld/2014292256_apeubritainfashionweek.htmlaccessed[20-07-2011]`.

Stephen J. Klinger. Identifying trends with volume analysis. *The Stocks and Commodities*, 15:12:556–560, 1997.

Brajesh Kumar, Ajay Pandey, and Priyanka Singh. The dynamic relationship between price and trading volume - evidence from indian stock market. Working Papers id:2379, esocialsciences.com, 2010.

Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, a social network or a news media? In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 591–600, New York, NY, USA, 2010. ACM.

Victor L Bernard and Jacob K Thomas. Post-earnings- announcement drift : Delayed price response or risk premium ? *Journal of Accounting Research*, 27(1):1–36, 1989.

Hsuan-Tien Lin and Chih-Jen Lin. A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. *submitted to Neural Computation*, pages 1–32, 2003.

A W Lo and A C MacKinlay. A non-random walk down wall street. *The Economic Journal*, 113(491):F668–F670, 1999.

Burton G. Malkiel. *A Random Walk Down Wall Street, Completely Revised and Updated Edition*. W. W. Norton & Company, 2003a.

Burton G. Malkiel. The efficient market hypothesis and its critics. *Journal of Economic Perspectives*, 17(1):59–82, 2003b.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

Nicole Martinelli. Apple doesnt do twitter, but steve jobs stepping down is killing the trending topics, 2011. URL `http://www.cultofmac.com/apple-doesnt-do-twitter-but-steve-jobs-stepping-down-is-killing-the-trending-topics/110180accessed[25-08-2011]`.

Marc-André Mittermayer. Forecasting intraday stock price trends with text mining techniques. *hicss*, 03:30064b, 2004. ISSN 1530-1605.

Trevor Mogg. Twitter users respond to steve jobs decision to step down, 2011. URL `http://www.digitaltrends.com/apple/twitter-users-respond-to-steve-jobs-decision-to-step-down/accessed[25-08-2011]`.

I G Morgan. Stock prices and heteroscedasticity. *Journal of Business*, 49(4):496–508, October 1976.

Francesco Parrella. Online support vector regression. *Department of Information Science, University of Genoa Italy*, 2007.

S. Petrovic, M. Osborne, and V. Lavrenko. The Edinburgh Twitter Corpus. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media*, page 25, 2010.

M.J. Pring. *Technical analysis explained: the successful investor's guide to spotting investment trends and turning points.* McGraw-Hill, 2002.

Michael S Rashes. Massively confused investors making conspicuously ignorant choices (mcimcic). *Journal of Finance*, LVI(5):1911–1927, 2001.

Joshua Ritterman, Miles Osborne, and Ewan Klein. Using prediction markets and twitter to predict a swine flu pandemic. *Forecast*, 2009.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users. *Proceedings of the 19th international conference on World wide web WWW 10*, page 851, 2010.

Robert P. Schumaker and Hsinchun Chen. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Trans. Inf. Syst.*, 27:12:1–12:19, March 2009.

Robert J. Shiller. *Irrational exuberance.* Princeton Univ. Pr., Princeton, NJ [u.a.], 2001.

Walter Sun. Relationship between trading volume and security prices and returns. *Analysis*, c, 2003.

Devendra Tayal. Comparative analysis of the impact of blogging and micro-blogging on market performance. *Engineering*, 1(3):176–182, 2009.

James Thomas and Katia Sycara. Integrating genetic algorithms and text learning for financial prediction. In *in Proceedings of the Genetic and Evolutionary Computing 2000 Conference Workshop on Data Mining with Evolutionary Algorithms, Las Vegas*, pages 72–75, 2000.

M.C. Thomsett. *Mastering Fundermental Analysis: How to Spot and Pick Winning Stocks Like the Pros.* John Wiley & Sons Australia, Limited, 2000.

Andranik Tumasjan, Timm O Sprenger, Philipp G Sandner, and Isabell M Welpe. Predicting elections with twitter : What 140 characters reveal about political sentiment. *Word Journal Of The International Linguistic Association*, pages 178–185, 2010.

Twitter. 200 million tweets per day, 2011. URL `http://blog.twitter.com/2011/06/200-million-tweets-per-day.html`.

Zhou Wang and Alan Conrad Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(January):98–117, 2009.

Audrey Watters. How recent changes to twitter's terms of service might hurt academic research, 2011. URL `http://www.readwriteweb.com/archives/how_recent_changes_to_twitters_terms_of_service_mi.php`.

