7-13-2021 2:00 PM

Predicting Stock Market Sector Sentiment Through News Article Based Textual Analysis

William A. Beldman, *The University of Western Ontario*

Supervisor: Mercer, Robert E., *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science

Abstract

Investors seek to take advantage of computer technology to gain an edge on their investments. This can be done through quantitative (historical number-based) analysis or qualitative (natural language-based) analysis. Subject matter experts have been known to make predictions between 70 and 79% accuracy at best and less than 50% accuracy on average. Sophisticated algorithms through qualitative analysis are known to demonstrate more successful market predictions for specific stocks. It stands to reason that the same technique could work just as well or better for attempting to predict entire sectors of the stock market. By using indices and exchange traded funds, it is possible to track entire sectors of the stock market and make evaluations. The S&P 500 Energy index demonstrated itself to be especially bullish over the three-month period from January 22, 2008 to April 22, 2008 and especially bearish over the three-month period from July 14, 2008 to October 10, 2008. The three-month period from January 30, 2012 to April 30, 2012 was the least volatile period and can serve as a base-line measure for the other two. By extracting news stories over these three time periods, and analyzing specific parts of speech (verb, adverb, and adjective) in those stories over those periods, it is possible to make predictions about expected changes in the S&P 500 Energy index between 55% and 85%. The bullish three-month period was especially predictable at a rate of 85.22% on average. A bigram analysis over the same three three-month periods is also attempted with an accuracy between 50% and 55% on average. Applying an artificial neural network over the same three three-month periods can achieve an accuracy of between 55% and 65%.

Keywords: Natural language processing, NLP, Monogram, Bigram, Verb, Adverb, Adjective, Parts of speech, Stock market, Exchange traded funds, ETF, S&P, Factiva, New York Stock Exchange, NYSE, Artificial intelligence, AI

Lay Abstract

Investors seek to take advantage of computer technology to gain an edge on their investments. This can be done through analysis of number-based data such as sales figures or stock prices, or through analysis of word-based data such as press releases or news stories. Subject matter experts have been known to make predictions between 70 and 79% accuracy at best and less than 50% accuracy on average. Sophisticated algorithms are known to demonstrate more successful market predictions for specific stocks than subject matter experts. It stands to reason that the same technique could work just as well or better for attempting to predict entire sectors of the stock market. By using indices and exchange traded funds, it is possible to track entire sectors of the stock market and make evaluations. The S&P 500 Energy index demonstrated itself to be especially bullish over the three-month period from January 22, 2008 to April 22, 2008 and especially bearish over the three-month period from July 14, 2008 to October 10, 2008. The three-month period from January 30, 2012 to April 30, 2012 was the least volatile period and can serve as a base-line measure for the other two. An analysis of news stories over these three time periods was done using Artificial Intelligence techniques to predict stock market sentiment for the energy sector.

Keywords: Natural language processing, NLP, Monogram, Bigram, Verb, Adverb, Adjective, Parts of speech, Stock market, Exchange traded funds, ETF, S&P, Factiva, New York Stock Exchange, NYSE, Artificial intelligence, AI

Table of Contents

List of Tables

List of Figures

List of Appendices

Nomenclature

ANN Artificial Neural Network

DJIA The Dow Jones Industrial Average

ETF Exchange Traded Fund

GICS The Global Industry Classification Standard

LSTM Long Short-Term Memory

NLTK Natural Language Toolkit

NYSE The New York Stock Exchange

S&P Standard & Poors

Chapter 1

Introduction

There is a natural appeal to applying technological advancements for the purposes of economic advancement. This is true of a primitive tool such as using the plow to increase crop yield. This is true of a highly specialized machine to increase factory output. Applying the computer for direct economic benefit has been no exception. Computers are highly attractive tools for increasing economic output since they perform a subset of tasks that humans can do at a fraction of the time. The stock market presents humans the ability to increase wealth through nothing more than analysis and investment. A human will analyze the trends of the market and make decisions about where it would be best to invest their money. If their decision was wise, the investment will generate returns for the savvy investor. However, if their decisions were unwise, the investor risks losing their initial investment. Thus, any investor is required to attempt to maximize the benefits while minimizing the risks. Any investor could analyze any source of data they wish to better inform their decision, but with a plethora of data and often a narrow window of time to make such decisions, such an exercise will always present the investor with a limited data set to inform their decision. Thus, it would be natural to apply the modern computer as a tool to both expand the source of potential data and to make the same decisions in a fraction of the time.

Much research has been done attempting to apply sophisticated programs to maximize profits for the investor while minimizing losses. This paper will not directly attempt to demonstrate or simulate profits versus losses. Rather, this paper will merely demonstrate the predictability of daily stock market movements and the implications on profits and losses can be easily extrapolated.

1.1 A Historical Overview

The earliest stock market dates to the 14th century AD when Venetian traders would gather to exchange government securities. Antwerp, Belgium held a stock exchange as early as 1531 where exchanges of government, business, and individual debts took place. The Dutch, British, and

the French were the first to use such exchanges to fund expeditions to the East Indies by selling "shares" in a venture to willing investors. The respective companies could fund their expeditions and the investors could protect themselves from catastrophic loss by diversifying their investment across multiple expeditions. (Beattie (2017)) Since then, stock exchanges serving all types of investors and organizations can be found around the world including Tokyo, London, Shanghai, Hong Kong, and New York. The largest exchange is the New York Stock Exchange (NYSE) and is the exchange that is the subject of this study. (World-Stock-Exchanges.net (2017))

1.1.1 Tracking the stock market

Naturally, it would be advantageous to an investor for the price of various stocks to be traced over time allowing them to gain some insight into the general investor sentiment. Various organizations publish indices which are an aggregate price representation of a specific pool of stocks or stock types. For example, perhaps the most recognizable is the Dow Jones Industrial Average (DJIA) which tracks 30 significant stocks on the NYSE (Staff (2017a)). Various organizations track all manner of stocks by grouping them per categorical labelling such as market capitalization or the industry the company engages in. The number of such organizations is extensive but the S&P Global is perhaps one of the most recognizable. One of its subsidiary companies is the S&P Dow Jones Indices which manages and publishes the DJIA.

Stocks on the stock market present themselves in many different forms across the centuries. Some stocks represent simple shares in specific companies or government debts. Over time the stock market has reached a very sophisticated level where stocks represent many different ingenious investment vehicles. With indices well established and tracked, it is possible to develop a single fund that is itself an investment in a pool of stocks that the indices track. Paul Samuelson first argued the case for such a fund in 1974 suggesting that it would be beneficial to setup a portfolio that does nothing more than simply track the S&P 500 index. Such a fund is known today as an exchange traded fund (ETF) (Staff (2017b)). There are thousands of ETFs available to an investor, but they all have one goal in common: The fund attempts to track in line with an index rather than attempt to outperform an index. An ETF is a highly advantageous investment vehicle to an investor. The investor can protect their losses by distributing their investments across multiple companies that make up the ETF rather than expose themselves against the rise and fall of any single company. Companies also benefit as it attracts investors who might otherwise view them as too risky.

Shrewd investments in the stock market can produce large windfalls for an investor. However, poor investment choices can cause the investor to lose everything invested. Thus, it is imperative that any investor run through the difficult exercise of selecting investments that will maximize

their profit while also minimizing their risk. Naturally, the riskiest investments can produce the largest profits while the safest investments usually produce minimal profit. An ETF helps protect the investor, but it is still subject to the same profit and loss situations of any other stock. It is this constant balance between profit and risk that any investor will search for any information that helps guarantee the former and protects them from the later. (Dubner (2017))

1.2 Quantitative and Qualitative Analysis

To make the best investments, it is imperative that an investor gather information about potential investment options from the best sources available. Exactly which investment sources are best is difficult to deduce and often comes down to the preferences of the investor. However, each source of input an investor could consult falls into one of two categories: quantitative analysis and qualitative analysis. As the name implies, quantitative analysis involves analysis dealing with discrete numbers while qualitative analysis involves analysis dealing with abstract data. (Costantino and Coletti (2008) p. 2-12, 20-26)

1.2.1 Quantitative Analysis

The term quantitative analysis or the gathering of quantitative information applies to information that can be explicitly expressed in terms of numbers. Typically, this would include either the price of a stock or a derivative of the price, such as the price change over time, or the highs and lows over certain periods of time. Other popular sources of data include the volume which is the number of exchanged shares or the market capitalization which is the value of the shares available for trading in the stock market. Secondary information such as the earnings per share or information on dividends can also inform traders but do not typically serve as primary sources.

Given the nature of the quantitative data, it is typically represented to traders in a simplified fashion such as in a chart or through a tabular representation. This allows the trader to identify trends in the price movement of a stock more readily, such as a growth trend, or a low profit yield projection. Graphs and tables can be overlaid with other data so visual comparisons can be made with similar stocks, entire sectors of the stock market, or to the market as a whole.

The precise meaning of quantitative data comes down to trader preference and trader conclusions about the same data and can be difficult to predict. For example, a trader may identify that a 5% price change in a stock over a 5-day period while similar stocks received no such growth could signal a good buy opportunity as the anomaly could continue for a while increasing the value of their share. On the other hand, a trader may identify the same growth and fear that the trend will not continue, and the stock may soon drop to the level of similar stocks. In such a scenario, the trader

may choose to sell any existing shares or employ more sophisticated techniques for generating profit such as shorting the stock. Thus, quantitative analysis can not be sufficient in determining trader sentiment about a stock. Humans are unpredictable in this sense and thus other sources of input are employed to decide what exactly the quantitative data is saying.

The other serious disadvantage with quantitative data is that it is a representation of the stock after the events have already occurred. Using quantitative data gives the trader only a retrospective picture of the stock and (as is theorized) does not necessarily provide any prospective information. It is possible that by the time a trader realizes that a stock has risen in value by 5% over a 5-day period, the trader has also realized the 5-day lost potential for profit which cannot be undone.

1.2.2 Qualitative Analysis

Qualitative data can serve to supplement the trader with data that quantitative data does not serve. Qualitative data or qualitative analysis on the data involves gathering data about the stock that cannot be discretely represented with numbers. Typically, this involves extra knowledge about the company the stock represents. The most common source of such data is news stories about the company. The story may be very specific information about the actions of the company such as changes to the board of directors, or it may be more abstract information such as statistics on consumer spending and the implications that will have on the company represented by that stock.

The value of such qualitative data can be difficult to deduce as news stories can range from precise facts to speculative commentary. The trader must then deduce the value of their input data and decide whether the data is useful to inform their trading decisions or not. Qualitative data can also be overwhelming as the breadth and depth of information concerning a company can be infinite. The relationship between this data to quantitative data is also not always well-defined so it is up to the trader to establish that link intuitively. Some news stories may not necessarily be about the company the stock represents but about the stock representing the company making information extraction even more difficult to deduce.

1.2.3 A General Approach to Using Quantitative and Qualitative Analysis

A successful trader would combine both quantitative analysis and qualitative analysis to make the best decisions. The common approach for any investor, as summarized by Costantino and Coletti, takes three steps. First, the trader assesses the global position and risks of the current portfolio using the quantitative information available. Second, the trader takes a view of the market, based on the current quantitative and qualitative information available. Finally, the trader decides the strategy to put in place using the quantitative and risk-management information available. It is critical that the first two steps must be performed quickly before the market changes. This is

where augmenting trader knowledge with expert systems using algorithmic trading or automated sentiment analysis comes in to play.

1.3 Guiding approach for this study

Understanding the three step approach of assessing one's current position, gathering quantitative and qualitative data, and acting on that data quickly, the goal in this study is to explore methods of enhancing the second step. This will be done by gathering qualitative data and employing various techniques against the data to automatically infer sentiment. By automatically determining the likely sentiment, one would be better positioned to use the results to create an effective strategy.

Chapter 2

Background

2.1 A Theoretical Limitation - The random-walk theory

There is research regarding when it is possible to determine whether a trader has made the best decision possible. (Fama, 1965) first outlined a philosophical starting point from which all market movement can be compared. He argues that there does not appear to be any special hidden information that traders can take advantage of through intelligent analysis. In fact, the evidence seems to suggest that the price of any given stock is reflective of all the knowledge available and therefore that price is the most efficient price possible. To what extent this theoretical limit is true is somewhat open to question as there are some investors who appear to be able to beat the market consistently, however this could still be explained by a regression to the mean over a longer period. If one subscribes to the theory that the efficient market hypothesis is true, it should not be possible to employ any method to accurately predict stock price movement.

Similarly, Fama also demonstrates that stocks appear to demonstrate the trademark of a random walk. That is to say, price movement appears to be so unpredictable that the price movement is really indistinguishable from a random walk. Therefore, there is no amount of analysis of past data that could successfully predict future movement. In fact, Fama subscribes to this theory so vehemently that he suggests "the empirical evidence produced by this and other studies in support of the random-walk model is now so voluminous, the counterarguments of the chart reader will be completely lacking in force if they are not equally well supported by empirical work.".

In support of this research, Fama states the random-walk theory can be proven by demonstrating that each successive price movement is independent of previous prices and that those price movements demonstrate a probability distribution. In support of the independence theory, Fama concedes that individual investors behave rationally and incorporate intrinsic value as well as prior prices, yet in aggregate, investors cannot make perfect agreements on what the actual intrinsic value of a stock actually is. As data regarding the stock becomes available with no specific pattern,

investors will make reactions to that data at unpredictable rates and sometimes, in unpredictable ways. The necessary corollary then is the predicted future price of a stock at any point in time will follow a normal or Gaussian distribution. Fama would go on to demonstrate this fact by following prices across thirty different stocks on the DJIA from the end of 1957 to September 26, 1962.

2.2 Putting the theoretical limit to the test

From this seminal paper began the Sisyphean task of compiling empirical work to counter the random-walk model. Using the random-walk model as the baseline, and aided by computers, many researchers have put together work to develop novel ways to predict stock market movement either through quantitative, or qualitative analysis, or a combination of both.

One early attempt was published by (Wuthrich et al., 1998) in 1998. The researchers chose not to analyze individual stocks but compiled price movements for five different stock market indices. These indices were the Dow Jones Industrial Average (Dow), the Nikkei 225 (Nky), the Financial Times 100 Index (Ftse), the Hang Seng Index (Hsi), and the Singapore Straits Index (Sti). Wuthrich, etal. would then analyze news stories and through a frequency analysis, simply count the number of keywords found, apply weighting to those keywords, and use those weights to attempt to make predictions about how those words may (or may not) be influencing the indices. Covering a three-month period from December 6th, 1997 to March 6th, 1998, this covers 60 stock trading days or test cases. While the ability to accurately predict the price movement and the magnitude of movement was around 43.6% accuracy, it was at least promising to see that the system failed to predict the price movement direction only 19% of the time on average. In the case of the Dow Jones Industrial Average where news sources were most plentiful, the system failed to predict price direction only 8.3% of the time. Thus, it can be concluded that it could be possible to predict future price direction based on news story keywords about 80% of the time.

(Gidófalvi, 2001), published in 2001 an analysis that followed a similar approach. In this case, Gidófalvi chose a specific set of stocks to analyze and attempted to only predict directional movement (ie. Up, down, or unchanged). Gidófalvi also introduced the idea of using a Naïve Bayes Classifier to analyze the news stories. The 12 stocks chosen were CSCO, SUNW, MSFT, ORCL, WCOM, YHOO, RHAT, DELL, AMZN, LU, EBAY, and INTC, all from the NASDAQ and covered a three-month time period from November 14, 1999 to Februrary 11, 2000. The training set versus testing set was established ahead of time at January 10, 2000 at 9:34AM such that news articles published before this time were part of the training set and articles published after this time were part of the testing set. While Gidófalvi was able to identify a strong correlation between news articles and the behavior of stock prices both 20 minutes before and 20 minutes after a news story came out, the classifier had relatively poor predictive power. This may be the result of either using

a very narrow window of time (40 minutes) and most traders may not react to news stories quite this quickly. Also, important news tends to be repeated from multiple sources which all fell into the analysis but only the first article published will have any real significance on the price movement.

(Koppel and Shtrimberg, 2005) covered news stories from 2000 to 2002 and attempted to classify the sentiment of the news stories in accordance with the market's reaction based on the price that day. This is unlike previous work where human subjects would classify the news sentiment manually. They were able to classify the news sentiment at an accuracy of 70.3% using a linear SVM. Other learners such as the Naïve Bayes and decision trees yielded essentially the same results. A deeper review of the results was in line with expectations where words such as "shortfall", "negative", and "investigation" corresponded with negative news stories. Surprisingly, they found that positive news stories were not clearly identified with positive words, but rather could be identified by the absence of signature negative words. As a result, the positive news stories could make predictions up to 83.3% while negative news stories at 66.0%. The researchers suggest that the lower rating for negative news stories likely stems from news articles that are misclassified as they tend to not contain the signature negative words in the first place.

It is important to note that these results are not isolated to popular and well-known stock markets such as the NYSE. (Falinouss, 2007) followed the 20 most active stocks on the Tehran Stock Exchange Technology Management Company (TSESC) from 1383 to 1384 (2004 to 2005 in the Gregorian calendar). From related news stories on Yahoo! Business, Falinouss predicted price movement direction with an accuracy of 83%. (Aase, 2011) followed a similar approach for 20 stocks on the Oslo Stock Exchange (OSE). The stock prices were merged with news stories from Hegnar online, Newsweb, and Thomson Reuters ONE and a simulated profit of 3.33% per trade was achieved. (Li et al., 2014) also ran a similar approach on 22 stocks in the Hang Seng Index (HSI) on Hong Kong Stock Exchange over the five-year period from January 2003 to March 2008. News stories were gathered from FINET which is a major financial news vendor in Hong Kong. However instead of attempting to achieve accuracy on sentiment prediction, they only compared sentiment analysis versus a simple bag-of-words approach and found sentiment analysis to be useful for predicting stock, sector, and index levels. Finally, (Yasef Kaya and Elif Karsligil, 2010) followed this approach for Microsoft's stock (MSFT) specifically over a one-year period. Kaya used news stories from fool.com and achieved an accuracy prediction rate of 61%. Kaya exploited a certain pattern in parts of speech following word patterns of noun-verb such as "revenue generates" for positive sentiment or "loss have" for negative sentiment suggesting there is potential in following this approach with a bigram analysis.

As popularity of artificial neural networks rose in the 1980s and 1990s, the application of artificial neural networks to stock market prediction inevitably followed. An early attempt was employed by (White, 1988) following the daily stock price for IBM using 1974 to 1978 as the

training data and both 1972 to 1974 and 1978 to 1980 as testing data. The results are poor however the author notes three key findings: First, using neural networks against efficient markets will not be an easy task. Second, employing neural networks in this domain will be subject to overfitting. Third, a simple neural network is capable of modelling rich dynamic behaviour which would be required for any stock market predictor.

(Kimoto et al., 1990) followed a similar approach against the much broader TOPIX (Tokyo Stock Exchange Price Index). In addition to usual quantitative inputs extracted from the daily TOPIX values, several additional external quantitative inputs were used to influence the neural network including data such as the DJIA and foreign exchange rate. Covering January 1987 to September 1989, it was demonstrated that profits could be realized suggesting there could be value.

A much deeper and wider approach was employed by (Yoon and Swales, 1991) analyzing multiple groups of companies extracted from the Fortune 500 and Business week publications. In addition to using quantitative data, this is an early example of employing qualitative data as input to the neural network. In this case, the president's letter to shareholder was included and nine recurring themes were extracted and included as input. Again, the predictive capabilities were positive. 91% of the training data would match and 77.5% of the testing data would match.

Through the early 2000s, interest in artificial neural networks waned in favour of support vector machines. Several studies merged neural networks with SVM such as (Hassan et al., 2007) which employed quantitative data for Apple, IBM, and Dell from February 10, 2003 to September 10, 2004 for the training set and September 13, 2004 to January 21, 2005 for the test set. The results produced were considered as good as other non neural network approaches and how one should interpret that result can be somewhat ambiguous. A similar approach was employed by (Kara et al., 2011) against the ISE 100 Index (Istanbul Stock Exchange). Using 10 different quantitative indicators across January 2, 1997 to December 31, 2007, the results were generally positive. A neural network had a predictive accuracy of 75.74% and SVM had a predictive accuracy of 71.52%.

By 2010, computing power rose to a new level renewing an interest in artificial neural networks. Some of the more recent uses of neural networks also added other techniques. Support vector machines were used as noted previously. Other studies began to add other ideas like hybrid neural networks, genetic algorithms, and deep learning.

(Mostafa, 2010) employing a neural network against daily quantitative data on the KSE (Kuwait Stock Exchange) from June 17, 2001 to November 30, 2003. The results were considered good and supported earlier positive findings.

(Bollen et al., 2011) employed a novel input set for the time by adding Twitter sentiment to a neural network to make predictions about the DJIA. This appears to be a relatively rare use of qualitative data for neural networks for stock market prediction. Approximately 2.7 million users had their 9,853,498 tweets curated and evaluated for sentiment from February 28, 2008 to Decem-

ber 19, 2008. The tweets demonstrated good predictive accuracy with 86.7% correspondence with daily up/down movements of the DJIA.

More recent studies include (Wei, 2016) which used a hybrid neural network against the TAIEX (Taiwan Stock Exchange) Capitalization Weighted Stock Index from 2000 to 2006 across 7 sub datasets and the HSI (Hang Seng Stock Index) from 2000 to 2004 across 5 sub datasets. The results were found to be superior to other models analyzed. (Inthachot et al., 2016) employed a neural network with genetic algorithms on the SET50 (Stock Exchange of Thailand) index. Using 11 types of quantitative data from January 5, 2009 to December 30, 2014, the results were also found to be superior to other models analyzed. (Singh and Srivastava, 2017) used a neural network with deep learning techniques against the Google price index on the NASDAQ. Using 36 types of quantitative data from August 19, 2004 to December 10, 2015, the results were also found to be superior to other models analyzed.

2.3 Recent research

In recent years, most research has focused heavily on artificial neural networks and various deep learning techniques. (Day and Lin, 2019) employed a long short-term memory (LSTM) deep learning model against quantitative data representing 20 ETFs listed in the Taiwan market from January 2, 2008 to December 31, 2018 for a total of 2,768 trading days. The LSTM deep learning model demonstrated improvement over other machine learning models. (Li et al., 2020) employed an LSTM against the same data over the same time period cited earlier and also found an LSTM to be the superior approach. (Jin et al., 2020) also employed, among other models, an LSTM model against 96,903 comments posted about AAPL (Apple) on stocktwits (https://stocktwits.com/) between March 4, 2013, and February 28, 2018. Stocktwits comments are already labelled as "bullish" or "bearish" allowing for a simple "positive" or "negative" sentiment translation. Again, the LSTM demonstrated itself as having superior accuracy. (Moghar and Hamiche, 2020) employed an LSTM against price data for GOOG (Google) from August 19, 2004 to December 19, 2019 and NKE (Nike) from January 4, 2010 to December 19, 2019 and showed the LSTM to be a promising model. Some recent surveys of stock market prediction through sentiment analysis can be found by (Thakkar and Chaudhari, 2021) and (Jiang, 2020).

Sentiment analysis in a more general sense continues to be researched. Researchers commonly use social media sources and employ sentiment analysis against other domains such as analysing customer feedback and movie reviews (Yadav and Vishwakarma, 2020) (Habimana et al., 2020). Sentiment analysis has also been employed to address the COVID-19 outbreak (Barkur et al., 2020) (Alamoodi et al., 2021).

2.4 Further Reading

Related work on the subject include (Mittermayer, 2004) who developed NewsCATS (News Categorization and Trading System). NewsCATS is a news categorization engine for extracting news stories and automatically categorizing them into different news types and deriving trading rules based on the corresponding stock found in the news stories. In a market simulation, Mittermayer could achieve a profit of 0.21% profit per trade compared with 0.06% profit per trade in the best case for a random trader. (Schumaker and Chen, 2009) built the AZFinText system which functions similarly to the NewsCATS. After following a similar approach over a five-week period from October 26, 2005 to November 28, 2005, achieved 57.1% classifier accuracy for up to 20 minutes after the story is published. Schumaker and Chen translated this into a predicted 2.06% return in a trading simulation. An interested reader may also wish to review (Hagenau et al., 2013) for a meta-analysis on the aforementioned work and other accuracy attempts.

With regards to artificial neural networks, (Abu-Mostafa and Atiya, 1996) describe a model for anyone wishing to employ neural networks to financial forecasting. (Adya and Collopy, 1998) is a good meta-study covering 48 studies from 1988 to 1994. 22 of the 48 studies employed neural networks and 18 of those 22 studies resulted in positive results.

Readers interested in some work with support vector machines, in addition to the aforementioned studies, could also refer to (Tay and Cao, 2001; Cao and Tay, 2001, 2003) and (Kim, 2003).

Chapter 3

Overview

The general approach for this research is inline with some of the earlier research on the subject matter. A methodology for selecting which stocks to track is chosen. Once the stock is chosen, news articles pertaining to that stock are extracted and tracked alongside price movements of the stock. The stock movement will serve as an indicator to the general sentiment of the news story. Natural language processing techniques will be employed to then attempt to make judgments about what the expected sentiment will be and in turn could be used to predict the expected price movement. Finally, the accuracy of the results will be analyzed.

In contrast to earlier research on the subject, this research will not pertain to individual stocks tied to specific companies or organizations. Rather, the purpose of this research is to attempt to apply earlier techniques to entire sectors of the stock market. The hypothesis is that positive news stories for one company in the sector actually translates to positive sentiment for the entire sector regardless of which company may be in the news. Similarly, negative news stories for one company in the sector will translate into negative sentiment for the entire sector. Therefore, it is possible to track all news stories pertaining to a specific sector to make predictions about the sentiment of the sector itself.

This research consists of the following steps:

1. Identify a set of sectors which can be easily tracked by an ETF index (Section 3.1)
2. Identify a set of sectors which have a large set of associated news stories (Section 3.1)
3. Identify candidate sectors to track by taking the intersection of the above two sets (Section 3.1)
4. Out of the candidate set, identify a sector which demonstrates the largest price movements. This should translate into the clearest results when attempting to apply natural language processing techniques for price movement predictions. (Section 3.2)

5. Once the sector is selected, merge most daily news stories with the corresponding daily closing prices to classify the news stories. (Chapter 4)

6. Using the daily closing price as a proxy for true sentiment of the articles published that day, use a training set to make predictions on the likely sentiment of the remaining unclassified news stories. (Chapter 5)

7. Record the success versus failure rate (Chapters 6 and 7)

3.1 Identify a set of sectors

There is a standard employed by the financial industry to identify sectors in the global economy. The Global Industry Classification Standard (GICS) [https://www.msci.com/gics] identifies 11 different sectors. These 11 sectors are summarized in Appendix B on page 117. Energy covers mostly oil and gas. Materials covers most other raw manufacturing including mining. Industrials covers other types of manufacturing including capital goods, commercial and professional services, and transportation. Consumer discretionary covers any industry marketing directly to consumers through either goods or services with exception to consumable staples. Consumer staples covers remaining consumer goods that are typically consumed such as food and beverage. Health care covers both health-related equipment and services. Financials cover all financial economic activity including insurance. Information Technology covers both software and hardware computer equipment and services with exception to telecommunications. Telecommunications covers any form of communication. Utilities cover standard utility companies. Finally, Real Estate covers real estate activity such as real estate investment trusts and related development and services.

Standard & Poors, a leading financial services company, maintains exchange traded funds that match with the GICS sectors. The S&P 500 measure tracks the 500 stock indices with the largest market capitalization. S&P breaks down each sector similarly with 11 exchange traded funds that track the top 500 stock indices with the largest market capitalization in a sector. These 11 funds are S&P 500 Materials, S&P 500 Energy, S&P 500 Financials, S&P 500 Health Care, S&P 500 Industrials, S&P 500 Information Technology, S&P 500 Telecommunication Services, S&P 500 Utilities, S&P 500 Consumer Staples, S&P 500 Consumer Discretionary, and S&P 500 Real Estate.

With 11 exchange traded funds with values that can be quantitatively tracked, the next task is to extract price values over an extended period. These values would be useful to identify top candidate time periods for extended analysis. The time period selected for this research was ten years. This is the maximum time period that is publicly available and provides a sufficient number of data points from which to work with. Therefore, price values from January 31, 2006 to February 23, 2016 was

used. With ten years of data readily available and therefore 2469 data points from which to choose, the next step was to determine a good subset to analyze. It would be cumbersome to try to gather news articles for all stock indices for the entire ten-year period. I suspected that over such a long period of time, with a large number of upwards and downwards trends in the market, the results would be largely irrelevant for data analysis anyway. Therefore, for the purposes of comparison between individual indices and for comparison to individual time periods, I specifically calculated the best, worst, and most average price movements over 3, 6, 12, 24, 36, 60, and 120-month periods. That is to say, three-month, six-month, 1 year, 2 years, 3 years, 5 years, and 10-year time periods. Given some of the variance based on the date the data was downloaded, each index produced 12 potential ten-year periods from which to choose from. The 10-year period is recorded just for completeness and perspective on the data and was not intended to be used as a reasonable candidate for further analysis.

It is important to note that the New York Stock Exchange is not open every day of the year. It is open every weekday with exception to observed holidays. It is closed on the standard banking holidays of New Years Day, Martin Luther King Day, Washington's Birthday (also known as President's Day), Good Friday, Memorial Day, Independence Day, Labor Day, Thanksgiving, and Christmas. During the observed time period, the NYSE was also closed Tuesday, January 2nd, 2007 as an observed day of mourning for the passing of former president Gerald Ford. This accounts for the 2469 data points rather than the 2600 that would be expected if it were open 5 days a week for 52 weeks for ten years. The number of trading days in a given year is therefore 252 days unless an exception is made (Intercontinental Exchange, 2017) (Strategesis, 2017). Out of the remaining time periods of three months, six months, 1 year, 2 years, 3 years, and 5 years, the next step was to determine the largest rise in price, the largest drop in price, and the least movement in price over each of those time periods. This calculation was done with the python script identified in Appendix A on page 69 where each data point was extracted out of a CSV file and compared with the data point a number of days earlier in accordance with each time period. That is to say, 63 days for three months, 126 days for six months, 252 days for 1 year, 504 days for 2 years, 756 days for 3 years, 1260 days for 5 years, and 2520 days for ten years.

The market capitalization changes over the reviewed time period can be found in Figure 3.1 on the following page.

3.2 Finding large price movements

The next step was to analyze each of the 33 data points produced and identify whether there is a particular index that stands out amongst the rest as particularly volatile over a certain time period in comparison to the others. If it were possible to identify expected price movements using natural

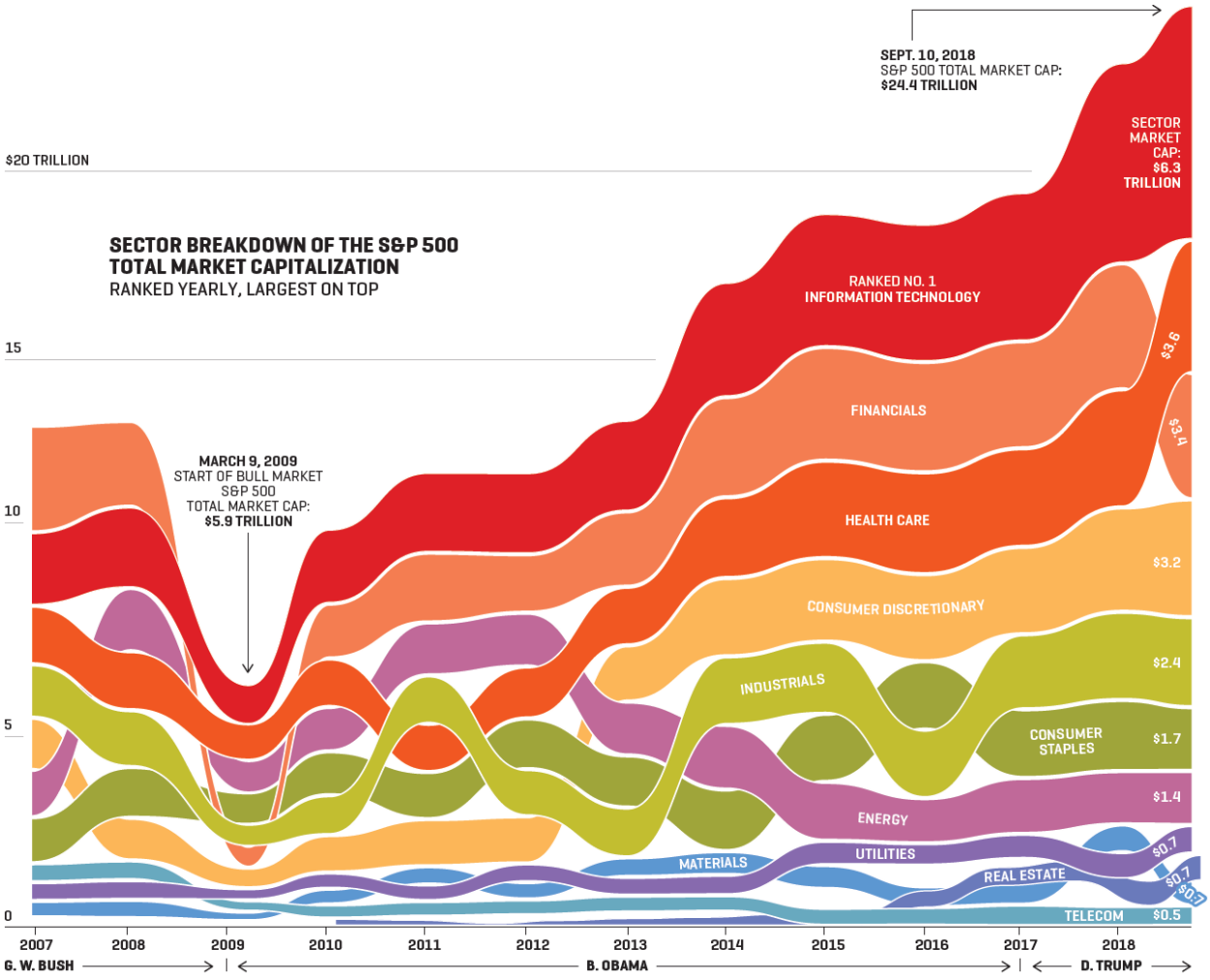

Figure 3.1: Graphic demonstrating relative market capitalization differences over the analyzed time period (Rapp and Leaf, 2018)

language processing techniques at all, it would likely become most apparent against an index that is demonstrably volatile. To find the best index to analyze, I recorded each of the 11 indices as recorded in Appendix B on page 117. The best, worst, and most average movements over each time period was recorded. Note that these values are raw moving averages over the time periods and purposely do not take into account the potential for significant price rises and drops within each time period. For example, it is possible that while the S&P 500 Materials index achieved its greatest price movement of $60.55 between March 3, 2009 and June 2, 2009, it could have dropped significantly and recovered significantly within that three-month period. The point is not to over analyze the data to find the perfect index for further study, but rather to identify one that seems to stand out amongst the rest.

The 11 best, worst, and most average data points were compared against each other to find the average price, the standard deviation, and then to compare the normalized values against each other. In the three-month time period, the S&P 500 Energy index stood out significantly in both the best and worst categories. This particular index rose $121.53 from January 22, 2008 to April 22, 2008 which is almost one and half deviations from the average movement the other 10 indices had. Coincidentally, the S&P 500 Energy index also dropped the most significantly compared to the other 10 indices. This time period was from July 14, 2008 to October 10, 2008. In this case, the index dropped $272.44 which is almost 2 and a half deviations more than the other indices during their worst three-month period. The most average time period for the 11 indices is recorded just for completeness and further comparison. The most average data did not factor any further in the study.

It is not surprising that each of the 11 indices dropped significantly during the autumn of 2008. This coincides with massive drops in all equities observed during chaotic times in the U.S. economy known as the housing crisis and the Great Recession. Large amounts of capital withdrew from the stock market at this time. It was also interesting to note that August 22, 2008 to November 20, 2008 was the most significant three-month price drop in 8 out of the 11 indices. As it turns out, the S&P 500, which measures the stock market as a whole, reached its lowest point in 11 and a half years on November 20, 2008. Thus, it should not be surprising that most of the indices dropped their lowest amount during the same time period. Nevertheless, the S&P 500 Energy index was hit the hardest as it dropped at a rate larger than all the others, relatively speaking.

The S&P 500 Energy index immediately began to stand out as an ideal candidate to analyze. Other indices which might have been worth considering was the S&P 500 Health Care which experienced a significant three-month rise in late 2014 but demonstrated a relatively average three-month drop at its lowest point. S&P 500 Information Technology experienced a significant rise in mid 2015 but did not demonstrate a convincing drop in autumn 2008, especially relative to the S&P 500 Energy index. S&P 500 Telecommunication Services did stand out as interesting as it

only rose by $25.35 the second quarter of 2007 and dropped by $39.44 in autumn 2008. The three-month rise was the smallest of the 11 indices, and the three-month drop was the smallest drop of the 11 indices. This was out of the ordinary, but it is probable that this index is not particularly popular compared to the rest and therefore is less susceptible to significant price movements. Had S&P 500 Energy not stood out as particularly interesting, it might have been worthwhile to investigate the S&P 500 Telecommunication Services more thoroughly.

After analyzing the best, worst, and most average three-month time periods, the next step was to look at large time periods to see how things changed over greater periods of time. The S&P 500 Energy index continued to stand out amongst the rest during a six-month period. From August 26, 2010 to February 25, 2011, the S&P 500 Energy index grew by $196.77 which was more than two standard deviations off the average movement of all the other indices. It is also interesting to note that this six-month period did not overlap the previously discovered best three-month rise in price. The drop from May 23, 2008 to November 20, 2008 was also the most significant drop of the 11 as it stood at more than two standard deviations less than all the others. The S&P 500 Health Care and S&P 500 Information Technology indices demonstrated and even greater regression to the mean over the six-month time period. The S&P Telecommunication Services continued to demonstrate resistance to volatility.

The trend continued to hold true for the twelve-month time period as the S&P 500 Energy rose $194.53 from September 20, 2006 to September 21, 2007. Only S&P 500 Health Care grew more points over a twelve-month period. The largest drop from April 22, 2008 to April 22, 2009 was the second largest drop over a twelve-month period second to only the S&P 500 Financials index. Given the uncertainty in the U.S. economy around 2007, it is not surprising that the financial sector dropped significantly. The other candidate indices of S&P 500 Health Care, Information Technology, and Telecommunication Services continued to appear more average than the S&P 500 Energy index.

The remaining two, three, and five-year periods showed the S&P 500 Energy price movements to appear more average amongst the other ten. From this analysis, it appears Energy seems to demonstrate a particular susceptibility to significant volatility over shorter periods of time but tends to hold steady over longer periods of time. This is confirmed by the ten-year analysis as between the entire period from February 3, 2006 and February 9, 2016, the S&P 500 Energy index grew by only $0.71. With exception to the S&P 500 Financials index, all other indices grew quite significantly over this ten-year period. One might then see the S&P 500 Energy index as a particularly poor investment as it did not generate any significant growth, however, it did not demonstrate any significant drop and therefore could be viewed as a useful store of wealth.

3.3 Choosing an index to track

After this analysis, with 231 data points reviewed across 7 time periods, I decided the S&P 500 Energy index demonstrated itself as the most useful index for further analysis. It has periods of significant growth and significant falls, but this appears to not be anomalous like, say, the financial sector did in 2008. This is especially true over the shorter three-month time period, so I focused my attention to the periods from January 22, 2008 to April 22, 2008, July 14, 2008 to October 10, 2008, and January 30, 2012 to April 30, 2012. For completeness, these three time periods break down as follows: January 22, 2008 to April 22, 2008 experienced 42 positive price movement days and 21 negative price movement days. From July 14, 2008 to October 10, 2008, the S&P 500 Energy index experienced 23 positive price movement days and 40 negative price movement days. From January 30, 2012 to April 30, 2012, the S&P 500 Energy index experienced 32 positive price movement days and 32 negative price movement days. It is important to note that each of the time periods had a non-trivial number of negative days during the period of high growth and a non-trivial number of positive days during the period of a high drop in price. This will help ensure that the news stories gathered will not be significantly skewed in either direction during any time period. Conversely, one might decide that if the price movement was almost entirely positive for a three-month period, the associated news stories would be entirely positive during this time period and therefore the analysis may not be as useful as negative news stories are not being brought into the mix to help even out the analysis. By including both positive and negative news stories in this analysis, it becomes more resistant to allowing the results to be self-fulfilling so that positive stories are released during positive time periods and negative stories are released during negative time periods. Charts for the three analyzed time periods can be found in Figures 3.2, 3.3, and 3.4 on the following page.

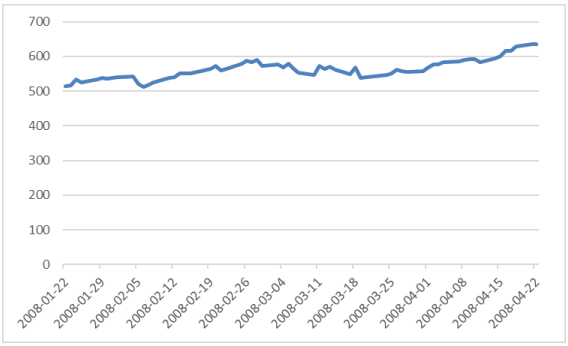

Figure 3.2: S&P 500 Energy (January 22, 2008 - April 22, 2008)

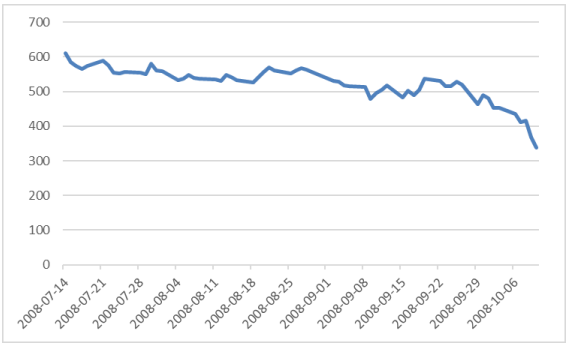

Figure 3.3: S&P 500 Energy (July 14, 2008 - October 10, 2008)

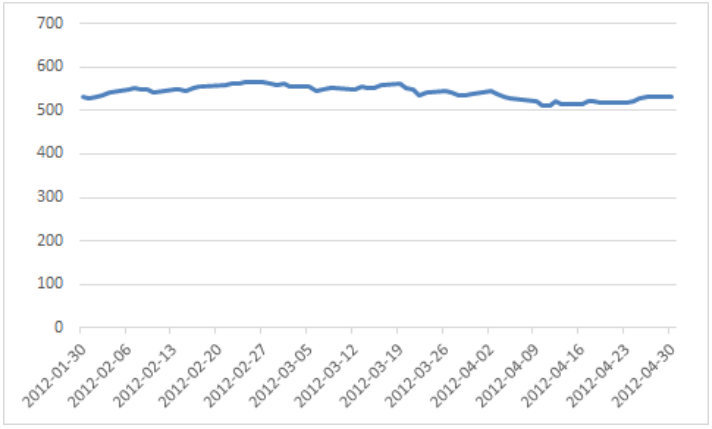

Figure 3.4: S&P 500 Energy (January 30, 2012 - April 30, 2012)

Chapter 4

Tracking News Stories

With the S&P 500 Energy index firmly selected as the index for further analysis, and the two significant time periods of January 22, 2008 to April 22, 2008 and July 14, 2008 to October 10, 2008 identified, the next step was to collect a database of news stories covering every day during this time period. Factiva (https://www.dowjones.com/products/factiva/) is a global news database offered by Dow Jones for the purpose of, among other features, enabling traders to keep tabs on news in the sector of their interest. Therefore, I was able to request all news articles from relevant sources across the time periods pertaining specifically to the energy sector.

Factiva draws from a significant number of sources, so it is challenging to narrow down the search criteria to only relevant news sources without excluding anything important. The entire list of criteria can be found in Appendix B.2 on page 131

The guiding principle in the search selection was to be sure to include data that most traders would have included in their sources and exclude any criteria that either a trader would be unlikely to use as well as any spurious data.

The text filter was included to ensure that any data used has a word count greater than 50. This choice of 50 is arbitrary but was put in place to ensure that only useful news stories would be included. Any news story less than 50 words was likely to not include regular English sentences and would probably not be useful.

The date selection was pre-defined based on the earlier analysis. The argument could be made that it might be a good idea to include some news stories from a few days prior to the time period in question as these stories could have framed some of the decisions made on the first day. However, I felt this was not a significant enough factor for consideration.

The source selection was the most difficult decision to make. Through trial and error, I found that there was a lot of duplication from various news sources, especially when considering newswires. The "Source" search criteria of "Major News and Business Sources: U.S." seemed to produce the most useful results. This identified news stories from major news stories across the United States

including highly relevant newspapers such as the Wall Street Journal and the New York times which seemed to be the most probable sources for most traders in this industry.

The criteria of Industry turned out to be a happy coincidence. Factiva did not have a perfect one to one correspondence between industry and an S&P 500 index. There are 17 industries to choose from but there are only 11 S&P 500 indices. For example, Factiva provides the option to search both Industrial Goods and Transportation/Logistics, which, according to the definition, would probably both fall under the S&P 500 Industrials index. However, Factiva did provide specific industries labeled as Health Care/Life Sciences, Technology, Telecommunications Services, Utilities, and Energy which could very easily be mapped to the S&P 500 Health Care, S&P 500 Information Technology, S&P 500 Telecommunication Services, S&P 500 Utilities, and S&P 500 Energy indices respectively. Therefore, specifying an industry of "Energy" was sufficient for analyzing news articles relevant to the energy sector.

The last modification to the search criteria I had to make was to purposely exclude the "Dow Jones Newswires – All sources". Including this criterion ballooned the search result count unnecessarily and most of that information could be found in the articles in the other sources anyway.

The other pieces of search criteria including "All Authors", "All Companies", "All Subjects", and "All Regions", and "English" were default settings in a Factiva search and I did not see any reason to consider changing it. I toyed with the idea of narrowing in on the North American region only but given the global nature of the U.S. economy, this did not seem like a worthwhile exercise.

The final result was 1817 news articles from the period January 22, 2008 to April 22, 2008, 2115 news articles from the period July 14, 2008 to October 10, 2008, and 3549 news articles from January 30, 2012 to April 30, 2012. This was a grand total of 5863 individual news stories across the three time periods. Since Factiva does not inherently support any mass downloading of their data, the news articles had to be download in HTML format and extracted manually. Therefore, these numbers seemed to strike a reasonable balance between manual effort required to obtain it and breadth of data to analyze.

Before the news articles could be analyzed, I decided it would be prudent to load the results into a database for reliable and repeated extraction and analysis. Therefore, I downloaded the results of Factiva into a set of HTML files. I wrote the script parseHTML.py which utilized the BeautifulSoup package to read the HTML files, pick out the headline, author(s), date, and article and inserted the results into a MySQL database.

Chapter 5

Applying Sentiment

The next challenge concerned labelling the sentiment of the news stories. Since the purpose of this research was to extract sentiment from news stories and use that for prediction, it is first required to label the news stories as either positive in sentiment or negative. I certainly could not personally label the articles myself as I would have risked significantly interfering with the results and would not be in accordance with standard and well-established research practices. Therefore, this approach was immediately rejected.

In accordance with previously cited work, another option is to employ an individual or set of individuals who can read the news articles and label the article as to whether they feel it expresses positive sentiment or negative sentiment for the Energy sector. Using a service such as Amazon Mechanical Turk might be a reasonable use-case for this. However, this option did not appear to be a desirable approach for several reasons. First, involving a human being runs the risk of influencing the subjects and therefore I would have to be sure to follow a double-blind approach to avoid this. Second, since sentiment is inherently subjective, I run the risk of my labels coming out purely based on the subjectivity of the person who labelled it. The stock market is awash with expert analysis with plenty of disagreement so what one person might consider to be positive, another may consider it to be negative. Third, and for similar reasons, the individual or individuals labelling the news articles may have varying degrees of expertise in the subject matter. Therefore, it is reasonable to expect a less experienced individual to label an article as positive while a more experienced person may label it to be negative. It would not be possible to pick an individual or set of individuals who is perfectly experienced to accurately make these decisions. Finally, an individual may feel pigeonholed by choosing a news article to be positive or negative when the article is ambiguous enough that it could probably be labelled either way. For these reasons, I did not choose to involve human assistance in labelling the articles.

Another way used in other previously cited work is to use the information already provided by the stock market. If the price of the S&P 500 Energy index closed higher than the previous trading

day, while it is not perfectly safe, it is still reasonably safe to assume that the news stories published on that day were probably positive. Similarly, a lower closing price from the previous day would indicate the news stories probably reflected negative sentiment. Therefore, if I were to use the stock price as a proxy for the sentiment of that day's news articles, I can eliminate the concerns from using human assistance. This comes with its own set of risks. First, it is possible that the stock price is more reflective of the overall investor mood that day and may not have anything at all to do with the news articles published that day. Negative news articles could be published on a day the S&P 500 finished higher than the previous day, or vice versa. Second, it is possible that a negative news article could be drowned out by all the positive news that day and therefore that article would surely be mislabeled. Third, a news article sentiment could be ambiguous, but this approach will forcibly apply a label. This will run the potential of poisoning the results with words that are not in line with the expected sentiment. Nevertheless, weighing the benefits and the risks between using humans to manually label the news article versus using a computer to label the news articles based on closing price alone, using an automated approach appeared to present the safest and most expedient approach.

To accommodate this approach, I added a field in my database to not only include the day's closing price, but to also include the difference in price from the previous day. A positive value would indicate the news from that day can be positive while a negative value would label the news articles from that day to be negative. By including this data, I was also able to quickly identify days where the mood shifted significantly from the previous day. For example, a price rise of $1 as the S&P 500 Energy index did on May 6, 2014 may be small enough to be considered noise. However, a price rise of $62.60 on October 13, 2008 or a drop of -$62.27 on October 15, 2008 represent significant investor mood shifts over the previous days. Since the data was already present, not only did I analyze sentiment from every day in the three time periods, I was also able to analyze sentiment specifically from days where there is significant price movement to see if this would translate into more accurate predictability.

Chapter 6

Evaluating Sentiment

6.1 Unigram and Bigram analysis

Up to this point, I had all the data properly lined up for analysis. I had the price of the S&P 500 Energy index over the three time periods I identified as the most relevant for analysis. I had news articles from a significantly diverse set of sources covering the three time periods. I had the change in price movement from the previous day so I could identify whether the day could be considered a positive news day or a negative news day. The final step would be to extract the markers that indicate a positive or negative news article out of 90% of the articles in each time period. This would be known as the "training set". Then, I would use the remaining 10% of articles to test the ability to predict whether the price would move positively or negatively given the markers found in the news story. This would be known as the "testing set".

For this analysis, I wrote the script read_db.py. It is largely based on the script written by Harrison Kinsley in his YouTube tutorial "NLTK with Python 3 for Natural Language Processing" and modified for the purposes of this research (Kinsley, 2017). In simplest terms, the script would iterate 100 times over each of the three time periods and, for each iteration, randomly assign a training set and a testing set. The choice of 100 is arbitrary but it appeared to be significant enough to prevent any variability in the data from excessively influencing the results in any direction.

For each time period, 100 times over, I would obtain a collection of positive news articles and negative news articles. Since the data was readily available, I simultaneously looked at only significant price movements in those time periods. "Significant" in this context included price movements of more than $10 in either direction. Then, for each iteration, I would take the entire collection of both positive and negative news articles and use these results to attempt to predict for a given day.

In the evaluate function, before any analysis could be done, I would first have to reduce the set of words I am working with down to a useful level. The first step stripped out any data associated

with the news article that was not the article itself. It is possible that one could include criteria such as the headline or the author for analysis, but this did not appear to be a worthwhile exercise as the content was significant enough in and of itself. Next, I had to tag each word in the document with its part-of-speech label. Then, I had to be selective about which words were worthwhile keeping on hand for analysis and which words could be considered useless. As is standard practice in the field, stop words were removed from all news articles. I also focused specifically on adjectives, adverbs, and verbs including their sub-types. The Penn Treebank defines the following word types as adjectives, adverbs, and verbs: JJ Adjective; JJR Adjective, comparative; JJS Adjective, superlative; RB Adverb; RBR Adverb, comparative; RBS Adverb, superlative; RP Particle; VB Verb, base form; VBD Verb, past tense; VBG Verb, gerund or present participle; VBN Verb, past participle; VBP Verb, non-3rd person singular present; and VBZ Verb, 3rd person singular present. Since the specific sub-types were not relevant, it was sufficient for my program to only check the first character of the label for J, R, or V. I did not expect nouns would be useful for this kind of analysis as they would be domain specific and therefore the results may not be safely inferred to other sectors. Other parts of speech were not included as they are less common and likely could not contribute much value above and beyond adjectives, adverbs, and verbs. The next step was to obtain a frequency distribution of the remaining words to see which words appeared most frequently and to weed out any outlier words that appear only once or twice. To ensure that only common words were included in this analysis, I selected only the top 33% of words.

By this point, I had a collection (word_features) of all the common words. This would be the set of words that I would use for analysis. Next, I would go back through the entire list of articles (documents) for this time period and identify whether or not a common word is found in the article or not. Therefore, each news article was now transformed into a set of the form:

```
{
        {
                word1: Boolean,
                word2: Boolean,
                ...
                wordN: Boolean
        },
        {positive or negative}
}
```

For example, if we assume that “dropped” and “rose” were both common words, a news article from a day where the price rose from the previous day might include a positive word “rose” but not include the negative word “dropped”. Conversely, a day where the price closed lower might contain a news article where the term “dropped” was found but “rose” was not found. Therefore, each news article would appear in the following form:

```
{
        {
                rose: True,
                dropped: False,
                ...
                wordN: Boolean
```

```
        },
        {positive}
}

{
        {
                rose: False,
                dropped: True,
                ...
                wordN: Boolean
        },
        {negative}
}
```

The entire set of articles was then randomized and then divided into a training set and a testing set. With two sets on hand, the next step was to employ a classifier to analyze the training set and attempt to make predictions about the news articles in the testing set. By extension, this will tell me how accurate the classifier can predict the movement in the closing price for a particular day if I were to choose to make this analysis more concrete. In the interest of keeping the scope of this project contained, I decided to simply present the results of the classifiers.

6.1.1 Classifiers

The python NLTK package includes a set of different classifiers to use. Rather than simply selecting one classifier, it was just as easy to use several classifiers. I could then either choose to respect only the best classifiers, or reject the worst classifiers, or I could keep the average of all the classifiers together and consider that to be my final classification attempt. Considering these options, I chose to use the average of all classifiers. Taking the average of all classifiers may translate into a better overall classification. It would also cause a classifier which happened to perform poorly on a particular run to be balanced by the other classifiers creating a more trustworthy result. Classifiers found in the NLTK package and the scikit-learn library are the Naïve Bayes Classifier, Multinomial Naïve Bayes Classifier, Bernoulli Naïve Bayes Classifier, Logistic Regression Classifier, Linear Support Vector Classification, Nu-Support Vector Classifier, and Stochastic Gradient Descent Classifier. The Linear Support Vector Classifier, Nu-Support Vector Classifier, and the Stochastic Gradient Descent Classifier proved to be unreliable and did not function as well as the others, so I did not include these classifiers in the final analysis. Thus, the macro average $\overline{C}$ can be expressed formally as:

$$\overline{C} = \frac{NaïveBayes + MultinomialNaïveBayes + BernoulliNaïveBayes + LogisticRegression}{4}$$

and over 100 iterations, this can be expressed formally as:

$$ClassificationAverage = \sum_{n=1}^{100} \frac{\overline{C_n}}{100}$$

This is the key value to be examined. To get both a sense of the standard deviation and a sense for how reliable the classifier may or may not be, both the maximum and minimum values are also retained:

$$ClassificationMaximum = \max(C_1, C_2, ..., C_{100})$$

$$ClassificationMinimum = \min(C_1, C_2, ..., C_{100})$$

6.1.2 Naïve Bayes Classifier

The Naïve Bayes Classifier is a classifier which determines the probability of a given label using the Bayes rule. Formally, this can be expressed as

$$P(label|features) = \frac{P(label)P(feature|label)}{P(features)}$$

such that the probability of each word (feature) is assigned positive or negative (label) and is based only on known probabilities and is irrespective of context. The sentiment of the entire article is then scaled up based on the sum probabilities of the words in the article. Formally, this can be expressed as

$$P(label|features) = \frac{P(label)P(feature_1|label)P(feature_2|label)...P(feature_n|label)}{\sum P(label)P(feature_1|label)P(feature_2|label)...P(feature_n|label)}$$

6.1.3 Multinomial Naïve Bayes Classifier

As pioneered by (Lewis and Gale, 1994) and explained by (Rennie et al., 2003), the Naïve Bayes Classifier could be generally useful. However, the Naïve Bayes Classifier does not take into account word occurrences. Thus, one might expect that less common words could wildly influence the final probabilities while very common words could no longer be considered useful to the determination of a label. Therefore, the probability of a given label for a document is also multiplied by the probability of a label for a given word that occurred. It would not be immediately clear whether or not this could be an issue or not as an uncommon word may be useful. For example, the term “jackpot” would likely not be a common word but its presence in an article might be a strong indicator of positive sentiment, irrespective of the rest of the text.