M S A Wolfram. Modelling the stock market using twitter. *iccsinformaticsedacuk*, 2010. URL `http://www.iccs.informatics.ed.ac.uk/~miles/msc-projects/wolfram.pdf`.

B.O.N. Wyss. *Fundamentals of the stock market*. Fundamentals of Investing. McGraw-Hill, 2000.

B. Wthrich, D. Permunetilleke, S. Leung, V. Cho, J. Zhang, and W. Lam. Daily prediction of major stock indices from textual www data. In *IN PROCEEDINGS OF THE 4TH INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING - KDD-98*, pages 364–368, 1998.

Ailun Yi. Stock market prediction based on public attentions: a social web mining approach. *Science*, 2009.

Charles C. Ying. Stock market prices and volumes of sales. *Econometrica*, 34:676–685, 1966.

Xue Zhang, Hauke Fuehres, and Peter A Gloor. Predicting stock market indicators through twitter i hope it is not as bad as i fear. *Anxiety*, pages 1–8, 2009.

Appendix A

Web Crawler

```
from threading import Thread
from urllib2 import urlopen
from pymongo import Connection
from UserDict import UserDict
from time import sleep
from BeautifulSoup import BeautifulSoup
import logging

logging.basicConfig(level=logging.WARNING, format='%(asctime)s␣%(levelname)s␣%(message)s')

vi = set()

class Downloader(Thread):
    def __init__(self, db):
        Thread.__init__(self)
        self.db = db

    def run(self):
        logging.info('starting␣downloader')

        while True:
            try:
                node = self.db.to_download.find_and_modify(limit=1, remove=True)

                if node is None:
                    logging.info('nothing␣to␣download,␣waiting␣...')
                    sleep(5)
                    continue

                link = node['link']
                user = node['user']

                if user in vi:
                    logging.warning('already␣visited␣user:␣{0}'.format(user))
                vi.add(user)

                logging.info('crawling:{0}'.format(link))
                html = urlopen(link).read()

                self.db.to_parse.insert({'user':user, 'html':html})
                self.db.visited.insert({'user':user})
            except Exception:
                logging.exception('error␣downloading␣html')


class Parser(Thread):
    def __init__(self, db):
        Thread.__init__(self)
        self.db = db

    def run(self):
        while True:
            node = None
            try:
                node = self.db.to_parse.find_and_modify(limit=1, remove=True)

                if node is None:
                    logging.info('nothing␣to␣parse,␣waiting␣...')
                    sleep(5)
                    continue

                soup = BeautifulSoup(node['html'])

                profile = self.extract_profile(soup)
                self.db.profiles.insert(profile)

                contacts = self.extract_contacts(soup)

                for c in contacts:
                    exists = self.db.visited.find_one({'user':c})
                    if exists is None:
                        self.db.to_download.insert({'user': c, 'link':'http://www.twitter.com/' + c})

                tweets = self.extract_tweets(soup, profile['user'])
                if len(tweets) == 0:
                    continue

                self.db.tweets.insert(tweets)

                more_url = soup.find('a', 'round_more')
                if more_url is not None:
                    self.db.to_download.insert({'user':profile['user'], 'link':'http://www.twitter.com/' + more_url['href']
            except AttributeError:
                pass
            except Exception:
                logging.exception('error␣parsing␣html␣for␣user:␣{0}'.format(node['user']))
    def extract_profile(self, soup):
        username = soup.find('div', 'screen-name').string
        location = soup.find('span', 'adr').string
        following = soup.find('span', id='following_count').string
        followers = soup.find('span', id='follower_count').string
        return {'user':username, 'location':location, 'following':following, 'followers':followers}

    def extract_tweets(self, soup, user):
        tweets = []
        for span in soup.findAll('span', 'status-body'):
            tweet_id = span.find('a', 'entry-date')['href']
            date = span.find('span', 'published_timestamp')['data'][11:-2]
            content = ''.join(str(t) for t in span.find('span', 'entry-content').contents)
            tweets.append({'tweet_id':tweet_id, 'user':user, 'date':date, 'content': content})
        return tweets

    def extract_contacts(self, soup):
        return set([c['href'][1:] for c in soup.findAll('a', attrs={'rel':'contact'})])


def crawler():
    from sys import argv

    try:
        connection = Connection()
        db = connection.twitter

        if '-reset' in argv:
            logging.info('reseting_db')
            logging.info('dropping_twitter_db')
            connection.drop_database('twitter')

            db = connection.twitter
            db.to_download.insert({'user':'stephenfry', 'link':'http://www.twitter.com/stephenfry'})

        for i in range(20):
            downloader = Downloader(db)
            downloader.start()

        parser = Parser(db)
        parser.start()
    except Exception:
        logging.exception('oops')

crawler()
```