6.1.4 Bernoulli Naïve Bayes Classifier

As explained by (Rennie et al., 2003), the Bernoulli Naïve Bayes Classifier does not take into account word occurrences but instead retains only a boolean value to indicate the presence of a word in the document or not. Thus, not only is the usual Naïve Bayes value calculated, the final label for a document is also compared against both the occurrence of the words that appeared in the document and the non-occurrence of the words which did not appear in the document. This has a normalizing effect as short documents would be balanced against the large number of words that did not occur but could have.

6.1.5 Logistic Regression Classifier

If this paper were concerned with exactly how strongly a document is positive or negative, a simple logistic regression model would make sense for classification as the results could be mapped on a continuum from, say $0 \leq l \leq 1$where l would define the confidence level to which a sentiment could be applied. However since this paper is only concerned with the binary "positive" and "negative" labels, it is acceptable to forcibly classify a document on a sigmoid function. Formally, such a function is expressed as

$$S(x) = \frac{1}{1 + e^{-x}}$$

where x represents the true value of the classification attempt in a regression analysis and therefore $S(x)$ represents the probability of labelling an article "positive" or "negative" which is the classification I want to retain.

6.1.6 Linear Support Vector Classification

Linear Support Vector Classification employs the standard Support Vector Model and classifies the results on a simple linear division. According to the scikit-learn library, the LinearSVC is merely a specialized case and more efficiently implemented version of the generic Support Vector Classifier. The primal problem is expressed formally as

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1} \max(0, y_i(w^T \phi(x_i) + b)),$$

Since the results produced by this classifier appeared to be clearly unreliable, this classifier was ultimately not used.

6.1.7 Nu-Support Vector Classifier

By the early 2000s, inherent limitations in traditional SVM algorithms necessitated the need to develop more advanced algorithms that would correct these limitations. As described by (Schölkopf et al., 2000), and as described in the documentation, the Nu-Support Vector Classifier is mathematically equivalent to the aforementioned Linear Support Vector Classification except added flexibility is added by employing a parameter $\nu \in (0, 1]$ rather than using the parameter C which employs any positive value. Unlike a standard SVM employing the parameter C, Nu-Support Vector Classification necessarily sets an upper bound on the fraction of margin of errors and a lower bound on the fraction of support vectors. This allows for more tolerance of potential outliers from a noisy data set. Since the results produced by this classifier appeared to be clearly unreliable, this classifier was ultimately not used.

6.1.8 Stochastic Gradient Descent Classifier

The Stochastic Gradient Descent Classifier is explained in the documentation as merely a Linear Classifier with stochastic gradient descent training. This is an iterative approach where the gradient of the loss is updated regularly as the classifier is trained. Since the results produced by this classifier appeared to be clearly unreliable, this classifier was ultimately not used.

6.2 Artificial Neural Network analysis

The other approach deviated from n-gram classifiers and followed the more holistic approach of employing a unidirectional LSTM artificial recurrent neural network (ANN). When employing an ANN, I did not divide the source text into individual words or a set of words and use that for analysis. Instead, I chose to analyze the text as a whole and see how well this works as a sentiment classifier. Most of the code is forked from a project intended to predict sentiment in reviews of Netflix movies (Opperman, 2019). In principle the idea is still the same. Clean the text to remove punctuation characters and other characters that would not be of any use (clean_file.pyA.5). Then divide the text into a training and test set (tf_record_writer.pyA.6). Finally, apply an artificial neural network against the training text to try to improve classification (train.pyA.7). The output would display the results of each iteration or "epoch". Three values would be displayed: Training loss, Training Accuracy, and Test Accuracy. The Training loss indicates the relationship between the training set and the test set. The training accuracy indicates the ANN's ability to accurately predict sentiment on the training set. The test accuracy indicates the ANN's ability to accurately predict sentiment on the test set. In other words, how the ANN is able to perform against text it

has not "seen" before, and therefore has not trained against. A typical output would look similar to the following:

```
epoch_nr: 0, train_loss: 0.654, train_acc: 0.629, test_acc: 0.737
epoch_nr: 1, train_loss: 0.451, train_acc: 0.809, test_acc: 0.753
epoch_nr: 2, train_loss: 0.294, train_acc: 0.889, test_acc: 0.762
epoch_nr: 3, train_loss: 0.205, train_acc: 0.930, test_acc: 0.740
epoch_nr: 4, train_loss: 0.141, train_acc: 0.951, test_acc: 0.754
epoch_nr: 5, train_loss: 0.108, train_acc: 0.965, test_acc: 0.743
epoch_nr: 6, train_loss: 0.085, train_acc: 0.973, test_acc: 0.723
epoch_nr: 7, train_loss: 0.069, train_acc: 0.977, test_acc: 0.738
epoch_nr: 8, train_loss: 0.055, train_acc: 0.984, test_acc: 0.725
epoch_nr: 9, train_loss: 0.048, train_acc: 0.986, test_acc: 0.727
epoch_nr: 10, train_loss: 0.044, train_acc: 0.987, test_acc: 0.723
```

This example shows the training loss falling indicating the ANN is approaching its best classification ability given the training set. The training accuracy remains high at near 0.9 indicating the ANN is functioning well against its own training set. The test accuracy remains near 0.7 which for my purposes would indicate improvement over random chance. However, it should also be noted that improvement over each epoch on the test set is not improving indicating this example is overfitting its training data. As an added window into how the ANN is functioning, train.py also prints some random samples at each epoch to demonstrate its perceived sentiment of the text. The following demonstrates some sample output:

```
  Review: "new yorkers who are already stuck with some of the nation s highest electric bills will have to cough
up 4 to 6 more next month under a rate hike granted to con ed yesterday the exact amount of the increase will
vary according to how much power customers use but state public service commission officials said that con ed s
residential customers in new york city and westchester will pay an average of 4 7 percent more and that s just
the beginning rising oil and natural gas prices which are not regulated by the commission will likely add a few
more dollars to bills con ed was disappointed with the ruling which will bring it an extra 425 million over the
next year about a third of what it wanted we can not meet expectations for maintaining and improving the system
without greater investments the company said in a statement but the company is still getting a revenue boost that
would be the envy of any other business with the cash it collects for maintaining wires and other infrastructure
jumping about 12 percent that led company critics to say the psc is letting con ed take too much from its customers
pockets without demanding any reforms in return they should be insisting on concessions for that money they lost
the opportunity to do that today said assemblyman michael gianaris d queens a big part of the increase will help
con ed recover 1 1 billion it has already spent in recent years on transformers cable and other transmission equipment
that money was beyond the amounts allowed by the psc in its last rate ruling for con ed three years ago some
public service commission members expressed skepticism about con ed s infrastructure push and said they were not convinced
the money was properly spent i don t feel a great sense of assurance that we have accurate cost estimates from
con ed said commissioner cheryl buley the psc lacks information about the company s spending after 15 years of ignoring
a state law requiring it to regularly audit con ed s capital spending the psc now plans to renew the audits and
ordered con ed to set aside 250 million for possible customer refunds if it decides the money wasn t properly spent
con ed is already preparing its next hike request"

  pos. sentiment: 0.81 %
  neg. sentiment: 0.19 %
```

Chapter 7

Results

7.1 Unigrams

The first task was to analyze predictability over the three time periods with only unigrams of specific parts of speech. A summary of all results can be found in Table 7.1 on the following page. For the time period from January 22, 2008 to April 22, 2008, over 100 iterations, my program was able to successfully predict sentiment of a given news article, on average, 85.22% of the time. The best classification attempt was successful 88.13% of the time and the worst classification attempt was 82.05%. Further inspection of these results revealed that the Linear Support Vector classifier and the Logistic Regression classifier were regularly able to make predictions above 98%. The Multinomial Naïve Bayes classifier was typically in the 80% to 90% range. The Naïve Bayes classifier and the Bernouilli classifiers regularly classified with 60% to 70% accuracy.

As referenced above, because the data was readily available, I immediately ran a second analysis over only the significant days where the price moved above or below $10 from the previous day. Or in other words, days where the closing price did not change much from the previous day were entirely excluded from this analysis. From the period of January 22, 2008 and April 22, 2008, there were 13 days where the index rose more than $10 and 8 days where the index fell more than $10. Using the average of all classifiers across 100 iterations, this produced a classification accuracy average of 59.32%. The best classification run was 72.50% and the worst classification run was 43.50%.

For the time period from July 14, 2008 to October 10, 2008, over 100 iterations, my program was able to successfully predict sentiment of a given news article, on average, 59.44% of the time. The best classification attempt was successful 67.88% of the time and the worst classification attempt was 52.42%. Further inspection of these results revealed that over this data set, there was no classifier that was well suited to classify the news articles. They all fell regularly inside the 50% to 70% accuracy range.

Time Period	Unigrams Average	Unigrams Maximum	Unigrams Minimum
January 22, 2008 to April 22, 2008 (Positive) (All Days)	85.22%	88.13%	82.04%
January 22, 2008 to April 22, 2008 (Positive) (Extreme Days)	59.32%	72.50%	43.50%
July 14, 2008 to October 10, 2008 (Negative) (All Days)	59.44%	67.88%	52.42%
July 14, 2008 to October 10, 2008 (Negative) (Extreme Days)	57.76%	67.35%	48.24%
January 30, 2012 to April 30, 2012 (Average) (All Days)	56.47%	62.78%	50.87%
January 30, 2012 to April 30, 2012 (Average) (Extreme Days)	57.03%	85.00%	15.00%

Table 7.1: Summary of all results for unigrams

As referenced above, because the data was readily available, I immediately ran a second analysis over only the significant days where the price moved greater or less than $10 from the previous day. Or in other words, days where the closing price did not change much from the previous day were entirely excluded from this analysis. From the period of July 14, 2008 and October 10, 2008, there were 14 days where the index rose more than $10 and 18 days where the index fell more than $10. Using the average of all classifiers across 100 iterations, this produced a classification accuracy average of 57.76%. The best classification run was 67.35% and the worst classification run was 48.24%.

For the time period from January 30, 2012 to April 30, 2012, over 100 iterations, my program was able to successfully predict sentiment of a given news article, on average, 56.47% of the time. The best classification attempt was successful 62.78% of the time and the worst classification attempt was 50.87%. Further inspection of these results revealed that over this data set, there was no classifier that was well suited to classify the news articles. They all fell regularly inside the 50% to 70% accuracy range.

As referenced above, because the data was readily available, I immediately ran a second analysis over only the significant days where the price moved greater or less than $10 from the previous day. Or in other words, days where the closing price did not change much from the previous day were entirely excluded from this analysis. Of course, given that it was already known that there was no significant price changes over this time period, it is not at all surprising that from the period of January 30, 2012 and April 30, 2012, there was 1 day where the index rose more than $10 and 2 days where the index fell more than $10. Since there were only 3 days to work with, the results cannot represent anything useful and is therefore considered spurious.

In addendum, I can be reasonably confident that my analysis is producing accurate results. The NLTK package provides a show_most_informative_features function. As is, the function will determine the most informative features and print them. In other words, it will analyze the data to determine which features appeared to be the most influential in determining sentiment. Unfortunately, the function did not provide any ability to store the data printed so I had to rewrite the function to put the data into a string that can be stored rather than printing the results to the screen. With the data stored in my database, I can go back and verify that the data found appears useful or not. An example can be found in Table 7.2. In this example, one run of the analysis between January 22, 2008 and April 22, 2008 determined that the word “acquiring” was more useful at predicting positive sentiment over negative sentiment at a ratio of 7.2 to 1.0. In the same run, the word “signaled” was more useful in predicting negative sentiment over positive sentiment at a ratio of 5.5 to 1.0. A review of news stories during this time confirm these words to be helpful. The word “acquiring” appeared in 15 news stories over 15 different days and only one of those 15 days turned out to be a day where the S&P 500 Energy index closed lower. The word “signaled”

appeared in 18 news stories over 13 days and only four of those 13 days turned out to be a day where the S&P 500 Energy index closed higher. A further sampling of the news stories verifies the sentiment prediction is accurate.

For example, consider the following story using the word “acquiring” from January 28, 2008 when the index rose $9.12:

> **Headline**: Gazprom Drills Deeper Into Europe --- U.K. Becomes Foothold in Entry to Consumer Market; Mistrust Lingers
>
> **Author**: By Guy Chazan
>
> **Date**: January 28, 2008
>
> **Change in share price from the previous day:** +$9.12 (positive sentiment)
>
> **Body**: [. . .] Gazprom is small, but they’re acquiring new customers every month at increasing speed," says Binoy Dharsi, an analyst at Datamonitor [. . .]

Or consider the following story using the word “acquiring” from February 28, 2008 when the index rose $8.14:

> **Headline**: Pasco in bid to buy utilities
>
> **Author**: Chuin-Wei Yap, Times Staff Writer
>
> **Date**: February 28, 2008
>
> **Change in share price from the previous day:** +$8.14 (positive sentiment)
>
> **Body**: [. . .] Commissioners voted Wednesday to join a multicounty agency that specializes in acquiring private water companies. [. . .]

Conversely, consider the following story using the word “signaled” from February 6, 2008 when the index dropped $8.21:

> **Headline**: National Oilwell Varco’s net rises, backlog jumps
>
> **Author**: Steve Gelsi, MarketWatch
>
> **Date**: February 6, 2008
>
> **Change in share price from the previous day:** -$8.21 (negative sentiment)
>
> **Body**: [. . .] However, the Houston-based company (NOV, US) also signaled it’s not immune from weakness in the North American market. Shares of National Oilwell Varco fell 6.7% to close at $57.85, a six-month low. [. . .]

Or consider the following story from March 7, 2008 when the index dropped $11.95:

> **Headline**: Coal sees red over green As environmental politics take hold in Washington, Peabody lines up lobbyists for counterattack.
>
> **Author**: By Deirdre Shesgreen Post-Dispatch Washington Bureau

Date: March 7, 2008

Change in share price from the previous day: -$11.95 (negative sentiment)

Body: [. . .] All three leading presidential candidates have signaled they would take a stronger approach to cutting greenhouse gas emissions than the Bush administration has. [. . .]

WORD	SENTIMENT	RATIO
contains	Positive over Negative	9.2 : 1.0
arrived	Positive over Negative	8.7 : 1.0
convertible	Positive over Negative	8.2 : 1.0
closes	Positive over Negative	7.2 : 1.0
acquiring	Positive over Negative	7.2 : 1.0
repurchase	Positive over Negative	7.1 : 1.0
spur	Positive over Negative	6.8 : 1.0
distribute	Positive over Negative	6.8 : 1.0
native	Positive over Negative	6.4 : 1.0
lining	Positive over Negative	6.1 : 1.0
atmosphere	Positive over Negative	5.8 : 1.0
sells	Positive over Negative	5.6 : 1.0
discussing	Negative over Positive	5.5 : 1.0
signaled	Negative over Positive	5.5 : 1.0
advising	Negative over Positive	5.5 : 1.0

Table 7.2: Most Informative Features. Sample run

7.2 Bigrams

It is possible that analyzing bigrams instead of specific parts of speech could reveal more interesting results. For example, the words “not” and “rise” might trick the analyzer into picking a positive sentiment given the presence of the word “rise”. However, if the phrase is identified as “not rise”, this indicates a very strong negative sentiment. Therefore, I decided to take the existing data and follow a similar process but analyzed bigram data instead. The results of this analysis can be found in Table 7.3 on the next page.

Time Period	Bigrams Average	Bigrams Maximum	Bigrams Minimum
January 22, 2008 to April 22, 2008 (Positive) (All Days)	53.04%	54.92%	51.19%
January 22, 2008 to April 22, 2008 (Positive) (Extreme Days)	53.19%	57.00%	50%
July 14, 2008 to October 10, 2008 (Negative) (All Days)	51.34%	53.33%	49.39%
July 14, 2008 to October 10, 2008 (Negative) (Extreme Days)	51.70%	53.82%	47.06%
January 30, 2012 to April 30, 2012 (Average) (All Days)	52.16%	53.39%	50.61%
January 30, 2012 to April 30, 2012 (Average) (Extreme Days)	49.07%	55.00%	42.50%

Table 7.3: Summary of all results for bigrams

In order to parse the data to produce the bigrams, I wrote a second function called evaluteBigrams and started with the same process as before. In the evaluateBigrams function, before any analysis could be done, I would first have to reduce the set of words I am working with down to a useful level. The first step stripped out any data associated with the news article that was not the article itself. It is possible that one could include criteria such as the headline or the author for analysis, but this did not appear to be a worthwhile exercise as the content was significant enough in and of itself. Because this was a bigram analysis, I did not remove any stop words or reduce the set of words to specific parts of speech. It might have been possible to look at specific bigrams that follow a certain pattern such as verb followed by adjective. For example, a phrase such as "rose quickly" or "dropped suddenly" might help make a strong indication of positive sentiment or negative sentiment. However, one would have to be cautious as restricting the bigrams to specific parts of speech might reduce the set of data to be too narrow for useful analysis.

In order to parse the corpus into bigrams, the NLTK provides a class called "BigramCollocationFinder" which can be used to obtain a list of all bigrams found. I then used the ngram_fd method to produce a frequency distribution of all bigrams found. Again, I retained only the top 33% of all bigrams found as many bigrams may appear only once in the entire corpus and would not be useful for analysis anyway. I then re-evaluated all news articles found and marked True or False if the commonly used bigram was found in the article or not. The articles were then divided into a training set and a testing set in a 90% training versus 10% split.

The naïve bayes classifier was able to use the training and testing sets as is, but the other classifiers did not. Since this exercise was more exploratory and not necessarily the primary focus of this study, I did not elect to include additional classifiers and use a voting approach like I did with the parts of speech evaluation.

For the time period from January 22, 2008 to April 22, 2008, over 100 iterations, my program was able to successfully predict sentiment of a given news article, on average, 53.04% of the time. The best classification attempt was successful 54.93% of the time and the worst classification attempt was 51.19%.

As referenced above, because the data was readily available, I immediately ran a second analysis over only the significant days where the price moved greater or less than $10 from the previous day. Or in other words, days where the closing price did not change much from the previous day were entirely excluded from this analysis. From the period of January 22, 2008 and April 22, 2008, there were 13 days where the index rose more than $10 and 8 days where the index fell more than $10. Using the average of all classifiers across 100 iterations, this produced a classification accuracy average of 53.20%. The best classification run was 57% and the worst classification run was 50%.

For the time period from July 14, 2008 to October 10, 2008, over 100 iterations, my program

was able to successfully predict sentiment of a given news article, on average, 51.34% of the time. The best classification attempt was successful 53.33% of the time and the worst classification attempt was 49.39%.

As referenced above, because the data was readily available, I immediately ran a second analysis over only the significant days where the price moved greater or less than $10 from the previous day. Or in other words, days where the closing price did not change much from the previous day were entirely excluded from this analysis. From the period of July 14, 2008 and October 10, 2008, there were 14 days where the index rose more than $10 and 18 days where the index fell more than $10. Using the average of all classifiers across 100 iterations, this produced a classification accuracy average of 51.70%. The best classification run was 53.82% and the worst classification run was 47.06%.

For the time period from January 30, 2012 to April 30, 2012, over 100 iterations, my program was able to successfully predict sentiment of a given news article, on average, 52.16% of the time. The best classification attempt was successful 53.39% of the time and the worst classification attempt was 50.61%.

As referenced above, because the data was readily available, I immediately ran a second analysis over only the significant days where the price moved greater or less than $10 from the previous day. Or in other words, days where the closing price did not change much from the previous day were entirely excluded from this analysis. Of course, given that it was already known that there was no significant price changes over this time period, it is not at all surprising that from the period of January 30, 2012 and April 30, 2012, there was 1 day where the index rose more than $10 and 2 days where the index fell more than $10. Since there were only 3 days to work with, the results cannot represent anything useful and is therefore considered spurious.

7.3 Artificial Neural Network - Full stories

After exploring various n-gram avenues, an alternative method for prediction that was tested was an ANN. It is possible that a more wholistic overview of the text could produce more reliable results than reviewing individual unigrams or bigrams. Often, sentiment in English is more readily conveyed through complete sentence structure rather than individual words or word phrases. Therefore, the true sentiment of the text may be "hidden" in the sentences which could be more easily extracted through an ANN. To test this theory, it would be necessary to use the same time periods and the same text so comparisons can be made. However, one should note that this method can produce positive or negative sentiment values for individual news stories, while the unigram and bigram approach produces positive or negative sentiment values in aggregate over the three month time period. Therefore, while the ANN approach provides detailed sentiment for individ-

ual news stories, it is not possible to directly compare those individual news stories back to the unigram or bigram approaches.

The text covered continued to follow the aforementioned three-month time periods of January 22, 2008 to April 22, 2008, July 14, 2008 to October 10, 2008, and January 30, 2012 to April 30, 2012. For each of the three time periods, the text was divided into a training set and a test set and the ANN was run against the text. In all cases, iterating over the default 10 iterations did not appear to demonstrate stable or useful results so I expanded to 20 iterations. While it might be reasonable to expand to a larger number, the program takes some time to run and moving beyond 20 iterations did not appear as though it would produce any results more useful than those already produced.

For the time period from January 22, 2008 to April 22, 2008, the results can be found in Figure 7.1

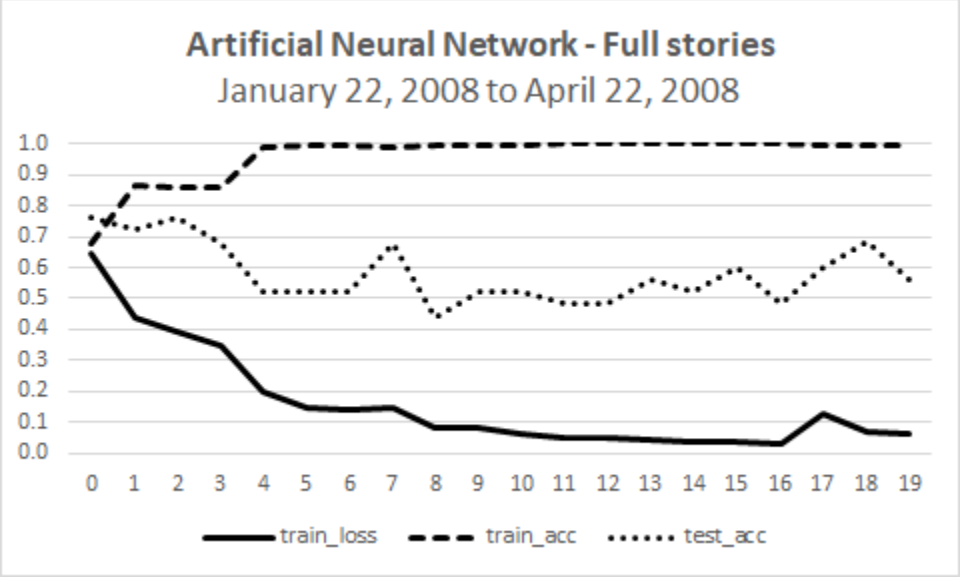

Figure 7.1: Artificial Neural Network - Full Stories from January 22, 2008 to April 22, 2008

Early in the process, it is clear the training loss is falling quickly, and the training accuracy very quickly rises. However, after stabilization, the test accuracy reaches slightly over .5 and does not deviate much. A review of the sample output demonstrates some semblance of correctness, but it also demonstrates a bit of "confusion". The results are either very confidently labelled or labelled with very little confidence. Sometimes the sentiment matches the expected sentiment and sometimes it does not. For example, the following news story was labelled very confidently, it matches the expected sentiment for that day, and given the context, this label is probably correct:

> **Headline**: Corporate News: Eni Signs Qatar Deal In Latest Expansion
> **Author**: By Spencer Swartz and Liam Moloney
> **Date**: April 21, 2008

Change in price from previous day: +$5.67 (positive sentiment)

Body: [. . .] ROME -- Italian oil-and-gas company Eni SpA continued its recent string of expanding ties with state-run oil companies, signing a deal to pursue multibillion-dollar oil and natural-gas projects with Qatar [. . .] Qatar's oil minister Abdullah bin Hamad Al-Attiyah welcomed Eni's return to Qatar and said both sides would invest billions in projects [. . .]

Epoch 17: train_loss: 0.126, train_acc: 0.992, test_acc: 0.600, pos. sentiment: 0.96 %, neg. sentiment: 0.04 %

Epoch 18: train_loss: 0.067, train_acc: 0.997, test_acc: 0.681, pos. sentiment: 0.99 %, neg. sentiment: 0.01 %

The following news story was labelled very confidently but then almost inversely in two separate epochs:

Headline: Business Brief -- Electric Power Development: Plan Calls for Considering Purchase of Mine Stakes

Author: UNKNOWN

Date: April 1, 2008

Change in share price from the previous day: +$11.46 (positive sentiment)

Body: [. . .] Electric Power Development Co. released its five-year business plan as a dispute with a U.K. investment fund over the company's management continued. [. . .] The utility aims to build a global coal-supply chain that would secure stable supply and prices, in response to tightening in the global market, said President Yoshihiko Nakagaki. "We have many long-term contracts, but this does not promise stable supplies anymore," he said. [. . .]

Epoch 17: train_loss: 0.126, train_acc: 0.992, test_acc: 0.600, pos. sentiment: 0.00 %, neg. sentiment: 1.00 %

Epoch 18: train_loss: 0.067, train_acc: 0.997, test_acc: 0.681, pos. sentiment: 0.68 %, neg. sentiment: 0.32 %

For the time period from July 14, 2008 to October 10, 2008,the results can be found in Figure 7.2

Once again, early in the process, it is clear the training loss is falling quickly, and the training accuracy very quickly rises. However, after stabilization, the test accuracy reaches slightly over .5 and does not deviate much. A review of the sample output demonstrates similar "confusion" as above. For example, the following news story is consistently and confidently labelled negatively but reviewing the story indicates the true sentiment might be a bit more nebulous:

Headline: WINDY CITY - BUILDING, BRIDGE 'FANS' CAN POWER NYC: MIKE; BREEZY DOES IT: HIZZONER

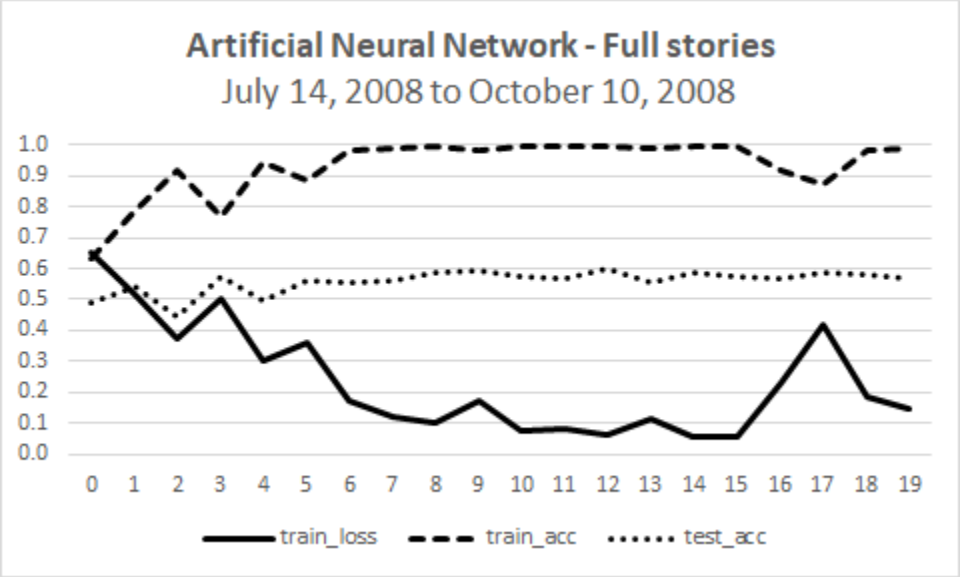

Figure 7.2: Artificial Neural Network - Full Stories from July 14, 2008 to October 10, 2008

Author: DAVID SEIFMAN

Date: August 20, 2008

Change in share price from the previous day: +$14.99 (positive sentiment)

Body: [. . .] New York would claim title as the genuine "Windy City" under a dramatic proposal by Mayor Bloomberg yesterday to develop wind turbines atop the Big Apple's bridges and skyscrapers. But that's not all. The mayor also tossed out the possibility of building wind farms way out in the Atlantic Ocean, miles from shore, that he said could generate roughly twice the energy of similar land-based facilities and supply 10 percent of the city's electricity needs within a decade. [. . .] "This isn't a wild idea at all," said Dale Jamieson, director of environmental studies at NYU. [. . .] Senate Major Leader Harry Reid (D-Nev.), who sponsored the summit, said he's taken with Bloomberg's novel idea."I . . . may steal it from you," he said.

Epoch 4: train_loss: 0.304, train_acc: 0.940, test_acc: 0.498, pos. sentiment: 0.00 %, neg. sentiment: 1.00 %

Epoch 5: train_loss: 0.360, train_acc: 0.887, test_acc: 0.560, pos. sentiment: 0.00 %, neg. sentiment: 1.00 %

For the time period from January 30, 2012 to April 30, 2012, the results can be found in Figure 7.3

As in the previous two runs, it is clear the training loss is falling quickly, and the training accuracy very quickly rises. However, after stabilization, the test accuracy reaches slightly over .5 and does not deviate much. A review of the sample output demonstrates similar "confusion" as above. It is possible to observe improvement over time of the following news story, despite the

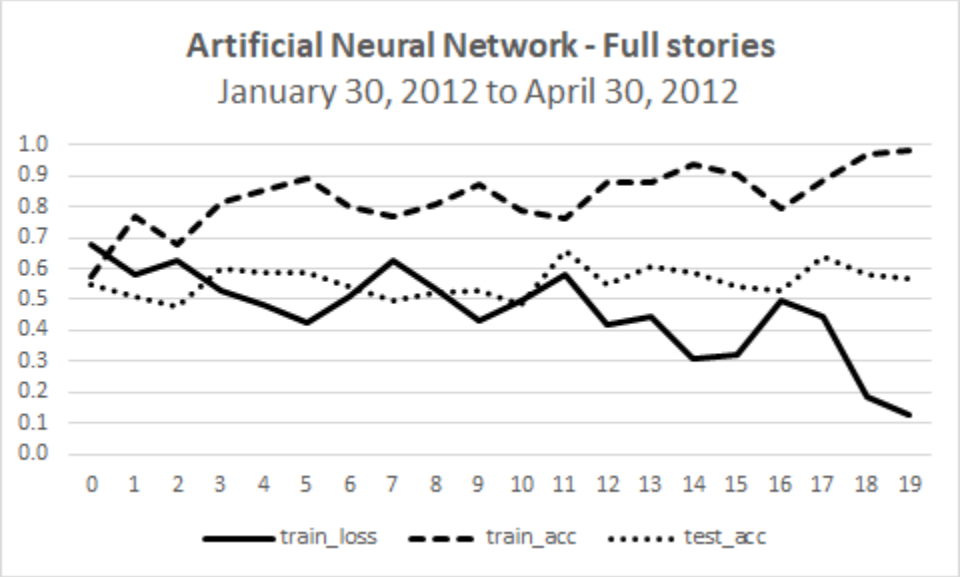

Figure 7.3: Artificial Neural Network - Full Stories from January 30, 2012 to April 30, 2012

fact it was a postive day for the sector:

> **Headline**: Corporate News: Chevron Field Leaking Again --- Second Incident of Seepage Offshore Brazil Likely to Intensify Political Backlash
>
> **Author**: By Daniel Gilbert
>
> **Date**: March 16, 2012
>
> **Change in share price from the previous day:** +$6.38 (positive sentiment)
>
> **Body**: [. . .] Chevron Corp. said Thursday that oil was seeping from the ocean floor near its operations off the coast of Brazil, the second such leak in recent months and a significant setback for the oil giant in a region crucial for its growth. [. . .] Chevron stands to take a financial hit from shutting in the oil production [. . .] But on Tuesday, Chevron Chief Executive John Watson told analysts in New York that the company's future in Brazil "remains to be seen." [. . .]
>
> **Epoch 0**: train_loss: 0.679, train_acc: 0.577, test_acc: 0.551, pos. sentiment: 0.41 %, neg. sentiment: 0.59 %
>
> **Epoch 14**: train_loss: 0.308, train_acc: 0.936, test_acc: 0.590, pos. sentiment: 0.05 %, neg. sentiment: 0.95 %
>
> **Epoch 17**: train_loss: 0.443, train_acc: 0.883, test_acc: 0.640, pos. sentiment: 0.00 %, neg. sentiment: 1.00 %

However, there is still plenty of erratic classifications as well which can be seen in the following news story:

Headline: India Grapples With Soaring Energy Costs; Government Aims to Spur Domestic Production, Minister Says, to Reduce Nation"s Dependence on Imported Oil and Gas

Author: By Amol Sharma and Matt Murray

Date: April 10, 2012

Change in share price from the previous day: -$10.10 (negative sentiment)

Body: [. . .] Indian oil and gas minister Jaipal Reddy said the nation's soaring energy import bill is becoming a growing economic challenge, but he expressed hope that prices will eventually moderate and that the government can provide incentives to spur more domestic production. [. . .] Economists say high energy costs have inflationary effects that could hold India back from its goal of consistent 9% gross-domestic-product growth. [. . .] Mr. Reddy said geopolitical tensions over Iran, which supplies about 12% of India's oil, are buoying prices and are a major worry for New Delhi [. . .]

Epoch 5: train_loss: 0.425, train_acc: 0.893, test_acc: 0.589, pos. sentiment: 0.17 %, neg. sentiment: 0.83 %

Epoch 15: train_loss: 0.319, train_acc: 0.903, test_acc: 0.540, pos. sentiment: 0.53 %, neg. sentiment: 0.47 %

7.4 Artificial Neural Network - Headlines only

In an attempt to mitigate the limitations observed in applying an ANN to the entire text, it would be reasonable to suggest that perhaps the headlines could produce results that are more manageable. Since the headlines are likely to contain relevant information which would influence investor sentiment and since the headlines are much shorter and easier for an ANN to work with, this had the potential to be a useful exercise. Indeed, since the text analyzed was considerably shorter, the runtime was much quicker, and it was far more reasonable to run through more epochs generating more reliable results.

The text covered continued to follow the aforementioned three-month time periods of January 22, 2008 to April 22, 2008, July 14, 2008 to October 10, 2008, and January 30, 2012 to April 30, 2012. For each of the three time periods, the text was divided into a training set and a test set and the ANN was run against the text.

For the time period from January 22, 2008 to April 22, 2008, the results can be found in Figure 7.4

Early in the process, it is clear the training loss is falling quickly, and the training accuracy very quickly rises. It is clear that the test is suffering from overfitting early in the process achieving high training accuracy and relatively high test accuracy. However, as more data is introduced in

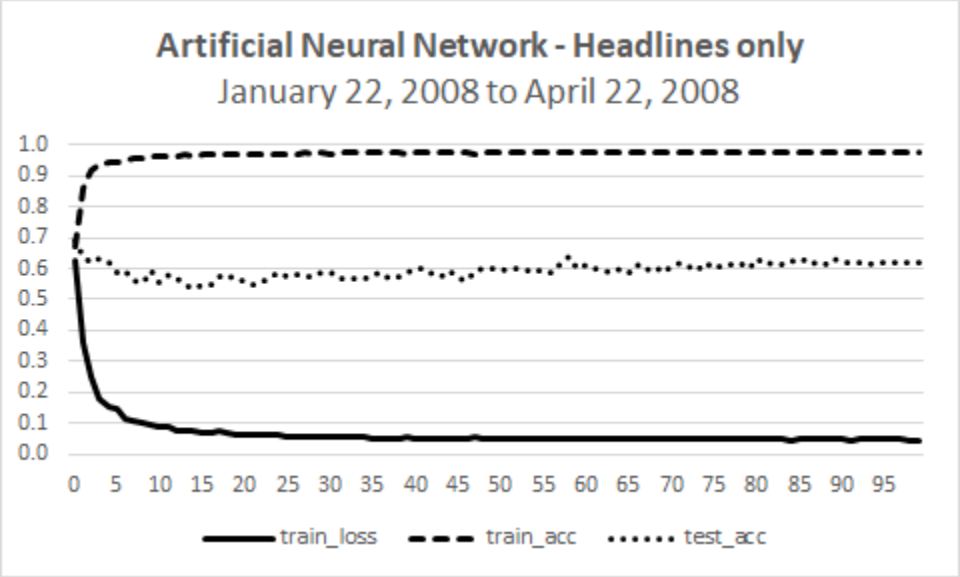

Figure 7.4: Artificial Neural Network - Headlines only from January 22, 2008 to April 22, 2008

future epochs, the test begins to coalesce around its true predictive value. In this case, the test accuracy falls slightly to below .600 and rises again until it coalesces around .621. A review of the sample output demonstrates some semblance of correctness, but it also demonstrates a bit of "confusion". The results are either very confidently labelled or labelled with very little confidence. For example, the following news story was labelled very confidently. While it is usually assigned a positive sentiment, and given the context, this label is probably correct, it will occasionally choose an incorrect sentiment:

Headline: OPEC stands fast, will adjust supply over long-term; Cartel plans $160 billion investment to increase capacity in four years

Author: Moming Zhou, MarketWatch

Date: April 22, 2008

Change in share price from the previous day: +$1.17 (positive sentiment)

Epoch 5: train_loss: 0.141, train_acc: 0.945, test_acc: 0.580, pos. sentiment: 0.53 %, neg. sentiment: 0.47 %

Epoch 12: train_loss: train_loss: 0.080, train_acc: 0.965, test_acc: 0.606, pos. sentiment: 0.89 %, neg. sentiment: 0.11 %

Epoch 41: train_loss: 0.051, train_acc: 0.974, test_acc: 0.599, pos. sentiment: 0.96 %, neg. sentiment: 0.04 %

Epoch 63: train_loss: 0.048, train_acc: 0.975, test_acc: 0.584, pos. sentiment: 0.02 %, neg. sentiment: 0.98 %

Epoch 98: train_loss: 0.045, train_acc: 0.975, test_acc: 0.621, pos. sentiment:

0.95 %, neg. sentiment: 0.05 %

Nevertheless, it does appear as though the sample headlines are usually correct as time goes on. This is further reinforced as samples from early epochs are generally neutral but later epochs are more definitive:

```
epoch_nr: 0, train_loss: 0.617, train_acc: 0.676, test_acc: 0.696
Test Samples:
  Review: "nicor s quarterly profit slips as revenue rises cautions on 2008 outlook sees weaker results across segments"
  pos. sentiment: 0.51 %
  neg. sentiment: 0.49 %

  Review: "upgrades helped contain fla blackout lessons learned from 03 northeast outage"
  pos. sentiment: 0.46 %
  neg. sentiment: 0.54 %

  Review: "iceland has power to burn the tiny island nation can teach the united states valuable lessons about energy
policy"
  pos. sentiment: 0.53 %
  neg. sentiment: 0.47 %

  Review: "ethanol output may damp gasoline profits"
  pos. sentiment: 0.62 %
  neg. sentiment: 0.38 %

  Review: "given hasbro mattel nestle novartis tyson"
  pos. sentiment: 0.42 %
  neg. sentiment: 0.58 %

epoch_nr: 1, train_loss: 0.346, train_acc: 0.868, test_acc: 0.629
Test Samples:
  Review: "politics economics algeria oil tie up thrives is sonatrach deal with statoilhydro a lesson to west"
  pos. sentiment: 0.86 %
  neg. sentiment: 0.14 %

  Review: "would be nuclear nations a risk global community needs to train follow up on countries that are novices
in generating power from atomic fission"
  pos. sentiment: 0.08 %
  neg. sentiment: 0.92 %

  Review: "international business not all share power merger view e on s chief executive says protectionism may slow
utility deals"
  pos. sentiment: 0.96 %
  neg. sentiment: 0.04 %

  Review: "dynegy cuts quarterly loss as revenue jumps"
  pos. sentiment: 0.95 %
  neg. sentiment: 0.05 %

  Review: "noted"
  pos. sentiment: 1.00 %
  neg. sentiment: 0.00 %
...
epoch_nr: 98, train_loss: 0.045, train_acc: 0.975, test_acc: 0.621
Test Samples:
  Review: "flight 93 memorial effort gains over 900 acres"
  pos. sentiment: 1.00 %
  neg. sentiment: 0.00 %

  Review: "would be nuclear nations a risk global community needs to train follow up on countries that are novices
in generating power from atomic fission"
  pos. sentiment: 0.04 %
  neg. sentiment: 0.96 %

  Review: "opec stands fast will adjust supply over long term cartel plans 160 billion investment to increase capacity
in four years"
  pos. sentiment: 0.95 %
  neg. sentiment: 0.05 %

  Review: "iran s nuclear threat"
  pos. sentiment: 1.00 %
```

```
  neg. sentiment: 0.00 %

  Review: "eurolinks daily view how to lose 7 2 billion a trader s tale"
  pos. sentiment: 0.67 %
  neg. sentiment: 0.33 %


epoch_nr: 99, train_loss: 0.045, train_acc: 0.976, test_acc: 0.621
Test Samples:
  Review: "business brief total sa share of saudi venture is shifted to partners"
  pos. sentiment: 0.50 %
  neg. sentiment: 0.50 %

  Review: "sempra energy doubles profit"
  pos. sentiment: 0.99 %
  neg. sentiment: 0.01 %

  Review: "clinton s multibillion dollar energy program"
  pos. sentiment: 0.16 %
  neg. sentiment: 0.84 %

  Review: "european asian markets rally on the back of financial shares indexes egged on after easter break by upbeat
investors"
  pos. sentiment: 0.96 %
  neg. sentiment: 0.04 %

  Review: "oil service shares rise natural gas sector up"
  pos. sentiment: 0.00 %
  neg. sentiment: 1.00 %
```

For the time period from July 14, 2008 to October 10, 2008, the results can be found in Figure 7.5

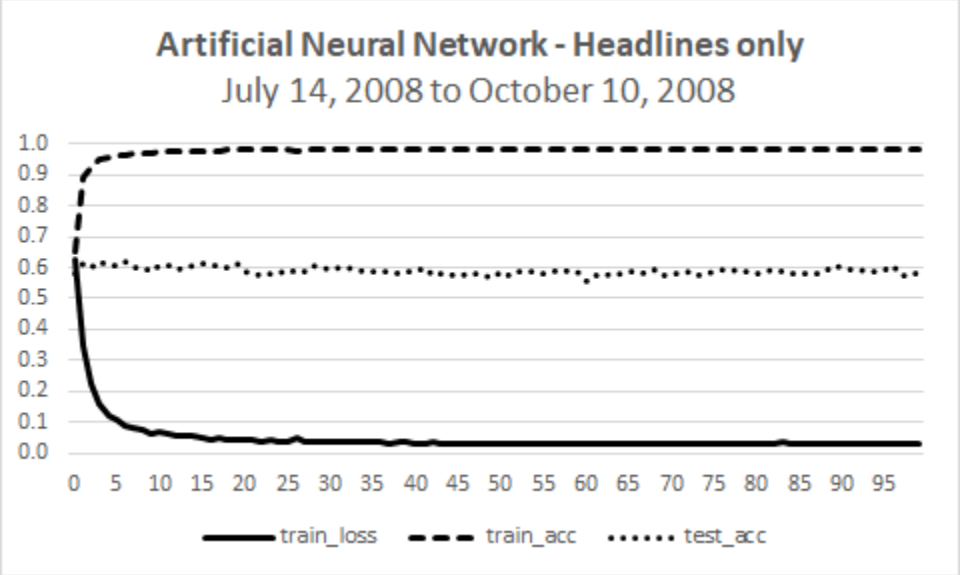

Figure 7.5: Artificial Neural Network - Headlines only from July 14, 2008 to October 10, 2008

Once again, early in the process, it is clear the training loss is falling quickly, and the training accuracy very quickly rises. The test continues to suffer from overfitting early in the process achieveing high training accuracy and relatively high test accuracy. Again, as more data is introduced in future epochs, the test begins to coalesce around its true predictive value. In this case, the test accuracy falls slightly to below .600 and continues to stay around .580. A review of the

sample output demonstrates similar "confusion" as above. For example, the following news story was labelled very confidently. While it is usually assigned a negative sentiment, and given the context, this label is probably correct, it did occasionally choose an incorrect sentiment:

> **Headline**: ConocoPhillips flirts with 52-week lows; Company's lackluster update, broad sector weakness weigh on shares
>
> **Author**: Steve Gelsi, MarketWatch
>
> **Date**: October 2, 2008
>
> **Change in share price from the previous day:** -$28.49 (negative sentiment)
>
> **Epoch 41**: train_loss: 0.032, train_acc: 0.983, test_acc: 0.597, pos. sentiment: 0.02 %, neg. sentiment: 0.98 %
>
> **Epoch 62**: train_loss: 0.030, train_acc: 0.984, test_acc: 0.578, pos. sentiment: 0.80 %, neg. sentiment: 0.20 %
>
> **Epoch 98**: train_loss: 0.027, train_acc: 0.984, test_acc: 0.575, pos. sentiment: 1.00 %, neg. sentiment: 0.00 %

Again, the sample headlines are usually labelled correctly as time goes on but this is somewhat less true for this sample set:

```
epoch_nr: 0, train_loss: 0.624, train_acc: 0.654, test_acc: 0.583
Test Samples:
  Review: "oil price drop cheers retail stocks"
  pos. sentiment: 0.01 %
  neg. sentiment: 0.99 %

  Review: "nyers feel heating oil bill chills"
  pos. sentiment: 0.24 %
  neg. sentiment: 0.76 %

  Review: "judges deal setback to proposed power line"
  pos. sentiment: 0.46 %
  neg. sentiment: 0.54 %

  Review: "u s power grid in better shape 5 years after blackout new standards systems help but concerns remain"
  pos. sentiment: 0.65 %
  neg. sentiment: 0.35 %

  Review: "credit crisis promises to delay entergy nuclear spinoff"
  pos. sentiment: 0.29 %
  neg. sentiment: 0.71 %

epoch_nr: 1, train_loss: 0.346, train_acc: 0.892, test_acc: 0.612
Test Samples:
  Review: "oil drilling off coast a dead end"
  pos. sentiment: 0.35 %
  neg. sentiment: 0.65 %

  Review: "some reality please"
  pos. sentiment: 0.61 %
  neg. sentiment: 0.39 %

  Review: "state department inspector to investigate texas oil company s deal in kurdistan"
  pos. sentiment: 0.27 %
  neg. sentiment: 0.73 %

  Review: "drillers pull in their horns commentary oil and gas sector slashes spending as demand tapers off"
  pos. sentiment: 0.56 %
  neg. sentiment: 0.44 %
```

```
  Review: "corrections for the record"
  pos. sentiment: 0.92 %
  neg. sentiment: 0.08 %
...
epoch_nr: 98, train_loss: 0.027, train_acc: 0.984, test_acc: 0.575
Test Samples:
  Review: "large stock focus financial stocks see rally fade late morgan stanley lehman decline genentech gains"
  pos. sentiment: 0.00 %
  neg. sentiment: 1.00 %

  Review: "conocophillips flirts with 52 week lows company s lackluster update broad sector weakness weigh on shares"
  pos. sentiment: 1.00 %
  neg. sentiment: 0.00 %

  Review: "buffett could reshape nuclear power industry"
  pos. sentiment: 1.00 %
  neg. sentiment: 0.00 %

  Review: "proposal would require generators at gas stations aim is to get fuel flowing after storms"
  pos. sentiment: 0.00 %
  neg. sentiment: 1.00 %


  Review: "thousands in area still in the dark residents business owners cope with lingering power outage from storm"

  pos. sentiment: 0.89 %
  neg. sentiment: 0.11 %

epoch_nr: 99, train_loss: 0.028, train_acc: 0.984, test_acc: 0.586
Test Samples:
  Review: "gas company to consider dividend boost"
  pos. sentiment: 0.53 %
  neg. sentiment: 0.47 %

  Review: "school board piles up the travel expenses"
  pos. sentiment: 0.08 %
  neg. sentiment: 0.92 %

  Review: "offshore drilling delegation from georgia leans in favor"
  pos. sentiment: 0.60 %
  neg. sentiment: 0.40 %

  Review: "international briefing"
  pos. sentiment: 1.00 %
  neg. sentiment: 0.00 %

  Review: "gas company to consider dividend boost"
  pos. sentiment: 0.53 %
  neg. sentiment: 0.47 %
```

For the time period from January 30, 2012 to April 30, 2012, the results can be found in Figure 7.6

As in the previous two runs, it is clear the training loss is falling quickly, and the training accuracy very quickly rises. This particular sample did not seem to show early overfitting. After, stabilization, the test accuracy reaches slightly over .560 and does not deviate much. A review of the sample output demonstrates similar "confusion" as above. For example, the following news story was labelled very confidently. While it is usually assigned a negative sentiment, and given the context, this label is probably correct. In this case, since the entire sector did not change much during this time period, one should expect the classification results to be somewhat less accurate:

> **Headline**: Oil stocks retreat, shed 1% for the week
> **Author**: Jim Jelter, MarketWatch

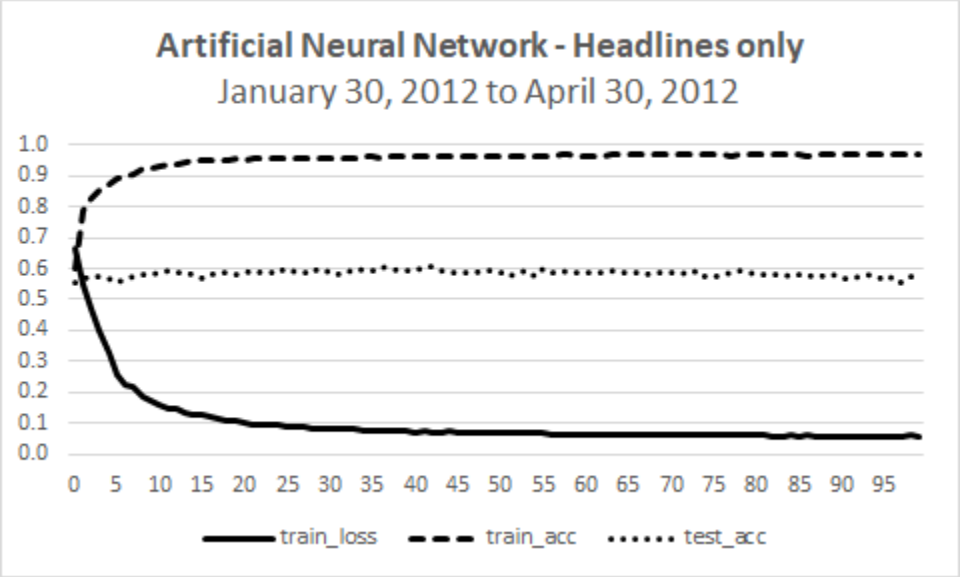

Figure 7.6: Artificial Neural Network - Headlines only from January 30, 2012 to April 30, 2012

Date: March 9, 2012

Change in share price from the previous day: -$0.76 (negative sentiment)

Epoch 1: train_loss: 0.551, train_acc: 0.786, test_acc: 0.570, pos. sentiment: 0.45 %, neg. sentiment: 0.55 %

Epoch 47: train_loss: 0.070, train_acc: 0.962, test_acc: 0.588, pos. sentiment: 0.02 %, neg. sentiment: 0.98 %

Epoch 69: train_loss: 0.062, train_acc: 0.966, test_acc: 0.589, pos. sentiment: 0.03 %, neg. sentiment: 0.97 %

Epoch 82: train_loss: 0.058, train_acc: 0.968, test_acc: 0.579, pos. sentiment: 0.04 %, neg. sentiment: 0.96 %

Again, the sample headlines are usually labelled correctly as time goes on but this is somewhat less true for this sample set:

```
epoch_nr: 0, train_loss: 0.662, train_acc: 0.601, test_acc: 0.555
Test Samples:
  Review: "a stock eye for the bond guy"
  pos. sentiment: 0.47 %
  neg. sentiment: 0.53 %

  Review: "refinery gets a look from delta perplexing analysts"
  pos. sentiment: 0.55 %
  neg. sentiment: 0.45 %

  Review: "exxon to keep its foot on gas pedal energy giant won t curb production despite natural gas slump strong
crude prices bolster quarterly earnings"
  pos. sentiment: 0.55 %
  neg. sentiment: 0.45 %


  Review: "gas set to test capacity limits"
  pos. sentiment: 0.52 %
  neg. sentiment: 0.48 %
```

```
  Review: "range resources moves into neutral territory"
  pos. sentiment: 0.60 %
  neg. sentiment: 0.40 %


epoch_nr: 1, train_loss: 0.551, train_acc: 0.786, test_acc: 0.570
Test Samples:
  Review: "engineers convene to seek ways to strengthen new york state s power grid"
  pos. sentiment: 0.42 %
  neg. sentiment: 0.58 %

  Review: "solarcity takes step toward public offering"
  pos. sentiment: 0.48 %
  neg. sentiment: 0.52 %

  Review: "u s stock futures up on oil s pullback data ecb also in focus priceline jumps on fourth quarter results"
  pos. sentiment: 0.72 %
  neg. sentiment: 0.28 %

  Review: "oil stocks retreat shed 1 for the week"
  pos. sentiment: 0.45 %
  neg. sentiment: 0.55 %

  Review: "friday s biggest gaining and declining stocks crocs deckers outdoor kindred kenneth cole salesforce com"
  pos. sentiment: 0.45 %
  neg. sentiment: 0.55 %
...
epoch_nr: 98, train_loss: 0.059, train_acc: 0.967, test_acc: 0.573
Test Samples:
  Review: "obscure banks plug gap in iran"
  pos. sentiment: 1.00 %
  neg. sentiment: 0.00 %

  Review: "high gas prices aren t just a figment of the imagination here s a chart showing just how much the price
at the pump has soared"
  pos. sentiment: 0.94 %
  neg. sentiment: 0.06 %

  Review: "pengrowth to buy nal energy in friendly stock deal"
  pos. sentiment: 0.00 %
  neg. sentiment: 1.00 %

  Review: "deevening links watergate exxon gas prices john kelly explains why the price of gas at the watergate exxon
is so high"
  pos. sentiment: 0.52 %
  neg. sentiment: 0.48 %

  Review: "start up bright automotive will close its doors officers at the electric vehicle start up bright automotive
says they are being forced to fold after three years of applying for federal funds and getting nowhere"
  pos. sentiment: 1.00 %
  neg. sentiment: 0.00 %

epoch_nr: 99, train_loss: 0.058, train_acc: 0.968, test_acc: 0.567
Test Samples:
  Review: "justices deny ex merrill executive s appeal in enron case"
  pos. sentiment: 1.00 %
  neg. sentiment: 0.00 %

  Review: "on our radar fracking dispute in pennsylvania green"
  pos. sentiment: 0.09 %
  neg. sentiment: 0.91 %

  Review: "financial briefing book feb 17"
  pos. sentiment: 0.23 %
  neg. sentiment: 0.77 %

  Review: "concentrating on petrobras projections"
  pos. sentiment: 0.63 %
  neg. sentiment: 0.37 %

  Review: "thursday s biggest gaining and declining stocks avid technology china cord blood wabtec"
  pos. sentiment: 1.00 %
```

```
neg. sentiment: 0.00 %
```

Chapter 8

Commentary

The result which stood out the most is the ability to be somewhat accurate in predicting price movement (and by implication, the probable sentiment) during the three-month period of greatest positive movement. From January 22, 2008 to April 22, 2008, the classifiers were able to combine for an average accuracy of 85.22% which is significant. The other two time periods showed some predictive accuracy but not to the same level. The predictive accuracy in this study is in line with a study on a similar domain where expert market forecasters could achieve accuracy between 70% and 79% at best and could achieve 48% accuracy on average suggesting this approach may have some potential (Bailey et al., 2017). The most likely reasoning for the differences in predictive accuracy in this study could be that during the period of positive movement, the energy sector might have very clear reasoning in the wording. For example, a news article from AP on January 23, 2008 showed a very clear positive sentiment for the energy sector. This article details a small drop in oil prices but details a general rise in the stock market as a whole including specific rises in shares for various oil companies:

> **Headline**: Index ticks up after Wall Street charges higher
> **Author**: Carla Mozee, MarketWatch
> **Date**: January 23, 2008
> **Change in share price from the previous day:** $2.83 (positive sentiment)
> **Body**: [. . .] U.S.-listed shares of foreign companies ended slightly higher Wednesday, reversing sharp losses as Wall Street staged a late-session rally led by stocks in the beleaguered financial sector. [. . .] A 2.8% fall in crude-oil prices to $86.75 a barrel on the New York Mercantile Exchange left most oil stocks lower. Total (TOT, US) fell 3.8%, Royal Dutch Shell (RDSA, US) tumbled 4%, BP (BP, US) lost 1.6% and StatoilHydro (STO, US) fell 3% to $25.30. But PetroChina (PTR, US) shares posted a 3.6% rise to $145.75 after a ratings upgrade to outperform from peer perform by Bear

Stearns. China Petroleum (SNP, US) shares surged 6.9% to $115.68 and CNOOC (CEO, US) rose 5.8% to $145.75.