Appendix B

SVR Modeller

```
from numpy import genfromtxt
from scikits.learn import cross_val
from multiprocessing import Pool
from scikits.learn.preprocessing import scale
from scikits.learn.svm import SVR
import datetime as dt
import math as math

def log(text):
    msg = '{0} - {1}'.format(dt.datetime.utcnow(), text)
    print msg
    with open('log.txt', 'a') as f:
        f.write(msg + '\n')

def load_data(company):
    path = '/users/andrew/uni-work/training_data/new_{0}.csv'.format(company)
    data = genfromtxt(path, unpack=True, delimiter = ',')
    return (data[0,:], data[1,:], data[2,:], data[3,:], data[4,:])

def build_features(data, selected_features, scaled):
    volumes, ma, wc, hc, qs = data
    features, responses = [], []

    for i in range(3, len(volumes) - 1):
        f = []

        if 'yy_volume' in selected_features:
            f.append(volumes[i - 2])
        if 'yyy_volume' in selected_features:
            f.append(volumes[i - 3])
        if 'y_volume' in selected_features:
            f.append(volumes[i - 1])
        if 'y_count' in selected_features:
            f.append(wc[i - 1])
        if 'y_hash' in selected_features:
            f.append(hc[i - 1])
        if 'y_score' in selected_features:
            f.append(qs[i - 1])
        if 'volume' in selected_features:
            f.append(volumes[i])
        if 'count' in selected_features:
            f.append(wc[i])
        if 'hash' in selected_features:
            f.append(hc[i])
        if 'score' in selected_features:
            f.append(qs[i])

        if 'volume_change' in selected_features:
            f.append(volumes[i -1] - volumes[i])

        if '1y_volume_change' in selected_features:
            f.append(volumes[i-2] - volumes[i-1])

        if '2y_volume_change' in selected_features:
            f.append(volumes[i-3] - volumes[i-2])


        if 'count_change' in selected_features:
            f.append(wc[i -1] -wc[i])
        if 'hash_change' in selected_features:
            f.append(hc[i -1] - hc[i])
        if 'score_change' in selected_features:
            f.append(qs[i -1] -qs[i])
        if 'score_change_y' in selected_features:
            f.append(qs[i -2] -qs[i - 1])

        features.append(f)
        responses.append(volumes[i + 1])

    if scaled:
        features = scale(features).tolist()

    for i in range(len(features)):
        print features[i][0], responses[i]

    return features, responses

def get_mape(base, predict):
    n = len(base)
    return sum(math.fabs((base[i] - predict[i]) / base[i]) for i in range(n)) / n

def get_rmse(base, predict):
    n = len(base)
    return math.sqrt(sum((base[i] - predict[i])**2 for i in range(n)) / n)

def get_svr(kernel, params):
    if kernel == 'linear':
        c, epsilon  = params
        return SVR(kernel = kernel, C = c, epsilon = epsilon)
    else:
        c, gamma = params
        return SVR(kernel = kernel, C = c, gamma = gamma)

def param_range(kernel):
    if kernel == 'linear':
        for c in (10**i for i in range(-5, 5)):
            for e in (10**i for i in range(-5, 5)):
                yield (c, e)
    else:
        for c in (10**i for i in range(-5, 5)):
            for e in (10**i for i in range(-5, 5)):
                yield (c, e)

def k_fold(kernel, features, labels, params, k = 10):
    predictions, actuals = [], []
    folds = cross_val.KFold(len(labels), k)

    for train_index, test_index in folds:
        train_x, train_y = [], []
        test_x, test_y = [], []

        for i, value in enumerate(train_index):
            if value:
                train_x.append(features[i])
                train_y.append(labels[i])
            else:
                test_x.append(features[i])
                test_y.append(labels[i])

        svr = get_svr(kernel, params)
        res = svr.fit(train_x, train_y)
        p = svr.predict(test_x)
        predictions.extend(p)
        actuals.extend(test_y)

    mape = get_mape(actuals, predictions)
    rmse = get_rmse(actuals, predictions)
    return mape, rmse, params

def best_params(pool, kernel, features, labels):
    results = (k_fold(kernel, features, labels, params) for params in param_range(kernel))