Conversely, the stories might be more ambiguous during a period of significant drop in price. Unlike during the positive period, it might be less clear whether the news is positive or not. Since investors are often risk averse, ambiguous sentiment might be self-fulfilling prophecy for negative sentiment. For example, this article is from October 10, 2008. It is an opinion piece proposing the merit of granting greater authority and autonomy to the states to direct their own energy policy:

Headline: Georgia must develop a bold energy plan
Author: MARK BURKHALTER
Date: October 10, 2008
Change in share price from the previous day: -$29.67 (negative sentiment)
Body: [...] Washington has failed us when it comes to energy policy. [...] it is time to return energy policy to the states. [...] Washington has proved itself incapable of solving our energy problems. [...] States that discover oil and gas off their shores could share in the royalties. The hundreds of millions of dollars could result in important economic stimulus needed during economic downturns such as tax cuts or a check for every citizen[...] It is technology, not gasoline, that is making our air cleaner. As newer vehicles come on line, we will continue to see a decrease in pollutants in the metro region. No natural disaster or tragedy should disrupt the state's economy, including the transport of commerce and educating of our children. Planning for the future across our energy portfolio will hopefully insulate us from an energy crisis of any stripe. [...]

Or the price movement might not have been directly related to the energy sector at all. Rather, the economic crisis as it related tangentially to the energy sector might have dominated investor mood. For example, the following article details the global credit crunch which is triggering falling oil prices and explains how this macroeconomic shift is affecting various economies who are heavily dependent on oil exports:

Headline: Oil's Drop Squeezes Producers; Economies of Iran, Venezuela Vulnerable as Crude Price Falls but Demand Stays Low
Author: By Neil King Jr. and Spencer Swartz
Date: October 10, 2008
Change in share price from the previous day: -$29.67 (negative sentiment)
Body: Big oil-producing countries are showing signs of distress as the global credit crunch and falling crude prices begin to squeeze government budgets and delay

> projects. [. . .] The global economic crisis is eating into oil demand, particularly in the U.S. and Europe, and helping drive down crude prices. Some forecasters said that despite a strong thirst for oil in Asia and the Middle East, global oil consumption could flatten out next year, potentially ending nearly a decade of steady demand growth. [. . .] "The global credit crunch has seen the number of international banks lending to the power and water sector decline," said Medley energy director Bill Farren-Price. [. . .] signs are emerging in other OPEC countries that energy projects could get caught in the financial fallout. The industry's efforts to pump more oil and natural gas already are suffering from high costs, technical challenges and political barriers. [. . .]

The significant drop off in accuracy for the "extreme days" could be the result of external investor sentiment across the entire stock market rather than any specific news coming out related to the energy sector. For example, the energy sector, in general, might move positively or negatively in very small amounts. However, investors may suddenly become bullish due to world events and may push more money out of safer investments such as bonds and into the stock market; a percentage of which may pump into the energy sector. Conversely, bearish sentiment might cause investors to pull out of the stock market which will translate into lesser investment in the energy market. For example, consider the following article from March 19, 2008 where the S&P 500 Energy index dropped $30.82 (It's largest drop between January 22, 2008 and April 22, 2008). It is implied that the drop is partially caused by the drop in oil prices but was primarily caused by rate cuts from the Federal Reserve and a general retreat from the stock market as a whole:

> **Headline**: Sector tumbles on crude, equity-market reversals
> **Author**: Steve Gelsi & Jim Jelter, MarketWatch
> **Date**: March 19, 2008
> **Change in share price from the previous day:** -$30.82 (negative sentiment)
> **Body**: [. . .] Energy stocks accelerated their decline Wednesday afternoon, hit by a bout of profit-taking across the equities and commodities markets that locked in some of the huge gains built in the previous session on the Federal Reserve's latest rate cut.The retreat also was driven by a sharp drop in oil prices after the Energy Information Administration reported that U.S. crude supplies rose 200,000 barrels to 311.8 million barrels in the week ending March 14. [. . .] But the energy market remains well supplied, and analysts pegged the drop in oil prices to fatigue in the commodities rally over the past few months and fears of further weakness in the global economy.[. . .]

It is also possible that the resulting data, with only 21 extreme days from January 22, 2008 to April 22, 2008 and 32 days from July 14, 2008 and October 10, 2008 to work with, is actually too small

and therefore unpredictable results are being generated.

The bigram analysis clearly did not produce useful results. Since any analysis over all three time periods of the bigrams regularly fell around 50% accuracy, the results would be considered on par with random chance. However, it should be noted that the bigram analysis used only the naïve bayes classifier. This differed from the unigram analysis which used a collection of classifiers and averaged the results of each classifier. Since the unigram analysis demonstrated that other classifiers produced better results than the naïve bayes classifier, it would be reasonable to conclude that the bigram analysis could have potentially been improved by employing more classifiers. Furthermore, the bigram analysis did not clean up the input by removing stop words or specific parts of speech. Choosing to not preprocess the input may have negatively affected the bigram results. It is possible that the presence of these words could have confused the naïve bayes classifier somewhat and had an influence on the poorer results.

Using an artificial neural network for sentiment classification demonstrated improved results. The three time periods analyzed demonstrated accuracy approaching 60% with headlines generally showing better accuracy than full stories. When analyzing full stories, it is probable the results are poor due to excessive noise in the data or because the ANN lacks sufficient resources to calculate all the data well. When analyzing headlines only, some improvement was observed but some headlines are only a few words in length and sometimes meaning is lost when stripping out punctuation. It is perhaps more probable the results are poor due to insufficient data which is the inverse problem from analyzing the entire text. It should be noted that a significant deviation for this approach over the unigram and bigram approach is very little cleansing of the input was performed. For example, stop words were not removed and common English phrases that are not specific to the subject matter were not removed. This would be a difficult approach to take anyway as the sentiment of a news article is more likely to be derived from common adjectives used in the English language rather than the nouns that are specific to the subject matter. Other confounding factors could be attributed to the same author repeating phrases for the sector they cover. This could confuse the ANN to attribute similar sentiment, when in fact, the author has simply chosen to use similar phrases for different contexts. For example, the phrase "investors weighed" appeared in two separate news stories. Note in the first story, investors are weighing positive comments.

> **Headline**: Oil service shares lead declines as sector waffles
>
> **Author**: Steve Gelsi, MarketWatch
>
> **Date**: January 31, 2008
>
> **Change in share price from the previous day:** $1.76 (positive sentiment)
>
> **Body**: [. . .] Energy shares fought their way back from lows of the session up into positive territory, but ended mixed as investors weighed positive comments from bond insurer MBIA (MBI, US). [. . .]

In the second story, investors are weighing a failure.

> **Headline**: Energy shares fall despite gains in broad market
> **Author**: Steve Gelsi, MarketWatch
> **Date**: September 26, 2008
> **Change in share price from the previous day:** -$9.20 (negative sentiment)
> **Body**: [. . .] Energy stocks ended lower despite a turnaround in the broad market on Friday as oil prices retreated and investors weighed the failure of Washington Mutual and the impact of a slower economy on petroleum demand. [. . .]

However, the same author used the same phrase in each story. Based on the date of publication, the story is calculated to have differing sentiments which could have created confusing results. Since "investors weighed" is a bigram, the bigram results could have been influenced negatively in a similar fashion as well.

Time Period (Average)	Unigrams	Bigrams	ANN Full Stories	ANN Headlines Only
January 22, 2008 to April 22, 2008 (Positive) (All Days)	85.22%	53.04%	56.00%	62.10%
July 14, 2008 to October 10, 2008 (Negative) (All Days)	59.44%	51.34%	56.50%	58.60%
January 30, 2012 to April 30, 2012 (Average) (All Days)	56.47%	52.16%	57.00%	56.70%

Table 8.1: Summary of all average results for all methods

According to the summary in Table 8.1, the time period from January 22, 2008 to April 22, 2008 was best predicted by the unigram method by far. Applying the ANN to the headlines was not as successful but would still be considered good. The bigram analysis and applying the ANN to the full stories were only a little better than random chance. Similarly, the time period from July 14, 2008 to October 10, 2008 was best predicted by the unigram model followed very closely by applying the ANN to the headlines. The bigram analysis and applying the ANN to the full stories lags behind with results only a little better than random chance. All methods applied to the time period from January 30, 2012 to April 30, 2012 had poor predictive value in general. The unigram and both ANN methods produced predictive results about the same and the bigram method continued to fall behind the others at a little better rate than random chance. In general, the resulting data indicates that the unigram analysis had the best predictive value followed by applying

the ANN to headlines. Applying the ANN to the full stories fell behind those two methods and the bigram method was consistently the least predictive.

Finally, the practice of pigeonholing the results into positive or negative sentiment is causing the data to become erratic. This is clear in all sample sets using all tested methods, but perhaps it comes through most clear in the period of least volatility when sentiment should be expected to be ambiguous. A news story that coincides with a positive news day could be "poisoning" the data causing future predictability for positive days to suffer, or vice-versa. It is entirely reasonable to believe that some news stories either express no true sentiment, or similarly, a news story could divide investor sentiment as to whether the news should be interpreted as positive or negative at all.

Chapter 9

Future Directions

Throughout this work, there have been multiple instances where decisions were made for continued analysis that any intrepid researcher could make the alternative decision and push the research in a similar but slightly different direction. This could provide a bit more perspective on these results and help clarify this field. The purpose of this research was to determine to what extent the price movement of entire sectors of the market could be predicted by analyzing news stories with natural language processing and sentiment analysis techniques. To continue down this vein in a slightly alternative direction, one could employ any one or combination of the following options:

First, the S&P 500 indices matching the GICS sectors seemed to be a reasonable selection of indices that could track a sector, but there are others to choose from. It would be interesting to discover whether a different index tracking the sector differently could produce similar results. Similarly, the S&P 500 selection was not required either. Standard and Poor's lists 38 Energy related indices, only one of which is the S&P 500 Energy index. Other choices could have been more focused energy sectors such as the "S&P Emerging BMI Energy" which is defined as an index which "provides investors with a benchmark that reflects those companies included in the S&P Emerging BMI that are classified as members of the GICS® energy sector and sub-industries". Or one could choose a fund that focusses on a different geographical sector such as the "S&P/NZX All Energy" which is defined as an index which "comprises members of the S&P/NZX All Index, considered the total market indicator for the New Zealand equity market, classified within the energy sector of the Global Industry Classification System (GICS®)." Standard & Poor's also provides broader indices such as the "S&P Composite 1500 Energy" which "comprises those companies included in in the S&P Composite 1500 that are classified as members of the GICS® energy sector."

Second, S&P 500 Energy index was selected based on its short-term volatility relative to the other indices analyzed. It is unknown without further analysis whether the results discovered here can be equally or proportionally applicable to entirely different sectors of the market. It would be interesting to discover if one of the other candidate indices such as the S&P 500 Health Care index

could produce similar results to the S&P 500 Energy index. Alternatively, it would be interesting to see if an index that had relatively little short-term volatility such as the S&P 500 Industrials could produce similar results. Alternatively, it would be interesting to see if an index that had relatively poor performance even if its best three month period such as the S&P 500 Telecommunications Services could produce similar results.

A different path for research would be to keep the same stock index selection of the S&P 500 Energy index and use a different source of news stories. The choice to use Major News and Business Sources: U.S. was explained above. However, there can be compelling arguments to use other sources of news stories. Some attractive options such as using only newswires might produce interesting results. Or one might choose to use a very specific source. This could be one that investors are very likely to be reading such as the Wall Street Journal, or one that only casual investors might be reading such as USA Today. Even the choice to use published news articles was not a requirement. One might choose to analyze transcripts from cable news channels with specific focus such as CNNMoney or with very broad focus such as CNN.

Assuming one chose to use the S&P 500 Energy index and Major News and Business Sources: U.S. for analysis, one could decide that the headlines themselves might produce similar results. In fact, one could reasonably make the argument that most investors working a fast-paced environment may be inclined to make trading decisions based purely on the content of the headlines and may not bother to read the content at all. This may be a difficult path to follow as the breadth of content contained in headlines is not extensive, however, since the headlines are intended to be succinct and clear, a researcher could avoid dealing with an author's editorialization choices such as avoiding the same words multiple times in an article.

Deciding whether an article conveyed positive or negative sentiment might be the most tenuous element of this work. The sentiment was decided based on the price movement of the index on the same day the article was released. Of course, negative news stories can come out on a day the price closes higher and positive news stories can come out on a day the price closes lower. One might consider following the alternative I discussed such as employing humans to analyze the articles and decide sentiment. The analyzers could be professionals in the field, or they could people with no specific skill set in the field.

If someone chose to use the S&P 500 Energy index and Major News and Business Sources: U.S. for analysis, one could select different time periods than I did. The choice to use 3 months was to keep the work focused but was not a requirement. One could decide to analyze news stories over one of the other time periods I analyzed such as the three positive, negative, and most average 6-month, 12-month, 2 years, 5 years, or 10-year periods. The decision to use the period of greatest rise, greatest drop, and least price movement was selected based on my perceived usefulness to the analysis. However, one might choose to use a completely different time period for reasons of their

own choosing. For example, one might wish to purposely avoid a U.S. presidential election cycle, or the "Great Recession", or any significant historical period at all. Or one might purposely select a period of great historical significance and compare it to one that is not particularly significant. For example, one might choose to analyze from November 6, 2011 and November 6, 2012 which would coincide with the U.S. presidential cycle, and then compare those results with November 6, 2012 to November 6, 2013.

A researcher could choose to keep the same index selection, sources, and time periods, but could choose instead to analyze different parts of speech. I chose adjectives, adverbs, and verbs, but one could decide to use a smaller set, or they could use a larger set. The parts of speech used for this analysis were very generic and included all tenses and forms. One might decide that only the present tense of a verb such as "rising" or "falling" will produce more interesting results over the past tense such as "rose" or "fell". Similarly, perhaps narrowing in on superlative adjectives such as "quickest" or "slowest" might be more interesting over the comparatives such as "quickly" and "slowly".

For my list of informative words, I used only the top 33% of all words found based on frequency. This appeared to be a useful cut-off. However, this number could be adjusted to include a smaller set such as the top 10% or a larger set such as the top 50%, or even a specific number such as the top 100 words could be used. One could at this point introduce a human factor to decide whether there are some words that should not be included in the analysis and manually remove those words from the set for analysis. For example, based on the sample provided above, the word "contains" might not be a very helpful word, but it coincidentally appeared much more often in positive sentiment articles. On the other hand, the word "spur" could quite clearly indicate positive sentiment, and someone would ensure that word is included. If the human factor were included at this stage, there might be opportunity to purposely keep words that are particularly relevant to the subject matter. For example, seemingly innocuous words such as "reserve" or "field" are words that one might not consider to be very interesting. However, within the context of the energy sector, these words are highly relevant.

The choice of classifiers was based on known classifiers that were available. The final results were based on an average of all the classifiers used. However, one might decide that, given the subject matter, some classifiers should get greater weight over others. One could also consider introducing different classifiers that were not used in this work.

The results of the bigram analysis turned out to be rather disappointing but could perhaps be improved. For example, one might consider keeping only bigrams that follow a specific pattern such as adjective-verb that can be found in phrases like "rose quickly" or "fell slowly". Again, the bigram analysis used the Naïve Bayes Classifier only and other classifiers could be introduced. Each classifier could have equal weighting or other classifiers could receive greater or lesser weight-

ing. Finally, one might consider using other ngrams such as trigrams or more. Phrases such as "price of oil rose quickly" or "price of oil fell quickly" could have significant influence on investor sentiment.

The artificial neural network could possibly be improved with greater attention to the training data. Removing stop words or certain phrases may be able to improve the test results. Alternatively, since an ANN works on the text in aggregate, some better pruning of entire stories would likely improve the results. For example, purposely removing ambiguous stories from the training set could make the remaining stories easier to classify either positively or negatively.

Chapter 10

Conclusions

In an effort to establish a technique for predicting market movement for a sector of the market using sentiment analysis was completed and found to have promising results. An introduction and review of stock markets and exchange traded funds was covered. Both quantitative and qualitative analysis techniques were introduced to explain what factors an investor may consider when investing in the stock market. A review of existing literature and research on the subject was done. The base ideas of the efficient market hypothesis and random-walk were explained. Recent research running counter to the efficient market hypothesis was covered lending credence to the idea that some areas of the stock market may be predictable in an automated way through intelligent algorithms.

Continuing down the qualitative analysis vein, I summarized 11 exchange traded funds that contained sufficient data for review. The 11 exchange traded funds were compared with one another over 7 different time periods in order to measure relative variability. The S&P 500 Energy index appeared excessively variable over short periods of time (3 months, 6 months, and 12 months) but also excessively stable over a 10-year period from February 2006 to February 2016. Therefore, the best sector to track appeared to be the energy sector.

I then gathered news stories related to the energy sector for the most volatile 3-month periods from January 22, 2008 to April 22, 2008, July 14, 2008 to October 10, 2008 and the most stable 3-month period of January 30, 2012 to April 30, 2012. The news stories were extracted and loaded into a database for easy analysis. I reduced the news stories to a manageable level by keeping, just the content of the articles, removing stop words, keeping only remaining verbs, adverbs, and adjectives, and only keeping the top 1/3rd of the most commonly used words. The news was then split into a training and testing set at a ratio of 9 to 1. The news was automatically classified as positive or negative based on the price of the ETF on that day. I then applied multiple classifiers against the training and test sets to attempt to classify the data. The classifiers together would then provide equal voting on how well they were able to classify the testing set. This process of splitting training and testing sets and trying the classifiers was repeated 100 times to eliminate any

variability and ensure a regression to the actual mean. To develop a bit of perspective, I ran the exact same analysis using only extremely volatile days. I also attempted to follow a similar method for bigrams rather than narrowing down on only verbs, adverbs, and adjectives.

Over the S&P 500 Energy Index's most positive 3-month period, my program was able to successfully classify the news stories with an accuracy of 85.22% on average. Over the most negative 3-month period, my program was able to successfully classify the news stories with an accuracy of 59.32%. Over the most average 3-month period, my program was able to successfully classify the news stories with an accuracy of 56.47%. Using only extremely volatile days, my program could classify news stories with an accuracy of 59.32%, 57.76%, and 57.03% respectively. Analyzing bigrams, my program could successfully classify news stories with an accuracy of 53.04%, 51.34%, and 52.16% respectively.

Finally, an artificial neural network was employed against the most volatile 3-month periods from January 22, 2008 to April 22, 2008, July 14, 2008 to October 10, 2008 and the most stable 3-month period of January 30, 2012 to April 30, 2012. The results were slightly better than random chance suggesting there is room for improvement. Applying the ANN to headlines only showed modest improvement. However, the ANN used was not sufficient to demonstrate a useful improvement on stock market prediction.

A review of potential future work was covered. This research covered a very narrow focus of a very specific exchange traded fund over a very specific set of time periods. There are several opportunities where a researcher could adjust some of the variables to cover different sectors over different time periods in different ways. To build on the success of the aforementioned 85.22% accuracy, the area with the most potential would be to cover a different sector of the market over its most positive three month period and attempt to use the classifiers in the same way to determine the accuracy of sentiment.

10.1 Research contributions

This study reinforced some existing ideas and added several new contributions to this field:

- Most of the previous research focused on either very general indicies such as the Dow Jones Industrial Average or a small and specific set of stocks. This study focused on a middle ground of tracking a large but specific sector of the stock market. Employing an exchange traded fund to represent a sector for this study is also unique.

- The process to select the ideal exchange traded fund to track by comparing past perfomance against similar funds over different time periods is a unique contribution. This helped narrow

the focus on a specific sector over a specific time period which allowed the number of news articles to be contained with specific criteria.

- The approach of using price changes as a proxy for the sentiment of news articles was demonstrated to be efficient and useful with no significant loss in true sentiment.
- Many earlier works employed only a Naïve Bayes Classifier but employing multiple classifiers together was demonstrated to be useful and more effective.
- Unigrams of specific parts of speech and bigrams were analyzed. The classification results were compared to each other and to ANN classification attempts.
- Many earlier works employed an ANN against quantitative data. This study employed an ANN against qualitative data and demonstrated some potential for predictability.

Bibliography

Aase, K.-G., 2011, Text Mining of News Articles for Stock Price Predictions: Master's thesis, Norwegian University of Science and Technology.

Abu-Mostafa, Y. S., and A. F. Atiya, 1996, Introduction to financial forecasting: Applied Intelligence, **6**, 205–213.

Adya, M., and F. Collopy, 1998, How effective are neural networks at forecasting and prediction? A review and evaluation: Journal of Forecasting, **17**, 481–495.

Alamoodi, A. H., B. B. Zaidan, A. A. Zaidan, O. S. Albahri, K. I. Mohammed, R. Q. Malik, E. M. Almahdi, M. A. Chyad, Z. Tareq, A. S. Albahri, H. Hameed, and M. Alaa, 2021, Sentiment analysis and its applications in fighting COVID-19 and infectious diseases: A systematic review: Expert Systems with Applications, **167**, 114155.

Bailey, D. H., J. M. Borwein, A. Salehipour, and M. Lopez de Prado, 2017, Evaluation and Ranking of Market Forecasters. https://ssrn.com/abstract=2944853.

Barkur, G., Vibha, and G. B. Kamath, 2020, Sentiment analysis of nationwide lockdown due to COVID 19 outbreak: Evidence from India: Asian Journal of Psychiatry, **51**, 102089.

Beattie, A., 2017 (accessed November 20, 2017), The birth of stock exchanges. (`https://www.investopedia.com/articles/07/stock-exchange-history.asp?ad=dirN&qo=investopediaSiteSearch&qsrc=0&o=40186`).

Bollen, J., H. Mao, and X. Zeng, 2011, Twitter mood predicts the stock market: Journal of Computational Science, **2**, 1–8.

Cao, L., and F. E. Tay, 2001, Financial Forecasting Using Support Vector Machines: Neural Computing & Applications, **10**, 184–192.

Cao, L. J., and F. E. Tay, 2003, Support vector machine with adaptive parameters in financial time series forecasting: IEEE Transactions on Neural Networks, **14**, 1506–1518.

Costantino, M., and P. Coletti, 2008, Information extraction in finance: Wit Press.

Day, M. Y., and J. T. Lin, 2019, Artificial intelligence for ETF market prediction and portfolio optimization: Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2019, 1026–1033.

Dubner, S. J., 2017 (accessed November 20, 2017), The stupidest thing you can do with your

money. (http://freakonomics.com/podcast/stupidest-money/).

Falinouss, P., 2007, Stock Trend Prediction Using News Articles A Text Mining Approach: Master's thesis, Lulea University of Technology.

Fama, E. F., 1965, The Behavior of Stock-Market Prices: Journal of Business, **38**, 34–105.

Gidófalvi, G., 2001, Using news articles to predict stock price movements: Department of Computer Science and Engineering, University of California San Diego, **manuscript**, 9pp.

Habimana, O., Y. Li, R. Li, X. Gu, and G. Yu, 2020, Sentiment analysis using deep learning approaches: an overview: Science China Information Sciences, **63**, 1–36.

Hagenau, M., M. Liebmann, and D. Neumann, 2013, Automated news reading: Stock price prediction based on financial news using context-capturing features: Decision Support Systems, **55**, 685–697.

Hassan, M. R., B. Nath, and M. Kirley, 2007, A fusion model of HMM, ANN and GA for stock market forecasting: Expert Systems with Applications, **33**, 171–180.

Intercontinental Exchange, I., 2017 (accessed November 20, 2017), Holidays - all markets. (https://www.nyse.com/markets/hours-calendars).

Inthachot, M., V. Boonjing, and S. Intakosum, 2016, Artificial Neural Network and Genetic Algorithm Hybrid Intelligence for Predicting Thai Stock Price Index Trend: Computational Intelligence and Neuroscience, **2016**, 1–8.

Jiang, W., 2020, Applications of deep learning in stock market prediction: recent progress: arXiv preprint arXiv_2003.01859.

Jin, Z., Y. Yang, and Y. Liu, 2020, Stock closing price prediction based on sentiment analysis and LSTM: Neural Computing and Applications, **32**, 9713–9729.

Kara, Y., M. Acar Boyacioglu, and Ö. K. Baykan, 2011, Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange: Expert Systems with Applications, **38**, 5311–5319.

Kim, K. J., 2003, Financial time series forecasting using support vector machines: Neurocomputing, **55**, 307–319.

Kimoto, T., K. Asakawa, M. Yoda, and M. Takeoka, 1990, Stock market prediction system with modular neural networks: 1990 IJCNN International Joint Conference on Neural Networks, IEEE, 1–6 vol.1.

Kinsley, H., 2015 (accessed November 20, 2017), Nltk with python 3 for natural language processing. (https://www.youtube.com/playlist?list=PLQVvvaa0QuDf2JswnfiGkliBInZnIC4HL).

Koppel, M., and I. Shtrimberg, 2005, Good news or bad news? Let the market decide: AAAI Spring Symposium - Technical Report, **SS-04-07**, 86–88.

Lewis, D. D., and W. A. Gale, 1994, A sequential algorithm for training text classifiers: Proceed-

ings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1994, 3–12.

Li, X., P. Wu, and W. Wang, 2020, Incorporating stock prices and news sentiments for stock market prediction: A case of Hong Kong: Information Processing and Management, **57**, 102212.

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

Mittermayer, M.-a., 2004, Forecasting Intraday stock price trends with text mining techniques: Proceedings of the 37th Hawaii International Conference on System Sciences - 2004, **00**, 1–10.

Moghar, A., and M. Hamiche, 2020, Stock Market Prediction Using LSTM Recurrent Neural Network: Procedia Computer Science, **170**, 1168–1173.

Mostafa, M. M., 2010, Forecasting stock exchange movements using neural networks: Empirical evidence from Kuwait: Expert Systems with Applications, **37**, 6302–6309.

Opperman, A., 2019 (accessed November 20, 2019), Sentiment analysis LSTM recurrent neural network's. (https://github.com/artem-oppermann/Sentiment-Analysis-of-Netflix-Reviews).

Rapp, N., and C. Leaf, 2018 (accessed October 6, 2018), Anatomy of a bull market. (http://fortune.com/longform/anatomy-of-a-bull-market/).

Rennie, J. D., L. Shih, J. Teevan, and D. Karger, 2003, Tackling the Poor Assumptions of Naive Bayes Text Classifiers: Proceedings, Twentieth International Conference on Machine Learning, 616–623.

Schölkopf, B., A. J. Smola, R. C. Williamson, and P. L. Bartlett, 2000, New support vector algorithms: Neural Computation, **12**, 1207–1245.

Schumaker, R. P., and H. Chen, 2009, Textual analysis of stock market prediction using breaking financial news: ACM Transactions on Information Systems, **27**, 1–19.

Singh, R., and S. Srivastava, 2017, Stock prediction using deep learning: Multimedia Tools and Applications, **76**, 18569–18584.

Staff, I., 2017 (accessed November 20, 2017)a, Dow Jones Industrial Average - DJIA. (https://www.investopedia.com/terms/d/djia.asp?ad=dirN&qo=investopediaSiteSearch&qsrc=0&o=40186).

———, 2017 (accessed November 20, 2017)b, Introduction to exchange-traded funds. (https://www.investopedia.com/articles/01/082901.asp?ad=dirN&qo=investopediaSiteSearch&qsrc=0&o=40186).

Strategesis, 2011 (accessed November 20, 2017), NYSE holidays, 2001-2011. (https://strategesis.wordpress.com/2011/04/01/nyse-holidays-1885-2011/).

Tay, F. E., and L. Cao, 2001, Application of support vector machines in financial time series forecasting: Omega, **29**, 309–317.

Thakkar, A., and K. Chaudhari, 2021, Fusion in stock market prediction: A decade survey on the necessity, recent developments, and potential future directions: Information Fusion, **65**, 95–107.

Wei, L.-Y., 2016, A hybrid ANFIS model based on empirical mode decomposition for stock time series forecasting: Applied Soft Computing, **42**, 368–376.

White, H., 1988, Economic prediction using neural networks: The case of IBM daily stock returns: IEEE International Conference on Neural Networks, IEEE, 451–458.

World-Stock-Exchanges.net, 2012 (accessed November 20, 2017), Ten largest stock exchanges in the world by market capitalization in 2011. (`http://www.world-stock-exchanges.net/top10.html`).

Wuthrich, B., V. Cho, S. Leung, D. Permunetilleke, K. Sankaran, and J. Zhang, 1998, Daily stock market forecast from textual web data: SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218), **3**, 1–6.

Yadav, A., and D. K. Vishwakarma, 2020, Sentiment analysis using deep learning architectures: a review: Artificial Intelligence Review, **53**, 4335–4385.

Yasef Kaya, M. I., and M. Elif Karsligil, 2010, Stock price prediction using financial news articles: Proceedings - 2010 2nd IEEE International Conference on Information and Financial Engineering, ICIFE 2010, 478–482.

Yoon, Y., and G. Swales, 1991, Predicting stock price performance: A neural network approach: Proceedings of the Annual Hawaii International Conference on System Sciences, IEEE Comput. Soc. Press, 156–162.

Appendix A

Source Code

A.1 ParseHTML.py

```
1  import glob
2  from bs4 import BeautifulSoup
3  import re
4  import mysql.connector
5  from mysql.connector import errorcode
6  from time import strptime
7  import logging, sys
8
9  def insertNews(cnx, headline, author, date, body):
10     logging.debug(headline, author, date, body)
11     date_split = date.split(' ')
12     mysqldate = "{}-{}-{}".format(date_split[2],
13                                   strptime(date_split[1], '%B').
                                        tm_mon,
14                                   date_split[0])
15     add_article = ("INSERT INTO news "
16                    "(theadline, tauthor, tdate, tbody) "
17                    "VALUES (%s, %s, %s, %s)")
18     data_article = (headline, author, mysqldate, body)
19     logging.debug(add_article, data_article)
20     cursor = cnx.cursor()
21     cursor.execute(add_article, data_article)
```

```
22
23 def getNews(globString, cnx):
24     filelist = glob.glob(globString)
25
26     for file in filelist:
27         logging.warning(file)
28         with open(file) as f:
29             html_doc = f.read()
30
31         soup = BeautifulSoup(html_doc, 'html.parser')
32
33         # logging.debug(soup.get_text())
34
35         for tag in soup.find_all(re.compile("^div"), class_="
               enArticle"):
36             try:
37                 headline = tag.find(re.compile("^div"), id="hd")
38                 headline = headline.get_text().strip()
39                 logging.debug("HEADLINE: " + headline)
40             except AttributeError:
41                 headline = "UNKNOWN"
42                 logging.debug("HEADLINE: " + headline)
43
44             try:
45                 author = tag.find(re.compile("^div"), class_="
                       author")
46                 author = author.get_text().strip()
47                 logging.debug("AUTHOR: " + author)
48             except AttributeError:
49                 author = "UNKNOWN"
50                 logging.debug("AUTHOR: " + author)
51
52             try:
53                 date = tag.find(re.compile("^div"), text=re.
                       compile("\d{1,2}\s\w+\s\d{4}"))
54                 date = date.get_text().strip()
```

```
55                 logging.info("DATE: " + date)
56             except AttributeError:
57                 date = "UNKNOWN"
58                 logging.info("DATE: " + date)
59
60             # input("X"*10)
61             article = ""
62             for child in tag.find_all(re.compile("^p"), class_="
                   articleParagraph"):
63                 article = article + child.get_text()
64             logging.debug("ARTICLE: " + article)
65             # input("-"*20)
66             # break
67             insertNews(cnx, headline, author, date, article)
68
69 if __name__ == '__main__':
70     logging.basicConfig(stream=sys.stderr, level=logging.WARNING)
71     try:
72       cnx = mysql.connector.connect(user='thesis-user', password=
             'thesis', host='127.0.0.1', database='thesis')
73       getNews('/media/sf_School/Exports/snp500Energy-*/*.html',
           cnx)
74       cnx.commit()
75     except mysql.connector.Error as err:
76       if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
77         logging.error("Something is wrong with your user name or
               password")
78       elif err.errno == errorcode.ER_BAD_DB_ERROR:
79         logging.error("Database does not exist")
80       else:
81         logging.error(err)
82     else:
83       cnx.close()
```

A.2 read_db.py

```
1  #!/usr/bin/python3
2  import itertools
3  import mysql.connector
4  from mysql.connector import errorcode
5  import logging, sys
6  from nltk.corpus import stopwords
7  from nltk.collocations import BigramCollocationFinder
8  from nltk.metrics import BigramAssocMeasures
9  import nltk
10 import random
11 # from nltk.corpus import movie_reviews
12 from nltk.classify.scikitlearn import SklearnClassifier
13 import pickle
14 from sklearn.naive_bayes import MultinomialNB, BernoulliNB
15 from sklearn.linear_model import LogisticRegression,
       SGDClassifier
16 from sklearn.svm import SVC, LinearSVC, NuSVC
17 from nltk.classify import ClassifierI
18 # from statistics import mode
19 from nltk.tokenize import word_tokenize
20 from datetime import datetime
21 from statistics import mean
22
23
24 def show_most_informative_features_rewrite(self, n=10):
25     strlist = []
26     # Determine the most relevant features, and display them.
27     cpdist = self._feature_probdist
28     # print('Most Informative Features')
29     strlist.append('Most␣Informative␣Features')
30
31     for (fname, fval) in self.most_informative_features(n):
32         def labelprob(l):
33             return cpdist[l, fname].prob(fval)
34
35         labels = sorted([l for l in self._labels
```

```
36                          if fval in cpdist[l, fname].samples()],
37                         key=labelprob)
38          if len(labels) == 1: continue
39          l0 = labels[0]
40          l1 = labels[-1]
41          if cpdist[l0, fname].prob(fval) == 0:
42              ratio = 'INF'
43          else:
44              ratio = '%8.1f' % (cpdist[l1, fname].prob(fval) /
45                                 cpdist[l0, fname].prob(fval))
46          # print(('%24s = %-14r %6s : %-6s = %s : 1.0' %
47          #       (fname, fval, ("%s" % l1)[:6], ("%s" % l0)[:6],
                ratio)))
48          strlist.append(('%24s = %-14r %6s : %-6s = %s : 1.0' %
49                          (fname, fval, ("%s" % l1)[:6], ("%s" % l0
                                )[:6], ratio)))
50
51      return strlist
52
53
54  def readDB(cursor, dateStart, dateEnd, neg=False,
        stock_overprevious=0):
55      if neg:
56          sql = ("SELECT * FROM thesis.stock_data,thesis.news "
57                 "WHERE thesis.news.tdate = thesis.stock_data.
                       stock_date "
58                 "AND stock_date >= %s "
59                 "AND stock_date <= %s "
60                 "AND stock_overprevious < %s "
61                 "ORDER BY stock_date"
62                 )
63      else:
64          sql = ("SELECT * FROM thesis.stock_data,thesis.news "
65                 "WHERE thesis.news.tdate = thesis.stock_data.
                       stock_date "
66                 "AND stock_date >= %s "
```

```
67                  "AND stock_date <= %s "
68                  "AND stock_overprevious >= %s "
69                  "ORDER BY stock_date"
70                  )
71      cursor.execute(sql, (dateStart, dateEnd, stock_overprevious))
72      return cursor.fetchall()
73
74
75  def evaluate(posNews, negNews, period):
76      # Pull out ONLY the body of the story (a[9]).
77      #   Include a[6] for the headline
78      short_pos = []
79      short_neg = []
80      for a in posNews:
81          short_pos.append(str(a[9]))
82      for a in negNews:
83          short_neg.append(str(a[9]))
84
85      # allowed_word_types is all the POS tags we are analyzing
            against
86      #   j is adjective, r is adverb, and v is verb
87      allowed_word_types = ["J", "R", "V"]
88      # allowed_word_types = ["J"]
89
90      # all_words contains all the words that are POS tagged in
            allowed_word_types
91      # eg.
92      #   ["word1", "word2", ... , "wordN"]
93      all_words = []
94
95      # documents contains all the news (positive and negative)
96      # eg.
97      #   ["full text of the news story 1 goes here", "pos"]
98      #   ["full text of the news story 2 goes here", "neg"]
99      documents = []
100
```

```
101     try:
102         documents_f = open("pickled_algos/documents-"+period+".
                pickle", "rb")
103         documents = pickle.load(documents_f)
104         documents_f.close()
105
106         all_words_f = open("pickled_algos/all_words-"+period+".
                pickle", "rb")
107         all_words = pickle.load(all_words_f)
108         all_words_f.close()
109     except FileNotFoundError:
110         # Populate the stopset so we can ignore these
111         stopset = set(stopwords.words('english'))
112
113         # Populate all_words and documents
114         for p in short_pos:
115             # Add the review to documents
116             documents.append((p, "pos"))
117
118             # Tokenize the review
119             words = word_tokenize(p)
120             pos = nltk.pos_tag(words)
121             for w in pos:
122                 # If the word is in our desired part-of-speech,
                        add it to all_words
123                 if w[0] not in stopset and w[1][0] in
                        allowed_word_types:
124                     all_words.append(w[0].lower())
125
126         # Tokenize negative reviews
127         for p in short_neg:
128             # Add the review to documents
129             documents.append((p, "neg"))
130
131             # Tokenize the review
132             words = word_tokenize(p)
```

```
133             pos = nltk.pos_tag(words)
134             for w in pos:
135                 if w[0] not in stopset and w[1] in
                        allowed_word_types:
136                     # If the word is in our desired part-of-
                            speech, add it to all_words
137                     all_words.append(w[0].lower())
138
139         # Take all reviews (positive and negative) and pickle
                them
140         save_documents = open("pickled_algos/documents-"+period+"
                .pickle", "wb")
141         pickle.dump(documents, save_documents)
142         save_documents.close()
143
144         # Take all words (tagged) and pickle them
145         all_words_pickle = open("pickled_algos/all_words-"+period
                +".pickle", "wb")
146         pickle.dump(all_words, all_words_pickle)
147         all_words_pickle.close()
148
149     # Take only the "interesting" words (ie. the ones that appear
            least often)
150     all_words = nltk.FreqDist(all_words)
151     # all_words.plot(int(len(all_words) * (1 / 3)))
152     #
153     # WHAT IS THE TOTAL SIZE OF ALL_WORDS
154     #
155     #
156     # THIS WAS WRONG...
157     # word_features = list(all_words.keys())[:5000]
158     # THIS MIGHT BE BETTER
159     word_features = [a for (a, b) in all_words.most_common(int(
            len(all_words) * (1 / 3)))]
160     # MIGHT BE A GOOD IDEA TO JUST MANUALLY CHOP OFF VALUES == 1,
            2, 3, 4 AND KEEP THE REST
```

```
161
162     # Take all words (positive and negative) and pickle them (top
            5000)
163     save_word_features = open("pickled_algos/word_features5k.
            pickle", "wb")
164     pickle.dump(word_features, save_word_features)
165     save_word_features.close()
166
167     # Parse the news story. If the word is in the "top 5000"
168     #   mark it as True, otherwise, leave it as False
169     #   eg.
170     #       {'have': False, 'uncover': True, ... }
171     def find_features(document):
172         words = word_tokenize(document)
173         features = {}
174         for w in word_features:
175             features[w] = (w in words)
176
177         return features
178
179     # featuresets contains the "top 5000" words: True if the word
            is in the document, otherwise False
180     #   Second field explains whether the document itself is
            postive or negative
181     # eg.
182     #   [
183     #       ({'dropped': False, 'rose': True, ... }, 'pos')
184     #       ({'dropped': True, 'rose': False, ... }, 'neg')
185     #   ]
186     featuresets = [(find_features(rev), category) for (rev,
            category) in documents]
187
188     logging.debug("Sample:\n" + str(featuresets[:1]))
189     random.shuffle(featuresets)
190     logging.debug(len(featuresets))
191
```

```
192     # Build a training set out of the first 90% of news stories
193     training_set = featuresets[:int(len(featuresets) * .9)]
194     logging.debug("Training\n" + str(training_set[:1]))
195
196     # Build a testing set out of the last 10% of news stories
197     testing_set = featuresets[int(len(featuresets) * .9):]
198     logging.debug("Testing\n" + str(testing_set[:1]))
199
200     # This is an attempt to analyze where my accuracy is getting
            it wrong and see if there's a pattern
201     #   that I can use to make my regular accuracy function
            better.
202     def accuracy_rewrite(classifier, gold):
203         # correct = [ l == r for ((fs, l), r) in zip(gold,results
                ) ]
204         # results = classifier.classify_many([fs for (fs,l) in
                gold])
205         # if correct:
206         #     return sum(correct)/len(correct)
207         # else:
208         #     return 0
209         errors = []
210         i = 0
211         for (name, tag) in gold:
212             guess = classifier.classify(name)
213             if guess != tag:
214                 errors.append((tag, guess, name))
215             i += 1
216         for (tag, guess, name) in errors:
217             print("TAG: ", tag)
218             print("GUESS: ", guess)
219             for a in name.keys():
220                 if name[a]:
221                     print("\t",a)
222             print("\n")
223
```

```
224     ################
225     ## Naive Bayes
226     #
227     # Train
228     classifier = nltk.NaiveBayesClassifier.train(training_set)
229     # Test
230     classifier_avg = (nltk.classify.accuracy(classifier,
            testing_set)) * 100
231     accuracy_rewrite(classifier, testing_set)
232     logging.debug("Original␣Naive␣Bayes␣Algo␣accuracy␣percent:␣{}
            ␣%".format(classifier_avg))
233     classifier.show_most_informative_features(15)
234     # most_informative_features = classifier.
            most_informative_features(15)
235     most_informative_features =
            show_most_informative_features_rewrite(classifier, 15)
236     # exit()
237
238     save_classifier = open("pickled_algos/originalnaivebayes5k.
            pickle", "wb")
239     pickle.dump(classifier, save_classifier)
240     save_classifier.close()
241
242     ################
243     ## Multinomial Naive Bayes
244     #
245     MNB_classifier = SklearnClassifier(MultinomialNB())
246     # Train
247     MNB_classifier.train(training_set)
248     # Test
249     MNB_classifier_avg = (nltk.classify.accuracy(MNB_classifier,
            testing_set)) * 100
250     logging.debug("MNB_classifier␣accuracy␣percent:␣{}␣%".format(
            MNB_classifier_avg))
251
252     save_classifier = open("pickled_algos/MNB_classifier5k.pickle
```

```
        ", "wb")
253     pickle.dump(MNB_classifier, save_classifier)
254     save_classifier.close()
255
256     ################
257     ## Bernoulli Naive Bayes
258     #
259     BernoulliNB_classifier = SklearnClassifier(BernoulliNB())
260     # Train
261     BernoulliNB_classifier.train(training_set)
262     # Test
263     BernoulliNB_classifier_avg = (nltk.classify.accuracy(
            BernoulliNB_classifier, testing_set)) * 100
264     logging.debug("BernoulliNB_classifier accuracy percent: {} %"
            .format(BernoulliNB_classifier_avg))
265
266     save_classifier = open("pickled_algos/
            BernoulliNB_classifier5k.pickle", "wb")
267     pickle.dump(BernoulliNB_classifier, save_classifier)
268     save_classifier.close()
269
270     ################
271     ## Logistic Regression
272     #
273     LogisticRegression_classifier = SklearnClassifier(
            LogisticRegression())
274     # Train
275     LogisticRegression_classifier.train(training_set)
276     # Test
277     LogisticRegression_classifier_avg = (nltk.classify.accuracy(
            LogisticRegression_classifier, testing_set)) * 100
278     logging.debug("LogisticRegression_classifier accuracy percent
            : {} %".format(LogisticRegression_classifier_avg))
279
280     save_classifier = open("pickled_algos/
            LogisticRegression_classifier5k.pickle", "wb")
```

```
281     pickle.dump(LogisticRegression_classifier, save_classifier)
282     save_classifier.close()
283
284     ################
285     ## Linear Support Vector Classification
286     #
287     LinearSVC_classifier = SklearnClassifier(LinearSVC())
288     # Train
289     LinearSVC_classifier.train(training_set)
290     # Test
291     LinearSVC_classifier_avg = (nltk.classify.accuracy(
            LinearSVC_classifier, testing_set)) * 100
292     logging.debug("LinearSVC_classifier accuracy percent: {} %".
            format(LinearSVC_classifier_avg))
293
294     save_classifier = open("pickled_algos/LinearSVC_classifier5k.
            pickle", "wb")
295     pickle.dump(LinearSVC_classifier, save_classifier)
296     save_classifier.close()
297
298     ################
299     ## Nu-Support Vector Classification
300     #
301     NuSVC_classifier = SklearnClassifier(NuSVC())
302     # Train
303     NuSVC_classifier.train(training_set)
304     # Test
305     NuSVC_classifier_avg = (nltk.classify.accuracy(
            NuSVC_classifier, testing_set)) * 100
306     logging.debug("NuSVC_classifier accuracy percent: {} %".
            format(NuSVC_classifier_avg))
307
308     save_classifier = open("pickled_algos/NuSVC_classifier5k.
            pickle", "wb")
309     pickle.dump(NuSVC_classifier, save_classifier)
310     save_classifier.close()
```

```
311
312         ################
313         ## Stochastic Gradient Descent Classification
314         #
315         SGDC_classifier = SklearnClassifier(SGDClassifier())
316         # Train
317         SGDC_classifier.train(training_set)
318         # Test
319         SGDC_classifier_avg = nltk.classify.accuracy(SGDC_classifier,
                testing_set) * 100
320         logging.debug("SGDClassifier accuracy percent: {}%".format(
               SGDC_classifier_avg))
321
322         save_classifier = open("pickled_algos/SGDC_classifier5k.
               pickle", "wb")
323         pickle.dump(SGDC_classifier, save_classifier)
324         save_classifier.close()
325
326         # Put the scores of all the classifiers in a list
327         voted_classifier_avg = [classifier_avg,
328                                 LinearSVC_classifier_avg,
329                                 MNB_classifier_avg,
330                                 BernoulliNB_classifier_avg,
331                                 LogisticRegression_classifier_avg]
332
333         # Take the basic average of all the classifiers
334         return (mean(voted_classifier_avg), most_informative_features
               , voted_classifier_avg)
335
336
337 def evaluateBigrams(posNews, negNews, period):
338         # Pull out ONLY the body of the story (a[9]).
339         #   Include a[6] for the headline
340         short_pos = []
341         short_neg = []
342         for a in posNews:
```

```
343             short_pos.append(str(a[9]))
344         for a in negNews:
345             short_neg.append(str(a[9]))
346 
347         # all_words contains all the words that are POS tagged in
                allowed_word_types
348         # eg.
349         #   ["word1", "word2", ... , "wordN"]
350         all_words = []
351 
352         # documents contains all the news (positive and negative)
353         # eg.
354         #   ["full text of the news story 1 goes here", "pos"]
355         #   ["full text of the news story 2 goes here", "neg"]
356         documents = []
357         logging.debug("HERE!")
358         try:
359             documents_f = open("pickled_algos/documents-bigram-"+
                    period+".pickle", "rb")
360             documents = pickle.load(documents_f)
361             documents_f.close()
362 
363             all_words_f = open("pickled_algos/all_words-bigram-"+
                    period+".pickle", "rb")
364             all_words = pickle.load(all_words_f)
365             all_words_f.close()
366         except FileNotFoundError:
367 
368             # Populate all_words and documents
369             for p in short_pos:
370                 # Add the review to documents
371                 documents.append((p, "pos"))
372 
373                 # Tokenize the review
374                 words = word_tokenize(p)
375                 pos = nltk.pos_tag(words)
```

```
376             for w in pos:
377                 all_words.append(w[0].lower())
378
379         # Tokenize negative reviews
380         for p in short_neg:
381             # Add the review to documents
382             documents.append((p, "neg"))
383
384             # Tokenize the review
385             words = word_tokenize(p)
386             pos = nltk.pos_tag(words)
387             for w in pos:
388                 all_words.append(w[0].lower())
389
390     # Take all reviews (positive and negative) and pickle them
391     save_documents = open("pickled_algos/documents-bigram-"+
            period+".pickle", "wb")
392     pickle.dump(documents, save_documents)
393     save_documents.close()
394
395     # Take all words (tagged) and pickle them
396     all_words_pickle = open("pickled_algos/all_words-bigram-"+
            period+".pickle", "wb")
397     pickle.dump(all_words, all_words_pickle)
398     all_words_pickle.close()
399
400     # Take only the "interesting" words (ie. the ones that appear
            least often)
401     all_words = BigramCollocationFinder.from_words(all_words)
402     # all_words.plot()
403     word_features = all_words.ngram_fd.most_common(int(len(
            all_words.ngram_fd)/3))
404     # word_features = dict([(ngram, True) for ngram in itertools.
            chain(all_words, word_features)])
405     word_features = dict([(a,True) for (a,b) in word_features])
406
```

```
407     # Take all words (positive and negative) and pickle them (top
            5000)
408     save_word_features = open("pickled_algos/word_features5k.
            pickle", "wb")
409     pickle.dump(word_features, save_word_features)
410     save_word_features.close()
411
412     # Parse the news story. If the word is in the "top 5000"
413     #   mark it as True, otherwise, leave it as False
414     #   eg.
415     #       {'have': False, 'uncover': True, ... }
416     def find_features(document):
417         words = word_tokenize(document)
418         all_words = BigramCollocationFinder.from_words(words)
419         features = {}
420         for w in word_features:
421             features[w] = (w in all_words.ngram_fd)
422
423         return features
424
425     # featuresets contains the "top 5000" words: True if the word
            is in the document, otherwise False
426     #   Second field explains whether the document itself is
            postive or negative
427     # eg.
428     #   [
429     #       ({'dropped': False, 'rose': True, ... }, 'pos')
430     #       ({'dropped': True, 'rose': False, ... }, 'neg')
431     #   ]
432     featuresets = [(find_features(rev), category) for (rev,
            category) in documents]
433
434     # logging.debug("Sample:\n" + str(featuresets[:1]))
435     random.shuffle(featuresets)
436     # logging.debug(len(featuresets))
437
```

```
438     # Build a training set out of the first 90% of news stories
439     training_set = featuresets[:int(len(featuresets) * .9)]
440     # logging.debug("Training\n" + str(training_set[:1]))
441
442     # Build a testing set out of the last 10% of news stories
443     testing_set = featuresets[int(len(featuresets) * .9):]
444     # logging.debug("Testing\n" + str(testing_set[:1]))
445
446     ################
447     ## Naive Bayes
448     #
449     # Train
450     classifier = nltk.NaiveBayesClassifier.train(training_set)
451     # Test
452     classifier_avg = (nltk.classify.accuracy(classifier,
            testing_set)) * 100
453     logging.debug("Original␣Naive␣Bayes␣Algo␣accuracy␣percent:␣{}
            ␣%".format(classifier_avg))
454     classifier.show_most_informative_features(50)
455
456     # most_informative_features = classifier.
            most_informative_features(15)
457     most_informative_features =
            show_most_informative_features_rewrite(classifier, 50)
458
459     save_classifier = open("pickled_algos/originalnaivebayes5k.
            pickle", "wb")
460     pickle.dump(classifier, save_classifier)
461     save_classifier.close()
462     return (classifier_avg, most_informative_features, [
            classifier_avg])
463     ################
464     ## Multinomial Naive Bayes
465     #
466     MNB_classifier = SklearnClassifier(MultinomialNB())
467     # Train
```

```
468     MNB_classifier.train(training_set)
469     # Test
470     MNB_classifier_avg = (nltk.classify.accuracy(MNB_classifier,
            testing_set)) * 100
471     logging.debug("MNB_classifier accuracy percent: {} %".format(
            MNB_classifier_avg))
472
473     save_classifier = open("pickled_algos/MNB_classifier5k.pickle
            ", "wb")
474     pickle.dump(MNB_classifier, save_classifier)
475     save_classifier.close()
476
477     ################
478     ## Bernoulli Naive Bayes
479     #
480     BernoulliNB_classifier = SklearnClassifier(BernoulliNB())
481     # Train
482     BernoulliNB_classifier.train(training_set)
483     # Test
484     BernoulliNB_classifier_avg = (nltk.classify.accuracy(
            BernoulliNB_classifier, testing_set)) * 100
485     logging.debug("BernoulliNB_classifier accuracy percent: {} %"
            .format(BernoulliNB_classifier_avg))
486
487     save_classifier = open("pickled_algos/
            BernoulliNB_classifier5k.pickle", "wb")
488     pickle.dump(BernoulliNB_classifier, save_classifier)
489     save_classifier.close()
490
491     ################
492     ## Logistic Regression
493     #
494     LogisticRegression_classifier = SklearnClassifier(
            LogisticRegression())
495     # Train
496     LogisticRegression_classifier.train(training_set)
```

```
497     # Test
498     LogisticRegression_classifier_avg = (nltk.classify.accuracy(
            LogisticRegression_classifier, testing_set)) * 100
499     logging.debug("LogisticRegression_classifier accuracy percent
            : {} %".format(LogisticRegression_classifier_avg))
500
501     save_classifier = open("pickled_algos/
            LogisticRegression_classifier5k.pickle", "wb")
502     pickle.dump(LogisticRegression_classifier, save_classifier)
503     save_classifier.close()
504
505     ################
506     ## Linear Support Vector Classification
507     #
508     LinearSVC_classifier = SklearnClassifier(LinearSVC())
509     # Train
510     LinearSVC_classifier.train(training_set)
511     # Test
512     LinearSVC_classifier_avg = (nltk.classify.accuracy(
            LinearSVC_classifier, testing_set)) * 100
513     logging.debug("LinearSVC_classifier accuracy percent: {} %".
            format(LinearSVC_classifier_avg))
514
515     save_classifier = open("pickled_algos/LinearSVC_classifier5k.
            pickle", "wb")
516     pickle.dump(LinearSVC_classifier, save_classifier)
517     save_classifier.close()
518
519     ################
520     ## Nu-Support Vector Classification
521     #
522     NuSVC_classifier = SklearnClassifier(NuSVC())
523     # Train
524     NuSVC_classifier.train(training_set)
525     # Test
526     NuSVC_classifier_avg = (nltk.classify.accuracy(
```

```
        NuSVC_classifier, testing_set)) * 100
527     logging.debug("NuSVC_classifier accuracy percent: {}%".
        format(NuSVC_classifier_avg))
528
529     save_classifier = open("pickled_algos/NuSVC_classifier5k.
        pickle", "wb")
530     pickle.dump(NuSVC_classifier, save_classifier)
531     save_classifier.close()
532
533     ################
534     ## Stochastic Gradient Descent Classification
535     #
536     SGDC_classifier = SklearnClassifier(SGDClassifier())
537     # Train
538     SGDC_classifier.train(training_set)
539     # Test
540     SGDC_classifier_avg = nltk.classify.accuracy(SGDC_classifier,
            testing_set) * 100
541     logging.debug("SGDClassifier accuracy percent: {}%".format(
        SGDC_classifier_avg))
542
543     save_classifier = open("pickled_algos/SGDC_classifier5k.
        pickle", "wb")
544     pickle.dump(SGDC_classifier, save_classifier)
545     save_classifier.close()
546
547     # Put the scores of all the classifiers in a list
548     voted_classifier_avg = [classifier_avg,
549                             LinearSVC_classifier_avg,
550                             MNB_classifier_avg,
551                             BernoulliNB_classifier_avg,
552                             LogisticRegression_classifier_avg]
553
554     # Take the basic average of all the classifiers
555     return (mean(voted_classifier_avg), most_informative_features
        , voted_classifier_avg)
```

```
556
557
558 if __name__ == '__main__':
559     logging.basicConfig(stream=sys.stderr, level=logging.DEBUG)
560     #logging.basicConfig(stream=sys.stderr, level=logging.INFO)
561     logging.info(str(datetime.now()))
562     try:
563         cnx = mysql.connector.connect(user='thesis-user',
                password='thesis', host='127.0.0.1', database='thesis'
                )
564         cursor = cnx.cursor()
565
566         start = ['2008/01/22', '2008/07/14', '2012/01/30']
567         end = ['2008/04/22', '2008/10/10', '2012/04/30']
568         period = ['up', 'down', 'avg']
569         for i in range(100):
570             logging.info("Iteration: {}".format(i))
571             for (i, item) in enumerate(start):
572                 dateStart = start[i]
573                 dateEnd = end[i]
574                 voted_classifier_avg = []
575                 voted_classifier_avg_extreme = []
576
577                 pos = readDB(cursor, dateStart, dateEnd)
578                 posExtreme = readDB(cursor, dateStart, dateEnd,
                    stock_overprevious=10)
579                 logging.info("Positive News Stories {} to {}: {}"
                    .format(dateStart, dateEnd, len(pos)))
580                 logging.info("Extreme Positive News Stories {} to
                    {}: {}".format(dateStart, dateEnd, len(
                    posExtreme)))
581
582                 neg = readDB(cursor, dateStart, dateEnd, neg=True
                    )
583                 negExtreme = readDB(cursor, dateStart, dateEnd,
                    neg=True, stock_overprevious=-10)
```

```
584                 logging.info("Negative␣News␣Stories␣{}␣to␣{}:␣{}"
                        .format(dateStart, dateEnd, len(neg)))
585                 logging.info("Extreme␣Negative␣News␣Stories␣{}␣to
                        ␣{}:␣{}".format(dateStart, dateEnd, len(
                        negExtreme)))
586
587                 # (avg, most_informative_features,
                        voted_classifier_avg) = evaluate(pos, neg)
588                 # logging.info("OVERALL ACCURACY PERCENT: {:f}
                        %".format(avg))
589                 # logging.info(str(datetime.now()))
590                 # (avgExtreme, most_informative_features_extreme,
                          voted_classifier_avg_extreme) = evaluate(
                        posExtreme, negExtreme)
591                 # logging.info("EXTREME OVERALL ACCURACY PERCENT:
                          {:f} %".format(avgExtreme))
592                 # logging.info(str(datetime.now()))
593
594                 choice = 'y'
595                 # choice = input("Bigram y/n: ")
596                 if choice == 'y':
597                     (avg, most_informative_features,
                            voted_classifier_avg) = evaluateBigrams(
                            pos, neg, period[i])
598                     logging.info("OVERALL␣ACCURACY␣PERCENT:␣{:f}␣
                            %".format(avg))
599                     logging.info(str(datetime.now()))
600                     (avgExtreme,
                            most_informative_features_extreme,
                            voted_classifier_avg_extreme) =
                            evaluateBigrams(posExtreme,
601
602
```

```
603                 logging.info("EXTREME␣OVERALL␣ACCURACY␣
                        PERCENT:␣{:f}␣%".format(avgExtreme))
604                 logging.info(str(datetime.now()))
605                 sql = ("INSERT␣INTO␣results_bigram␣"
606                        "(dateStart,dateEnd,avg,avgExtreme,
                            features,featuresExtreme,"
607                        "classifier_avg,␣
                            classifier_avg_extreme)␣"
608                        "VALUES(%s,␣%s,␣%s,␣%s,␣%s,␣%s,␣%s
                            )")
609                 cursor.execute(sql,
610                                (dateStart, dateEnd, avg,
                                    avgExtreme,
611                                 str(most_informative_features
                                    ), str(
                                    most_informative_features_extreme
                                    ),
612                                 voted_classifier_avg[0],
                                    voted_classifier_avg_extreme
                                    [0])
613                                )
614                 cnx.commit()
615             else:
616                 (avg, most_informative_features,
                        voted_classifier_avg) = evaluate(pos, neg,
                         period[i])
617                 logging.info("OVERALL␣ACCURACY␣PERCENT:␣{:f}␣
                        %".format(avg))
618                 logging.info(str(datetime.now()))
```

```
619                 (avgExtreme,
                        most_informative_features_extreme,
                        voted_classifier_avg_extreme) = evaluate(
                        posExtreme,
620
621
622                 logging.info("EXTREME OVERALL ACCURACY
                        PERCENT: {:f} %".format(avgExtreme))
623                 logging.info(str(datetime.now()))
624                 exit()
625                 sql = ("INSERT INTO results "
626                        "(dateStart,dateEnd,avg,avgExtreme,
                            features,featuresExtreme,"
627                        "classifier_avg, 
                            LinearSVC_classifier_avg,
                            MNB_classifier_avg, 
                            BernouelliNB_classifier_avg,"
628                        "LogisticRegression_classifier_avg,"
629                        "classifier_avg_extreme, 
                            LinearSVC_classifier_avg_extreme,
                            MNB_classifier_avg_extreme, 
                            BernouelliNB_classifier_avg_extreme
                            ,"
630                        "
                            LogisticRegression_classifier_avg_extreme
                            ) "
```

```
631                         "VALUES(%s , %s , %s , %s , %s , %s , %s
                                , %s , %s , %s , %s , %s , %s , %s )")
632                     cursor.execute(sql,
633                                    (dateStart, dateEnd, avg,
                                       avgExtreme,
634                                     str(most_informative_features
                                        ), str(
                                        most_informative_features_extreme
                                        ),
635                                     voted_classifier_avg[0],
                                        voted_classifier_avg[1],
                                        voted_classifier_avg[2],
636                                     voted_classifier_avg[3],
                                        voted_classifier_avg[4],
637                                     voted_classifier_avg_extreme
                                        [0],
                                        voted_classifier_avg_extreme
                                        [1],
638                                     voted_classifier_avg_extreme
                                        [2],
                                        voted_classifier_avg_extreme
                                        [3],
639                                     voted_classifier_avg_extreme
                                        [4])
640                                    )
641                     cnx.commit()
642             exit("Manual Exit Line 641")
643
644     except mysql.connector.Error as err:
645         if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
646             logging.error("Something is wrong with your user name
                  or password")
647         elif err.errno == errorcode.ER_BAD_DB_ERROR:
648             logging.error("Database does not exist")
649         else:
650             logging.error(err)
```

```
651         else:
652             cnx.close()
653 
654 
655 # Some options to try to improve accuracy
656 #   1. Manually classify news articles rather than depend on the
        closing price that day
657 #   2. http://thinknook.com/10-ways-to-improve-your-
        classification-algorithm-performance-2013-01-21/
658 #       Try to rewrite accuracy to print out ones that don't
        match - then investigate
659 #       Do bigrams instead of POS tagging
660 
661 # Run date == NULL - No stopword filtering, no bigram collections
662 # Run date ~ August 29, 2016 == stopword filtering, no bigram
        collections
663 # Run date ~ September 4 == Top 500 words instead of top 5000
664 # Run date == October 1 == Top 3000 words (but for real this time
        )
665 # Run date == October 10 == Used pickling to speed things up
666 
667 #
668 #
669 # Analyze the distribution to find the right cut-off, 500? 5000?
        Is it 5000/10000 words?! Potentially do not make a cut-off at
        all
670 #   Analyze the "5000" and determine if maybe they show up
        exactly 0 times perhaps in the test set.
671 # Analyze the most informative features and see if that can help
        determine where to cut off
672 # Extreme needs to have a dataset that matches non-extreme or
        reduce the list to 500 or so
673 # COMBINE POSTIVE AND NEGATEIVE DATE RANGES (eg. avoid allowing
        it to pick positive by default and on average it would be
        correct)
674 # # Continue with bigram pursuit: Adverb Adjective or Adjective-
```

```
    noun (eg. low return, high volatility)
# Larget test set to get better hits in my test set
#
#
```