    results = sorted(results)

    mape, rmse, params = results[len(results) - 1]
    log('crap:␣{0}␣-␣{1}'.format(mape, params))
    mape, rmse, params = results[0]
    log('best:␣{0}␣-␣{1}'.format(mape, params))
    return params
```

```
def forecast(pool, kernel, scaled, company, selected_features, C = None, epsilon = None):
    data = load_data(company)
    features, labels = build_features(data, selected_features, scaled)
    predictions, actuals = [], []

    # leave on out validation
    for i in range(len(labels)):
        log('company {0} - progress: {1}'.format(company, i))

        train_features = list(features)
        train_labels = list(labels)

        len_before_pop = len(train_features)

        test_feature = train_features.pop(i)
        test_label = train_labels.pop(i)

        len_after_pop = len(train_features)

        assert(len_before_pop != len_after_pop)
        print len_before_pop, len_after_pop



        params = None
        if C is None:
            params = best_params(pool, kernel, train_features, train_labels)
        else:
            params = (10**C, epsilon)

        svr = get_svr(kernel, params)
        res = svr.fit(train_features, train_labels)
        p = svr.predict(test_feature)

        predictions.append(p[0])
        actuals.append(test_label)

        log(get_mape([test_label], p))

    mape = get_mape(actuals, predictions)
    rmse = get_rmse(actuals, predictions)

    return mape, rmse, predictions, actuals

def write_results(test_name, company, results):
    mape, rmse, predictions, actuals = results
    with open('writeup/{0}_{1}.csv'.format(test_name, company), 'w') as f:
        f.write('test: {0} \n'.format(test_name))
        f.write('{0} \n'.format(company))
        f.write('{0} \n'.format(mape))
        f.write('{0} \n'.format(rmse))
        f.write('results: \n')
        for i in range(len(predictions)):
            f.write('{0},␣a:{1}\n'.format(predictions[i], actuals[i]))

def easier_write(test_name, results):
    appl = results['apple']
    goog = results['google']
    vod = results['vodafone']
    brby = results['burberry']

    with open('writeup/{0}_latex.csv'.format(test_name), 'w') as f:
        f.write('MAPE␣&␣%.3f␣&␣%.3f␣&␣%.3f␣&␣%.3f␣\n' % (appl['mape'], goog['mape'], vod['mape'], brby['mape']))
        f.write('RMSE␣&␣%d␣&␣%d␣&␣%d␣&␣%d␣\n' % (appl['rmse'], goog['rmse'], vod['rmse'], brby['rmse']))

    for k in results:
        predictions, actuals = results[k]['predictions'], results[k]['actuals']
        with open('writeup/{0}_{1}.csv'.format(test_name, k), 'w') as f:
            for i in range(len(predictions)):
                f.write('{0},␣{1}␣\n'.format(actuals[i], predictions[i]))



def test(test_name, features, kernel = 'linear', scaled = True, C = None, epsilon = None):
    pool = Pool(processes=2)
    results = {}
    for company in ['apple', 'google', 'vodafone', 'burberry']:
        results[company] = {}
        mape, rmse, predictions, actuals = forecast(pool, kernel, scaled, company, features, C, epsilon)
        log('mape:␣{0},␣rmse:␣{1},␣company:␣{2}'.format(mape, rmse, company))
        results[company]['mape'] = mape
        results[company]['rmse'] = rmse
        results[company]['predictions'] = predictions
        results[company]['actuals'] = actuals
    easier_write(test_name, results)

def params_test(kernel = 'linear', features = 'volume'):
    results = []
    for company in ['apple', 'google', 'vodafone', 'burberry']:
        for C, epsilon in param_range('linear'):
            mape, rmse, predictions, actuals = forecast(None, kernel, True, company, features, C, epsilon)
            with open('writeup/params_test.csv', 'a') as f:
                f.write('C:␣{0}␣$\gamma$:␣{1}␣&␣{2}␣&␣{3:.3f}␣&␣{4:.0f}␣\\␣\n'.format(C, epsilon, company, mape, rmse,0))

def percentage_change(company):
    data = load_data(company)
    volumes, ma, wc, hc, qs = data

    from numpy import mean, median

    changes = []

    for i in range(1, len(volumes)):
        change = volumes[i] - volumes[i - 1]
        change = change / volumes[i - 1]
        change = change * 100
        changes.append(change)

    print 'mean:', mean(changes)
    print 'median:', median(changes)
```