A.3 test.py

```
#!/usr/bin/python3
import os
import glob
os.chdir("/media/sf_School/Exports")
filelist = glob.glob("*.csv")
TRADING_DAYS = 252
SPLITS = [  1/4,  # 3 months
            1/2,   # 6 months
            1,    # 12 months
            2,    # 2 years
            3,      # 3 years
            5,    # 5 years
            10 ]  # 10 years
for i in filelist:
    with open(i) as file:
        data = file.read().split('\n')
    print(i)

    with open("OUTPUT" + i, "w") as file:
        for j in SPLITS:
            max, min, avg = [-10000, 0], [10000, 0], [10000, 0]

            file.write("PERIOD,END_DATE,START_DATE,END_VALUE,
                START_VALUE,DIFFERENCE\n")
            for k in range(len(data) - int(TRADING_DAYS*j) - 1):
                begin = data[k].split(',')
                end = data[k+int(TRADING_DAYS*j)].split(',')
                difference = float(end[1]) - float(begin[1])
                if avg[0] == 10000:
```

```
29                      avg = [abs(difference), k]
30                  if max[0] < difference:
31                      max = [difference, k]
32                  if min[0] > difference:
33                      min = [difference, k]
34                  if avg[0] > abs(difference):
35                      avg = [abs(difference), k]
36                  output = "{},{},{},{},{},{:.2f}".format(k,end[0],
                        begin[0],end[1],begin[1],difference)
37                  file.write(output + '\n')
38
39              file.write("Jump by " + str(j) + ',' + str(max) + ','
                    + str(min) + ',' + str(avg) + "\n\n")
40              # exit(0)
41          # input("Enter to continue...")
```

A.4 Database Schema

```
1  CREATE DATABASE IF NOT EXISTS `thesis` /*!40100 DEFAULT
       CHARACTER SET latin1 */;
2  USE `thesis`;
3  -- MySQL dump 10.13  Distrib 5.5.52, for debian-linux-gnu (x86_64
       )
4  --
5  -- Host: 127.0.0.1    Database: thesis
6  -- ------------------------------------------------------
7  -- Server version       5.5.52-0ubuntu0.14.04.1
8
9  /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
10 /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS
       */;
11 /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
12 /*!40101 SET NAMES utf8 */;
13 /*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
14 /*!40103 SET TIME_ZONE='+00:00' */;
15 /*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0
```

```
    */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
    FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='
    NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `news`
--

DROP TABLE IF EXISTS `news`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `news` (
  `idnews` int(11) NOT NULL AUTO_INCREMENT,
  `theadline` varchar(255) DEFAULT NULL,
  `tauthor` varchar(255) DEFAULT NULL,
  `tdate` datetime DEFAULT NULL,
  `tbody` blob,
  PRIMARY KEY (`idnews`)
) ENGINE=InnoDB AUTO_INCREMENT=5864 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `results`
--

DROP TABLE IF EXISTS `results`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `results` (
  `idresults` int(11) NOT NULL AUTO_INCREMENT,
  `runDate` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `dateStart` datetime DEFAULT NULL,
  `dateEnd` datetime DEFAULT NULL,
```

```
49    `avg` double DEFAULT NULL,
50    `avgExtreme` double DEFAULT NULL,
51    `features` blob,
52    `featuresExtreme` blob,
53    `classifier_avg` double DEFAULT NULL,
54    `LinearSVC_classifier_avg` double DEFAULT NULL,
55    `MNB_classifier_avg` double DEFAULT NULL,
56    `BernouelliNB_classifier_avg` double DEFAULT NULL,
57    `LogisticRegression_classifier_avg` double DEFAULT NULL,
58    `classifier_avg_extreme` double DEFAULT NULL,
59    `LinearSVC_classifier_avg_extreme` double DEFAULT NULL,
60    `MNB_classifier_avg_extreme` double DEFAULT NULL,
61    `BernouelliNB_classifier_avg_extreme` double DEFAULT NULL,
62    `LogisticRegression_classifier_avg_extreme` datetime DEFAULT
        NULL,
63    PRIMARY KEY (`idresults`),
64    UNIQUE KEY `idresults_UNIQUE` (`idresults`)
65  ) ENGINE=InnoDB AUTO_INCREMENT=1771 DEFAULT CHARSET=latin1;
66  /*!40101 SET character_set_client = @saved_cs_client */;
67
68  --
69  -- Table structure for table `results_bigram`
70  --
71
72  DROP TABLE IF EXISTS `results_bigram`;
73  /*!40101 SET @saved_cs_client     = @@character_set_client */;
74  /*!40101 SET character_set_client = utf8 */;
75  CREATE TABLE `results_bigram` (
76    `idresults` int(11) NOT NULL AUTO_INCREMENT,
77    `runDate` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
78    `dateStart` datetime DEFAULT NULL,
79    `dateEnd` datetime DEFAULT NULL,
80    `avg` double DEFAULT NULL,
81    `avgExtreme` double DEFAULT NULL,
82    `features` blob,
83    `featuresExtreme` blob,
```

```
84    `classifier_avg` double DEFAULT NULL,
85    `LinearSVC_classifier_avg` double DEFAULT NULL,
86    `MNB_classifier_avg` double DEFAULT NULL,
87    `BernouelliNB_classifier_avg` double DEFAULT NULL,
88    `LogisticRegression_classifier_avg` double DEFAULT NULL,
89    `classifier_avg_extreme` double DEFAULT NULL,
90    `LinearSVC_classifier_avg_extreme` double DEFAULT NULL,
91    `MNB_classifier_avg_extreme` double DEFAULT NULL,
92    `BernouelliNB_classifier_avg_extreme` double DEFAULT NULL,
93    `LogisticRegression_classifier_avg_extreme` datetime DEFAULT
        NULL,
94    PRIMARY KEY (`idresults`),
95    UNIQUE KEY `idresults_UNIQUE` (`idresults`)
96  ) ENGINE=InnoDB AUTO_INCREMENT=1774 DEFAULT CHARSET=latin1;
97  /*!40101 SET character_set_client = @saved_cs_client */;
98
99  --
100 -- Table structure for table `stock_data`
101 --
102
103 DROP TABLE IF EXISTS `stock_data`;
104 /*!40101 SET @saved_cs_client     = @@character_set_client */;
105 /*!40101 SET character_set_client = utf8 */;
106 CREATE TABLE `stock_data` (
107   `id` int(11) NOT NULL AUTO_INCREMENT,
108   `stock_name` varchar(45) DEFAULT NULL,
109   `stock_date` datetime DEFAULT NULL,
110   `stock_value` decimal(5,2) DEFAULT NULL,
111   `stock_overprevious` decimal(5,2) DEFAULT NULL,
112   PRIMARY KEY (`id`),
113   UNIQUE KEY `id_UNIQUE` (`id`),
114   UNIQUE KEY `date_UNIQUE` (`stock_date`)
115 ) ENGINE=InnoDB AUTO_INCREMENT=2534 DEFAULT CHARSET=latin1;
116 /*!40101 SET character_set_client = @saved_cs_client */;
```

A.5 clean_file.py

```
1  import string
2  from nltk import pos_tag, word_tokenize
3  import numpy as np
4  import re
5  import tensorflow as tf
6  from sklearn.utils import shuffle
7  import json
8  import os
9  import os.path as path
10 from pathlib import Path
11 import random
12
13 root_path = Path(__file__).parents[2]
14
15 POS_FILES_PATH=os.path.abspath(os.path.join(root_path, 'data/raw/
       rt-polarity.pos'))
16 NEG_FILES_PATH=os.path.abspath(os.path.join(root_path, 'data/raw/
       rt-polarity.neg'))
17
18 POS_FILES_CLENED_PATH=os.path.abspath(os.path.join(root_path, '
       data/preprocessed/pos_review_cleaned.txt'))
19 NEG_FILES_CLENED_PATH=os.path.abspath(os.path.join(root_path, '
       data/preprocessed/neg_review_cleaned.txt'))
20
21 FILES_CLEANED_LABELED=os.path.abspath(os.path.join(root_path, '
       data/preprocessed/reviews_labeled.txt'))
22 FINAL_FILE=os.path.abspath(os.path.join(root_path, 'data/
       preprocessed/reviews_shuffled.txt'))
23
24 TRAIN_DATA=os.path.abspath(os.path.join(root_path, 'data/
       preprocessed/train.txt'))
25 TEST_DATA=os.path.abspath(os.path.join(root_path, 'data/
       preprocessed/test.txt'))
26
```

```
27 WORD2IDX_PATH=os.path.abspath(os.path.join(root_path, 'data/
       preprocessed/word2idx.txt'))
28
29
30 TEST_SIZE=0.2
31
32
33 def clean_data():
34     '''Remove numbers and other special characters from the
           sentences in the raw files
35     and write cleaned sentences to new files.'''
36
37
38     files_path=[POS_FILES_PATH,NEG_FILES_PATH]
39     cleaned_files_path=[POS_FILES_CLENED_PATH,
           NEG_FILES_CLENED_PATH]
40
41     for file, cleaned_file in zip(files_path, cleaned_files_path)
           :
42
43         with open(cleaned_file, 'w') as writer:
44             for line in open(file,'r',encoding='iso-8859-15'):
45                 line=line.rstrip()
46                 if line:
47                     line=re.sub('[^A-Za-z0-9]+', ' ', line)
48                     line=line.lower()
49                     if len(line)>1:
50                         writer.write(line+'\n')
51
52 def write_word2idx():
53     '''Assign a number to each unique word. Create a dictionary
           where each key is a unique word with an integer
54     as value and write this dictionary to a file. '''
55
56     cleaned_files_path=[POS_FILES_CLENED_PATH,
           NEG_FILES_CLENED_PATH]
```

```
57
58      word2idx={'PAD':0}
59      num_words=1
60
61      for file in cleaned_files_path:
62          for line in open(file):
63              tokens=word_tokenize(line)
64              for token in tokens:
65                  if token not in word2idx:
66                      word2idx[token]=num_words
67                      num_words+=1
68
69      with open(WORD2IDX_PATH, 'w') as outfile:
70          json.dump(word2idx, outfile)
71
72
73  def add_label():
74      '''Add a label of either 0 (genative) or 1 (positive) to each
            review and write it to a new .txt-file'''
75
76      cleaned_files_path=[NEG_FILES_CLENED_PATH,
            POS_FILES_CLENED_PATH]
77      labels=[0,1]
78
79      with open(FILES_CLEANED_LABELED, 'w') as writer:
80          for file, label in zip(cleaned_files_path, labels):
81              for line in open(file):
82                  line=line.rstrip('\n') + '| ' + str(label)+'\n'
83                  writer.write(line)
84
85
86  def shuffle_file():
87      '''Shuffle the data in the file '''
88      with open(FILES_CLEANED_LABELED,'r') as source:
89          data = [ (random.random(), line) for line in source ]
90      data.sort()
```

```
91      with open(FINAL_FILE,'w') as target:
92          for _, line in data:
93              target.write( line )
94
95
96
97
98
99  def count_labels():
100     '''Count the number of positive and negative reviews '''
101
102     files=[TRAIN_DATA, TEST_DATA]
103
104     for file in files:
105
106         num_positives=0
107         num_negatives=0
108
109         for line in open(file):
110
111             splitted_line=line.split('|')
112             label_encoded=int(splitted_line[1].rstrip('\n'))
113
114             if label_encoded==0:
115                 num_negatives+=1
116             elif label_encoded==1:
117                 num_positives+=1
118
119         print('File␣%s␣contains␣%s␣positives␣and␣%s␣negatives'%(
              file, num_positives, num_negatives))
120
121 def print_word2idx():
122     '''Print the word2idx dictionary '''
123     with open(WORD2IDX_PATH) as json_file:
124         word2idx = json.load(json_file)
125
```

```
126     cleaned_files_path=[POS_FILES_CLENED_PATH,
          NEG_FILES_CLENED_PATH]
127
128     for file in cleaned_files_path:
129         for line in open(file):
130             tokens=word_tokenize(line)
131             if len(tokens)>1:
132                 for token in tokens:
133                     word2idx[token]
134
135 def file_len(fname):
136     '''Count the number of lines in the file '''
137     with open(fname) as f:
138         for i, l in enumerate(f):
139             pass
140     return i + 1
141
142 def train_test_split():
143     '''Split the data into training and testing set '''
144
145     n_lines=file_len(FINAL_FILE)
146     n_train=int(n_lines*(1.0-TEST_SIZE))
147
148     train_writer= open(TRAIN_DATA, 'w')
149     test_writer= open(TEST_DATA, 'w')
150
151     with open(FINAL_FILE) as f:
152         for i, l in enumerate(f):
153             if i<n_train:
154                 train_writer.write(l)
155             else:
156                 test_writer.write(l)
157
158
159 if __name__ == "__main__":
160
```

```
161         clean_data()
162         write_word2idx()
163         add_label()
164         shuffle_file()
165         #print_word2idx()
166         train_test_split()
167         count_labels()
```

A.6 tf_record_writer.py

```
1  import os
2  import tensorflow as tf
3  import sys
4  from scipy.misc import imread, imsave, imresize
5  from random import shuffle
6  import numpy as np
7  import matplotlib.pyplot as plt
8  import json
9  from nltk import pos_tag, word_tokenize
10 from pathlib import Path
11
12 root_path = Path(__file__).parents[2]
13
14
15 WORD2IDX_PATH=os.path.abspath(os.path.join(root_path, 'data/
       preprocessed/word2idx.txt'))
16
17 TRAIN_DATA=os.path.abspath(os.path.join(root_path, 'data/
       preprocessed/train.txt'))
18 TEST_DATA=os.path.abspath(os.path.join(root_path, 'data/
       preprocessed/test.txt'))
19
20 OUTPUT_DIR=os.path.abspath(os.path.join(root_path, 'data/
       tf_records'))
21
22
```

```
23 def _get_tf_filename(output_dir,name_tf_file,num_tf_file):
24     return '%s/%s_%i.tfrecord' % (output_dir, name_tf_file,
       num_tf_file)
25
26
27 def int64_feature(value):
28     if not isinstance(value, list):
29         value = [value]
30     return tf.train.Feature(int64_list=tf.train.Int64List(value=
       value))
31
32 def int64_feature2(value):
33     return tf.train.Feature(int64_list=tf.train.Int64List(value=
       value))
34
35 def float_feature(value):
36     if not isinstance(value, list):
37         value = [value]
38     return tf.train.Feature(float_list=tf.train.FloatList(value=
       value))
39
40 def bytes_feature(value):
41     if not isinstance(value, list):
42         value = [value]
43     return tf.train.Feature(bytes_list=tf.train.BytesList(value=
       value))
44
45
46
47 def _add_to_tf_records(line, tf_writer, word2idx):
48
49     splitted_line=line.split('|')
50     line=splitted_line[0]
51     label=int(splitted_line[1].rstrip('\n'))
52
53     if label==1:
```

```
54          label='positiv'
55          label_encoded=[0,1]
56      elif label==0:
57          label='negativ'
58          label_encoded=[1,0]
59
60      idx_sequence=[]
61      tokens=word_tokenize(line)
62
63
64      for token in tokens:
65          try:
66              idx=word2idx[token]
67          except KeyError:
68              print('token: %s could not be found in the dictionary
                    .'%token)
69              continue
70
71          idx_sequence.append(idx)
72
73      idx_sequence=np.array(idx_sequence)
74      seq_length=len(idx_sequence)
75      idx_sequence=idx_sequence.tostring()
76
77      example = tf.train.Example(features=tf.train.Features(feature
            ={'text_line/encoded': bytes_feature(idx_sequence),
78                                                                  '
                                                                    text_l
                                                                    /
                                                                    seq_le
                                                                    '
                                                                    :
                                                                    int64_
                                                                    (
                                                                    seq_le
```

```
            )
            ,
79          ,
            label
            /
            label
            ,
            :
            bytes_
            (
            tf
            .
            compat
            .
            as_byt
            (
            label
            )
            )
            ,
80          ,
            label
            /
            encode
            ,
            :
            int64_
            (
            label_
            )
81          })
            )
```

```
82
83      tf_writer.write(example.SerializeToString())
84
85  def run(output_dir, name_tf_file, data):
86
87      if not tf.gfile.Exists(output_dir):
88          tf.gfile.MakeDirs(output_dir)
89
90      with open(WORD2IDX_PATH) as json_file:
91          word2idx = json.load(json_file)
92
93
94      num_tf_file=0
95
96      tfrecords_filename=_get_tf_filename(output_dir,name_tf_file,
            num_tf_file)
97
98      with tf.python_io.TFRecordWriter(tfrecords_filename) as
            tf_writer:
99
100         for line in open(data):
101             _add_to_tf_records(line, tf_writer, word2idx)
102
103
104     print('\nFinished converting the text file')
105
106
107
108
109 if __name__ == "__main__":
110
111     run(OUTPUT_DIR,'training_file', TRAIN_DATA)
112     run(OUTPUT_DIR,'test_file', TEST_DATA)
```

A.7 train.py

```
1  """
2  Trains a LSTM network to perform Sentiment Analysis
3
4  Created on Wed Dec 26 18:38:43 2018
5
6  @author: Artem Oppermann
7  """
8  import json
9  import os
10 import tensorflow as tf
11
12 from data.dataset import get_training_data, get_test_data
13 from models.train_model import TrainModel
14 from data.utils import show_sample
15
16
17 tf.app.flags.DEFINE_string('train_path', os.path.abspath(os.path.
       join(os.path.dirname( "__file__" ), '..', 'data/tf_records/
       training_file_0.tfrecord')),
18                            'Path for the training data.')
19 tf.app.flags.DEFINE_string('test_path', os.path.abspath(os.path.
       join(os.path.dirname("__file__"), '..', 'data/tf_records/
       test_file_0.tfrecord')),
20                            'Path for the test data.')
21
22 tf.app.flags.DEFINE_string('word2idx', os.path.abspath(os.path.
       join(os.path.dirname("__file__"), '..', 'data/preprocessed/
       word2idx.txt')),
23                            'Path for the word2idx dictionary.')
24
25 tf.app.flags.DEFINE_string('checkpoints_path', os.path.abspath(os
       .path.join(os.path.dirname( __file__ ), '..', 'checkpoints/
       model.ckpt')),
26                            'Path for the test data.')
27
28 tf.app.flags.DEFINE_integer('num_epoch', 1000,
```

```
29                              'Number of training epoch.'
30                              )
31 tf.app.flags.DEFINE_integer('batch_size', 32,
32                              'Batch size of the training set.'
33                              )
34 tf.app.flags.DEFINE_float('learning_rate', 0.0005,
35                             'Learning rate of optimizer.'
36                             )
37
38 tf.app.flags.DEFINE_string('architecture', 'unidirectional',
39                             'Type of LSTM-Architecture, choose
                                 between "unidirectional" or "
                                 bidirectional"'
40                             )
41
42 tf.app.flags.DEFINE_integer('lstm_units', 100,
43                              'Number of the LSTM hidden units.'
44                              )
45
46 tf.flags.DEFINE_float('dropout_keep_prob', 0.5,
47                       '0<dropout_keep_prob<=1. Dropout keep-
                          probability')
48
49 tf.app.flags.DEFINE_integer('embedding_size', 100,
50                              'Dimension of the embedding vector
                                 for the vocabulary.'
51                              )
52 tf.app.flags.DEFINE_integer('num_classes', 2,
53                              'Number of output classes.'
54                              )
55
56 tf.app.flags.DEFINE_integer('n_train_samples', 8529,
57                              'Number of all training sentences.'
58                              )
59
60 tf.app.flags.DEFINE_integer('n_test_samples', 2133,
```

```
61                                  'Number of all training sentences.'
62                                  )
63 tf.app.flags.DEFINE_float('required_acc_checkpoint', 0.7,
64                               'The accuracy on the test set that must
                                    be achieved, before any checkpoints
                                    are saved.'
65                               )
66
67
68 FLAGS = tf.app.flags.FLAGS
69
70
71
72 def main(_):
73
74     with open(FLAGS.word2idx) as json_file:
75         word2idx = json.load(json_file)
76
77     training_graph=tf.Graph()
78
79     with training_graph.as_default():
80
81         train_model=TrainModel(FLAGS, len(word2idx))
82
83         training_dataset=get_training_data(FLAGS)
84         test_dataset=get_test_data(FLAGS)
85
86         iterator_train = training_dataset.
              make_initializable_iterator()
87         iterator_test=test_dataset.make_initializable_iterator()
88
89         x_train, y_train, _, seq_length_train = iterator_train.
              get_next()
90         x_test, y_test, _, seq_length_test =iterator_test.
              get_next()
91
```

```
92          dropout_keep_prob = tf.placeholder(tf.float32, name='
                dropout_keep_prob')
93
94          logits, probs=train_model.compute_prediction(x_train,
                seq_length_train, dropout_keep_prob, reuse_scope=False
                )
95          loss=train_model.compute_loss(logits, y_train)
96          train_op=train_model.train(loss)
97          accuracy_train = train_model.compute_accuracy(probs,
                y_train)
98
99          x=tf.identity(x_test)
100         logits_test, probs_test=train_model.compute_prediction(
                x_test, seq_length_test, dropout_keep_prob,
                reuse_scope=True)
101         accuracy_test = train_model.compute_accuracy(probs_test,
                y_test)
102
103         saver=tf.train.Saver()
104
105     with tf.Session(graph=training_graph) as sess:
106
107         sess.run(tf.global_variables_initializer())
108
109         n_batches=int(FLAGS.n_train_samples/FLAGS.batch_size)
110
111         for epoch in range(FLAGS.num_epoch):
112
113             sess.run(iterator_train.initializer)
114             sess.run(iterator_test.initializer)
115
116             traininig_loss=0
117             training_acc=0
118
119
120             feed_dict={dropout_keep_prob:0.5}
```

```
121
122
123             for n_batch in range(0, n_batches):
124
125                 _, l, acc, logits_, probs_=sess.run((train_op,
                        loss, accuracy_train, logits, probs),
                        feed_dict)
126
127                 traininig_loss+=l
128                 training_acc+=acc
129
130
131             feed_dict={dropout_keep_prob:1.0}
132
133             acc_avg_test=sess.run(accuracy_test, feed_dict)
134
135             loss_avg=traininig_loss/n_batches
136             acc_avg_train=training_acc/n_batches
137
138             print('epoch_nr: %i, train_loss: %.3f, train_acc: %.3
                    f, test_acc: %.3f'%(epoch, loss_avg, acc_avg_train
                    , acc_avg_test))
139
140             traininig_loss=0
141             training_acc=0
142
143             show_sample(FLAGS, sess, logits_test, probs_test,
                    dropout_keep_prob, x)
144
145             if FLAGS.required_acc_checkpoint>0.70:
146                 saver.save(sess, FLAGS.checkpoints_path)
147
148
149
150 if __name__ == "__main__":
151
```

```
152         tf.app.run()
```

Appendix B

Complete Tables

B.1 Stock Selection

B.1.1 3 Months

Fund	Value	Normalized	From	To
S&P 500 Materials	60.55	-0.41020997	2009-03-03	2009-06-02
S&P 500 Energy	121.53	1.470715888	2008-01-22	2008-04-22
S&P 500 Financials	84.35	0.323900155	2009-03-05	2009-06-04
S&P 500 Health Care	117.98	1.361216268	2014-10-16	2015-01-16
S&P 500 Industrials	73.91	0.001878739	2009-03-09	2009-06-08
S&P 500 Information Technology	113.86	1.23413502	2015-08-25	2015-11-23
S&P 500 Telecommunications Services	25.35	-1.495952676	2007-03-05	2007-06-04
S&P 500 Utilities	36.3	-1.158200329	2014-09-29	2014-12-29
S&P 500 Consumer Staples	58.03	-0.487939278	2012-12-28	2013-04-02
S&P 500 Consumer Discretionary	83.37	0.293672091	2015-08-25	2015-11-23
S&P 500 Real Estate	37.11	-1.133215908	2006-11-03	2007-02-07
Mean	*73.84909091*			
Std Dev	*32.42020399*			

Table B.1: 3 Months - Best Performance

Fund	Value	Normalized	From	To
S&P 500 Materials	-131.82	-0.112780631	2008-08-22	2008-11-20
S&P 500 Energy	-272.44	-2.4544093	2008-07-14	2008-10-10
S&P 500 Financials	-151.96	-0.44815541	2008-08-22	2008-11-20
S&P 500 Health Care	-123.51	0.025598933	2008-08-22	2008-11-20
S&P 500 Industrials	-142.64	-0.292957151	2008-08-22	2008-11-20
S&P 500 Information Technology	-171.56	-0.774538015	2008-08-22	2008-11-20
S&P 500 Telecommunications Services	-39.44	1.425547177	2008-07-14	2008-10-10
S&P 500 Utilities	-72.76	0.870696749	2008-07-14	2008-10-10
S&P 500 Consumer Staples	-70.58	0.906998488	2008-08-22	2008-11-20
S&P 500 Consumer Discretionary	-112.72	0.205275887	2008-08-22	2008-11-20
S&P 500 Real Estate	-86.09	0.648723273	2008-08-22	2008-11-20
Mean	*-125.0472727*			
Std Dev	*60.05222002*			

Table B.2: 3 Months - Worst performance

Fund	Value	Normalized	From	To
S&P 500 Materials	0	-0.516934138	2013-01-03	2013-04-05
S&P 500 Energy	0.09	2.894831171	2012-01-30	2012-04-30
S&P 500 Financials	0.03	0.620320965	2014-10-28	2015-01-29
S&P 500 Health Care	0.02	0.241235931	2008-06-24	2008-09-23
S&P 500 Industrials	0.01	-0.137849103	2014-05-15	2014-08-14
S&P 500 Information Technology	0.01	-0.137849103	2011-01-20	2011-04-20
S&P 500 Telecommunications Services	0	-0.516934138	2011-08-02	2011-10-31
S&P 500 Utilities	0	-0.516934138	2011-01-11	2011-04-12
S&P 500 Consumer Staples	-0.01	-0.896019172	2013-12-20	2014-03-25
S&P 500 Consumer Discretionary	0	-0.516934138	2006-03-08	2006-06-07
S&P 500 Real Estate	0	-0.516934138	2013-03-11	2013-06-10
Mean	*0.013636364*			
Std Dev	*0.026379306*			

Table B.3: 3 Months - Most Average performance

B.1.2 6 Months

Fund	Value	Normalized	From	To
S&P 500 Materials	72.73	-0.510096442	2009-03-02	2009-08-28
S&P 500 Energy	196.77	2.122363141	2010-08-26	2011-02-25
S&P 500 Financials	107.66	0.231211307	2009-03-06	2009-09-03
S&P 500 Health Care	155.36	1.243532511	2014-10-16	2015-04-20
S&P 500 Industrials	87.01	-0.207036761	2009-03-09	2009-09-04
S&P 500 Information Technology	131.92	0.746073203	2011-10-03	2012-04-03
S&P 500 Telecommunications Services	36.46	-1.279842565	2006-11-27	2007-05-31
S&P 500 Utilities	43.09	-1.139136284	2014-08-06	2015-02-05
S&P 500 Consumer Staples	79.45	-0.367480121	2012-11-14	2013-05-17
S&P 500 Consumer Discretionary	103.07	0.133799267	2014-10-13	2015-04-15
S&P 500 Real Estate	50.9	-0.973387257	2006-08-08	2007-02-08
Mean	96.76545455			
Std Dev	47.11943188			

Table B.4: 6 Months - Best Performance

Fund	Value	Normalized	From	To
S&P 500 Materials	-159.76	-0.163166024	2008-05-23	2008-11-20
S&P 500 Energy	-316.88	-2.389623495	2008-05-23	2008-11-20
S&P 500 Financials	-211.8	-0.900595039	2008-09-03	2009-03-05
S&P 500 Health Care	-137.97	0.145607577	2015-08-10	2016-02-09
S&P 500 Industrials	-179.06	-0.436655263	2008-08-28	2009-03-02
S&P 500 Information Technology	-181.02	-0.4644293	2008-05-23	2008-11-20
S&P 500 Telecommunications Services	-57.21	1.290011252	2008-05-23	2008-11-20
S&P 500 Utilities	-73.09	1.064984873	2008-05-23	2008-11-20
S&P 500 Consumer Staples	-93.63	0.773924305	2008-09-08	2009-03-10
S&P 500 Consumer Discretionary	-120.61	0.391606188	2008-05-23	2008-11-20
S&P 500 Real Estate	-99.67	0.688334927	2008-09-03	2009-03-05
Mean	-148.2454545			
Std Dev	70.56950428			

Table B.5: 6 Months - Worst Perfomance

Fund	Value	Normalized	From	To
S&P 500 Materials	0	-0.738455971	2008-01-24	2008-07-24
S&P 500 Energy	0.04	0.971652593	2012-10-09	2013-04-15
S&P 500 Financials	0.04	0.971652593	2010-06-03	2010-12-01
S&P 500 Health Care	-0.03	-2.021037393	2015-05-07	2015-11-04
S&P 500 Industrials	0.01	-0.31092883	2006-04-25	2006-10-23
S&P 500 Information Technology	0.06	1.826706875	2012-04-05	2012-10-04
S&P 500 Telecommunications Services	0	-0.738455971	2011-08-02	2011-10-31
S&P 500 Utilities	0.01	-0.31092883	2013-03-20	2013-09-18
S&P 500 Consumer Staples	0.03	0.544125452	2007-10-17	2008-04-18
S&P 500 Consumer Discretionary	0.02	0.116598311	2014-04-21	2014-10-17
S&P 500 Real Estate	0.01	-0.31092883	2015-04-21	2015-10-19
Mean	0.0172727			
Std Dev	0.0233903			

Table B.6: 6 Months - Most Average Performance

B.1.3 12 Months

FUND	VALUE	NORMALIZED	FROM	TO
S&P 500 Materials	91.33	-0.606334425	2009-03-05	2010-03-05
S&P 500 Energy	194.53	1.398941145	2006-09-20	2007-09-21
S&P 500 Financials	123.3	0.01487352	2009-03-06	2010-03-08
S&P 500 Health Care	207.73	1.655429881	2014-04-11	2015-04-14
S&P 500 Industrials	127.63	0.099009598	2012-12-28	2013-12-30
S&P 500 Information Technology	167.46	0.872944927	2009-03-09	2010-03-09
S&P 500 Telecommunications Services	56.88	-1.275731163	2006-05-30	2007-05-31
S&P 500 Utilities	61.07	-1.194315421	2006-05-17	2007-05-18
S&P 500 Consumer Staples	94.14	-0.551733414	2014-02-05	2015-02-05
S&P 500 Consumer Discretionary	159.62	0.720606163	2012-11-15	2013-11-15
S&P 500 Real Estate	64.19	-1.13369081	2006-02-07	2007-02-08
Mean	122.5345455			
Std Dev	51.46424838			

Table B.7: 12 Months - Best Performance

Fund	Value	Normalized	From	To
S&P 500 Materials	-149.53	0.164473779	2008-03-05	2009-03-05
S&P 500 Energy	-292.7	-1.780741156	2008-04-22	2009-04-22
S&P 500 Financials	-292.78	-1.781828096	2007-10-10	2008-10-09
S&P 500 Health Care	-150.45	0.15197397	2007-11-30	2008-12-01
S&P 500 Industrials	-198.43	-0.499918275	2008-03-05	2009-03-05
S&P 500 Information Technology	-200.3	-0.525325496	2007-10-26	2008-10-27
S&P 500 Telecommunications Services	-85.8	1.030357329	2007-10-11	2008-10-10
S&P 500 Utilities	-81.95	1.082666315	2007-12-05	2008-12-04
S&P 500 Consumer Staples	-83.37	1.063373131	2008-03-05	2009-03-05
S&P 500 Consumer Discretionary	-138.91	0.30876506	2007-10-26	2008-10-27
S&P 500 Real Estate	-103.77	0.786203439	2008-04-02	2009-04-01
Mean	-161.6354545			
Std Dev	73.60112108			

Table B.8: 12 Months - Worst Performance

Fund	Value	Normalized	From	To
S&P 500 Materials	0.01	-0.286299167	2007-07-30	2008-07-29
S&P 500 Energy	-0.02	-1.231086419	2009-09-10	2010-09-10
S&P 500 Financials	0.03	0.343559001	2009-09-09	2010-09-09
S&P 500 Health Care	0.02	0.028629917	2010-01-21	2011-01-20
S&P 500 Industrials	-0.03	-1.546015503	2010-12-06	2011-12-05
S&P 500 Information Technology	0.05	0.973417168	2015-01-07	2016-01-07
S&P 500 Telecommunications Services	-0.02	-1.231086419	2009-05-11	2010-05-11
S&P 500 Utilities	0.01	-0.286299167	2007-07-27	2008-07-28
S&P 500 Consumer Staples	0.06	1.288346252	2007-07-09	2008-07-08
S&P 500 Consumer Discretionary	0.07	1.603275336	2010-11-26	2011-11-25
S&P 500 Real Estate	0.03	0.343559001	2012-12-03	2013-12-03
Mean	0.019090909			
Std Dev	0.03175318			

Table B.9: 12 Months - Most Average Performance

B.1.4 2 Years

Fund	Value	Normalized	From	To
S&P 500 Materials	134.17	-0.459404914	2009-03-05	2011-03-04
S&P 500 Energy	274.97	1.098943562	2009-03-05	2011-03-04
S&P 500 Financials	141.8	-0.374957479	2009-03-09	2011-03-08
S&P 500 Health Care	353.88	1.972304913	2012-12-06	2014-12-08
S&P 500 Industrials	189.12	0.148771569	2012-06-05	2014-06-09
S&P 500 Information Technology	252.53	0.850581774	2013-02-28	2015-03-02
S&P 500 Telecommunications Services	46.63	-1.428281516	2010-07-02	2012-07-02
S&P 500 Utilities	72.9	-1.137529993	2012-12-27	2014-12-29
S&P 500 Consumer Staples	149.67	-0.287853768	2012-12-26	2014-12-26
S&P 500 Consumer Discretionary	231.66	0.619596456	2011-11-25	2013-11-27
S&P 500 Real Estate	85.13	-1.002170605	2009-03-05	2011-03-04
Mean	175.6781818			
Std Dev	90.3520632			

Table B.10: 2 Years - Best Performance

Fund	Value	Normalized	From	To
S&P 500 Materials	-121.99	0.509939118	2007-02-26	2009-02-25
S&P 500 Energy	-284.26	-1.188117049	2008-07-01	2010-07-01
S&P 500 Financials	-410.68	-2.511024923	2007-02-22	2009-02-23
S&P 500 Health Care	-145.02	0.268944273	2007-04-25	2009-04-24
S&P 500 Industrials	-188.15	-0.182384762	2007-03-08	2009-03-09
S&P 500 Information Technology	-166.06	0.048773559	2006-11-20	2008-11-20
S&P 500 Telecommunications Services	-78.29	0.967232854	2007-05-31	2009-06-01
S&P 500 Utilities	-86.71	0.879122711	2007-04-25	2009-04-24
S&P 500 Consumer Staples	-68.07	1.074178895	2007-03-08	2009-03-09
S&P 500 Consumer Discretionary	-178.52	-0.08161271	2007-02-22	2009-02-23
S&P 500 Real Estate	-150.18	0.214948033	2007-02-20	2009-02-19
Mean	-170.7209091			
Std Dev	95.56220996			

Table B.11: 2 Years - Worst Performance

Fund	Value	Normalized	From	To
S&P 500 Materials	-0.04	-0.445324748	2010-12-02	2012-12-04
S&P 500 Energy	0.12	0.066944243	2013-02-28	2015-03-02
S&P 500 Financials	0	-0.3172575	2008-10-29	2010-10-29
S&P 500 Health Care	-0.04	-0.445324748	2008-04-10	2010-04-12
S&P 500 Industrials	0.37	0.867364541	2006-06-28	2008-06-30
S&P 500 Information Technology	0.08	-0.061123005	2008-07-09	2010-07-09
S&P 500 Telecommunications Services	-0.06	-0.509358371	2013-03-21	2015-03-23
S&P 500 Utilities	0.94	2.69232282	2006-09-21	2008-09-23
S&P 500 Consumer Staples	0.03	-0.221207064	2006-10-06	2008-10-08
S&P 500 Consumer Discretionary	-0.36	-1.469862729	2008-08-15	2010-08-17
S&P 500 Real Estate	0.05	-0.15717344	2006-06-07	2008-06-09
Mean	0.099090909			
Std Dev	0.312335907			

Table B.12: 2 Years - Most Average Performance

B.1.5 3 Years

FUND	VALUE	NORMALIZED	FROM	TO
S&P 500 Materials	127.41	-0.698238611	2009-03-03	2012-03-01
S&P 500 Energy	255.15	0.427096024	2009-03-03	2012-03-01
S&P 500 Financials	170.4	-0.319515141	2011-11-23	2014-11-26
S&P 500 Health Care	454.49	2.183195962	2012-07-12	2015-07-16
S&P 500 Industrials	224.96	0.161135067	2011-11-23	2014-11-26
S&P 500 Information Technology	317.33	0.974875166	2011-11-25	2014-11-28
S&P 500 Telecommunications Services	64.55	-1.252008261	2010-05-06	2013-05-08
S&P 500 Utilities	75.1	-1.159067284	2012-01-24	2015-01-27
S&P 500 Consumer Staples	194.41	-0.107997335	2011-11-25	2014-11-28
S&P 500 Consumer Discretionary	294.41	0.772959792	2012-07-25	2015-07-29
S&P 500 Real Estate	95.15	-0.98243538	2009-03-30	2012-03-28
Mean	206.6690909			
Std Dev	113.5129019			

Table B.13: 3 Years - Best Performance

Fund	Value	Normalized	From	To
S&P 500 Materials	-86.21	0.624771599	2006-03-03	2009-03-05
S&P 500 Energy	-178.06	-0.540325551	2007-07-20	2010-07-21
S&P 500 Financials	-352.46	-2.752551387	2006-03-06	2009-03-06
S&P 500 Health Care	-121.83	0.172939694	2006-03-01	2009-03-03
S&P 500 Industrials	-164.93	-0.373774375	2006-03-03	2009-03-05
S&P 500 Information Technology	-138.98	-0.044604303	2006-03-07	2009-03-09
S&P 500 Telecommunications Services	-79.46	0.710393872	2007-05-31	2010-06-01
S&P 500 Utilities	-72.97	0.792718102	2007-05-21	2010-05-20
S&P 500 Consumer Staples	-42.94	1.17364208	2006-03-07	2009-03-09
S&P 500 Consumer Discretionary	-135.95	-0.006169416	2006-03-07	2009-03-09
S&P 500 Real Estate	-116.31	0.242959686	2007-02-07	2010-02-08
Mean	-135.4636364			
Std Dev	78.83462763			

Table B.14: 3 Years - Worst Performance

Fund	Value	Normalized	From	To
S&P 500 Materials	-0.14	-0.480029455	2008-03-20	2011-03-21
S&P 500 Energy	0.17	0.994655627	2006-11-03	2009-11-05
S&P 500 Financials	0.15	0.899514654	2008-12-18	2011-12-19
S&P 500 Health Care	-0.58	-2.573130862	2008-09-12	2011-09-13
S&P 500 Industrials	-0.18	-0.670311401	2008-03-07	2011-03-08
S&P 500 Information Technology	0.04	0.376239302	2007-12-17	2010-12-16
S&P 500 Telecommunications Services	-0.03	0.043245897	2008-07-16	2011-07-15
S&P 500 Utilities	-0.13	-0.432458968	2008-09-18	2011-09-19
S&P 500 Consumer Staples	0.01	0.233527843	2007-10-10	2010-10-11
S&P 500 Consumer Discretionary	0.06	0.471380275	2007-11-02	2010-11-03
S&P 500 Real Estate	0.2	1.137367087	2008-07-18	2011-07-19
Mean	-0.039090909			
Std Dev	0.210214373			

Table B.15: 3 Years - Most Average Performance

B.1.6 5 Years

Fund	Value	Normalized	From	To
S&P 500 Materials	192.28	-0.547415632	2009-03-02	2014-03-04
S&P 500 Energy	381.38	0.739809392	2009-07-07	2014-07-09
S&P 500 Financials	219.25	-0.363827801	2009-03-06	2014-03-10
S&P 500 Health Care	560.09	1.956308515	2010-07-16	2015-07-20
S&P 500 Industrials	321.66	0.333288618	2009-03-05	2014-03-07
S&P 500 Information Technology	396.43	0.842256439	2009-03-09	2014-03-11
S&P 500 Telecommunications Services	65.97	-1.407222046	2008-11-20	2013-11-22
S&P 500 Utilities	98.93	-1.182859609	2010-01-27	2015-01-29
S&P 500 Consumer Staples	242.24	-0.207332278	2010-07-16	2015-07-20
S&P 500 Consumer Discretionary	405.31	0.902703601	2009-03-05	2014-03-07
S&P 500 Real Estate	116.14	-1.065709198	2010-01-22	2015-01-26
Mean	272.6981818			
Std Dev	146.9051614			

Table B.16: 5 Years - Best Performance

Fund	Value	Normalized	From	To
S&P 500 Materials	-50.01	0.179269678	2007-07-13	2012-07-12
S&P 500 Energy	-147.76	-0.8458844	2011-02-07	2016-02-10
S&P 500 Financials	-330.43	-2.761637807	2006-12-18	2011-12-19
S&P 500 Health Care	-26.59	0.424887156	2006-10-02	2011-10-03
S&P 500 Industrials	-73.9	-0.07127693	2007-07-17	2012-07-16
S&P 500 Information Technology	14.69	0.857811559	2007-11-06	2012-11-07
S&P 500 Telecommunications Services	-42.45	0.258555252	2007-05-31	2012-05-30
S&P 500 Utilities	-44.96	0.232231602	2007-12-10	2012-12-11
S&P 500 Consumer Staples	39.77	1.120838304	2006-08-09	2011-08-10
S&P 500 Consumer Discretionary	-12.66	0.570978167	2006-11-22	2011-11-23
S&P 500 Real Estate	-63.84	0.034227418	2006-11-24	2011-11-25
Mean	-67.10363636			
Std Dev	95.35152039			

Table B.17: 5 Years - Worst Performance

Fund	Value	Normalized	From	To
S&P 500 Materials	0.05	-0.417775727	2006-12-06	2011-12-07
S&P 500 Energy	0.43	-0.385514457	2007-08-08	2012-08-07
S&P 500 Financials	0.29	-0.397400188	2008-07-03	2013-07-08
S&P 500 Health Care	0.02	-0.420322669	2006-03-13	2011-03-14
S&P 500 Industrials	-0.37	-0.45343292	2007-02-28	2012-02-28
S&P 500 Information Technology	14.69	0.825132144	2007-11-06	2012-11-07
S&P 500 Telecommunications Services	0.04	-0.418624708	2006-04-18	2011-04-18
S&P 500 Utilities	0.05	-0.417775727	2008-03-12	2013-03-14
S&P 500 Consumer Staples	39.77	2.954375955	2006-08-09	2011-08-10
S&P 500 Consumer Discretionary	-0.31	-0.448339035	2006-09-29	2011-09-30
S&P 500 Real Estate	0.02	-0.420322669	2007-11-23	2012-11-26
Mean	4.970909091			
Std Dev	11.77882959			

Table B.18: 5 Years - Most Average Performance

B.1.7 10 Years

Fund	Value	Normalized	From	To
S&P 500 Materials	70.17	-0.492949553	2006-02-13	2016-02-18
S&P 500 Energy	51.18	-0.612877148	2006-02-15	2016-02-22
S&P 500 Financials	-143.45	-1.842024485	2006-01-31	2016-02-04
S&P 500 Health Care	396.61	1.568617813	2006-02-10	2016-02-17
S&P 500 Industrials	157.16	0.056418617	2006-02-15	2016-02-22
S&P 500 Information Technology	339.15	1.205740529	2006-02-15	2016-02-22
S&P 500 Telecommunications Services	38.92	-0.690302758	2006-01-31	2016-02-04
S&P 500 Utilities	76.06	-0.455752421	2006-02-03	2016-02-09
S&P 500 Consumer Staples	282.98	0.851009984	2006-02-10	2016-02-17
S&P 500 Consumer Discretionary	321.48	1.094149132	2006-02-15	2016-02-22
S&P 500 Real Estate	40.23	-0.682029711	2006-01-31	2016-02-04
Mean	148.2263636			
Std Dev	158.345541			

Table B.19: 10 Years - Best Performance

Fund	Value	Normalized	From	To
S&P 500 Materials	54.13	-0.460703139	2006-02-07	2016-02-11
S&P 500 Energy	-7.57	-0.86529305	2006-02-06	2016-02-10
S&P 500 Financials	-157.16	-1.846210468	2006-02-07	2016-02-11
S&P 500 Health Care	362.01	1.558180842	2006-02-02	2016-02-08
S&P 500 Industrials	136.95	0.082378488	2006-02-07	2016-02-11
S&P 500 Information Technology	301.17	1.159230273	2006-02-02	2016-02-08
S&P 500 Telecommunications Services	30.03	-0.618735827	2006-02-16	2016-02-23
S&P 500 Utilities	70.47	-0.353555665	2006-02-08	2016-02-12
S&P 500 Consumer Staples	267.41	0.937853362	2006-02-02	2016-02-08
S&P 500 Consumer Discretionary	282.17	1.034640187	2006-02-06	2016-02-10
S&P 500 Real Estate	28.65	-0.627785002	2006-02-07	2016-02-11
Mean	124.3872727			
Std Dev	152.5000955			

Table B.20: 10 Years - Worst Performance

Fund	Value	Normalized	From	To
S&P 500 Materials	54.13	-0.483051405	2006-02-07	2016-02-11
S&P 500 Energy	0.71	-0.840177127	2006-02-03	2016-02-09
S&P 500 Financials	-143.45	-1.803921869	2006-01-31	2016-02-04
S&P 500 Health Care	362.01	1.575201447	2006-02-02	2016-02-08
S&P 500 Industrials	136.95	0.070620484	2006-02-07	2016-02-11
S&P 500 Information Technology	301.17	1.168471216	2006-02-02	2016-02-08
S&P 500 Telecommunications Services	30.03	-0.64416578	2006-02-16	2016-02-23
S&P 500 Utilities	70.47	-0.373814522	2006-02-08	2016-02-12
S&P 500 Consumer Staples	267.41	0.942777386	2006-02-02	2016-02-08
S&P 500 Consumer Discretionary	282.17	1.041451584	2006-02-06	2016-02-10
S&P 500 Real Estate	28.65	-0.653391416	2006-02-07	2016-02-11
Mean	126.3863636			
Std Dev	149.5831767			

Table B.21: 10 Years - Most Average Performance

B.2 Factiva search factors

CRITERION	VALUE
Text	wc>50
Date	22/01/2008 to 22/04/2008
Source	Major News and Business Sources: U.S.
Author	All Authors
Company	All Companies
Subject	All Subjects
Industry (see B.25 on the following page)	Energy
Region	All Regions
Language	English
Results Found	4567
Results Found (excluding dj newswires)	1817

Table B.22: S&P 500 Energy: Best 3 Months (22/01/2008 - 22/04/2008)

CRITERION	VALUE
Text	wc>50
Date	14/07/2008 to 10/10/2008
Source	Major News and Business Sources: U.S.
Author	All Authors
Company	All Companies
Subject	All Subjects
Industry (see B.25 on the next page)	Energy
Region	All Regions
Language	English
Results Found	3700
Results Found (excluding dj newswires)	2115

Table B.23: S&P 500 Energy: Worst 3 Months (14/07/2008 - 10/10/2008)

CRITERION	VALUE
Text	wc>50
Date	30/01/2012 to 30/04/2012
Source	Major News and Business Sources: U.S.
Author	All Authors
Company	All Companies
Subject	All Subjects
Industry (see B.25)	Energy
Region	All Regions
Language	English
Results Found	12,760
Results Found (excluding dj newswires)	3549

Table B.24: S&P 500 Energy: Most Average 3 Months (30/01/2012 - 30/04/2012)

The following mapping is based on a subjective interpretation of the fund description and industry choices available in the Factiva search selection.

FUND	INDUSTRY
S&P 500 Materials	Basic Materials/Resources
S&P 500 Energy	Energy
S&P 500 Financials	Financial Services
S&P 500 Health Care	Health Care/Life Sciences
S&P 500 Industrials	Industrial Goods, Transportation/Logistics
S&P 500 Information Technology	Technology
S&P 500 Telecommunication Services	Telecommunication Services
S&P 500 Utilities	Utilities
S&P 500 Consumer Staples	Agriculture, Business/Consumer Services, Consumer Goods
S&P 500 Consumer Discretionary	Automotive, Leisure/Arts/Hospitality, Media/Entertainment, Retail/Wholesale
S&P 500 Real Estate	Real Estate/Construction

Table B.25: S&P fund mapping to Factiva Industry criteria