Sentiment Analysis in Financial News

A thesis presented
by

Pablo Daniel Azar

to
Applied Mathematics
in partial fulfillment of the honors requirements
for the degree of
Bachelor of Arts

Harvard College
Cambridge, Massachusetts
April 1 2009

Abstract

This thesis studies the relation between the numerical information found in financial markets and the verbal information reported in financial news. I explore whether we can use market data to teach a computer to distinguish between positive news and negative news. If we have an algorithm that successfully determines polarity in financial texts, can we apply it to outside domains such as movie reviews? Can we use knowledge about polarity in texts to predict market returns?

These questions have been approached by two communities: one in finance, one in machine learning. These communities use different methods and are unaware of each other's work. This thesis compares and combines the existing methods to sentiment analysis in financial news, evaluating them using the above questions. I find that methods from machine learning are very effective at distinguishing between positive and negative news. However, algorithms trained on financial news are not useful for predicting sentiment in movie reviews. I also find that the methods from machine learning outperform the methods proposed by the finance community when predicting future stock returns. These predictions suggest strategies that generate positive returns when trading with public information, but these returns vanish after accounting for trading costs.

Contents

Acknowledgments

There are many people who have contributed to the thesis you are reading. I am indebted to Professor Efraim Benmelech for his advice during this project. He introduced me to research in Economics and encouraged my interest in the intersection of language and finance. I worked for him writing information retrieval programs, and he asked if I could look at the relation between news and markets. I took this idea and developed it under his guidance, resulting in the current work.

Both this thesis and my original research which lead to it would not have been possible without the support of the Institute for Quantitative Social Science at Harvard University.

This thesis is heavily influenced by my education in computer science. Professor Avrom Pfeffer first introduced me to machine learning and many of the methods that I apply in this thesis. Professor Stuart Shieber offered very good advice that shaped the way I present most of the results in this thesis.

Laura Serban and Erkko Etula forced me through the grueling process of writing and re-writing. They put time, energy and commitment not only on this work, but also on 10 other students' thesis. There are dozens of pages of obfuscated material you are not reading because Laura and Erkko read it and corrected it before it reached your hands. I am also grateful to Spencer Strub from the Harvard Writing Center for helping me edit this thesis.

HBS research librarians Lydia Petersen and Barbara Esty gave me invaluable advice on data sources for news and market data.

I worked with Stephen Chang, Dimitrios Antos and Shrenik Shah on a project for a Computational Finance class taught by Christopher Thorpe. That project, on a disjoint but related topic, helped me form ideas and develop the experimental techniques that I use throughout this thesis.

Writing this has required many months of work. The work was made more enjoyable with Stephanie Liem's support, as well as the support of my family: Hector, Sara, Deborah, Matt and Jose.

Chapter 1

Introduction and summary

Researchers have been interested in automatically detecting sentiment in texts for at least a decade. If we label some texts as positive and some texts as negative and feed these examples into a computer, is there an algorithm that learns the characteristics of each emotion? The seminal work in sentiment analysis [Pang, Lee, and Vaithyanathan (2002)] used movie reviews to train an algorithm that detects sentiment in text. Movie reviews are a good source for this kind of work because they clearly express an opinion, and because they are accompanied by a numeric rating that makes it easier to train learning algorithms on this data.

This thesis focuses on another set of automatically labelled texts: financial news. Every day, investors receive news about different firms. These news convey information, which lead different investors to make decisions. These decisions impact the market, which aggregates the information that investors receive and reflects it via the price of the firms.

Through this process, information is converted from a verbal form to a numeric form. This is useful, because it allows information to be easily summarized and compared. There may be disagreement about the true meaning of a piece of news, but there rarely is disagreement about market returns. Positive returns are good, negative returns are bad. This allows us to label news in an analogous way to labeling movie reviews: news that have a significantly positive impact on returns are good, news that have a significantly negative impact on returns are bad.

There is one fundamental way in which classifying financial news is a harder problem than classifying movie reviews: the data is much noisier. In the case of movie reviews, the text writer is the same person who assigns a positive or negative classification. In financial news the text is written by a reporter who has no control over how the market will react. The classification is assigned by aggregating the trades of everyone who reacts to the news, as well as the trades of other agents whose decisions are independent of what is in the newspapers. Thus, the return on a firm on a given day may not be well connected to the news that was published about the firm on that day. This is unlike the strong connection between classification and text in a movie review.

The core question of this thesis is whether we can still use this automatically labelled data to train algorithms that detect sentiment in text. The results I present show that this is feasible. In fact, some algorithms have a performance comparable to humans on this task, as I will show in chapter 4.

Given an algorithm for classifying text according to sentiment, we can ask whether this algorithm helps us predict future returns. I present results that show that automatic sentiment detection can be used to generate returns if one could trade before news are announced. However, in the more realistic situation where on trades after news are announced, one cannot obtain positive returns after accounting for trading costs.

Another question we can ask is whether we can use what we learn from financial news to classify other texts as positive or negative. To answer this question, I present results where the learning algorithms are trained on financial news and tested on movie reviews. The results show that applying knowledge from the financial domain to the movie review domain gives results that are no better than random guessing. Similarly, learning from movie reviews is not useful for classifying financial news.

1.1 Previous Work

There have been two independent lines of work in sentiment analysis. In the Natural Language Processing (NLP) community[1], words that are relevant to sentiment are automatically learned from the data. Each news story is summarized as a binary vector $(w_1, ..., w_n)$. The attribute w_i is equal to 1 if word i is present in the story and equal to zero if word i is absent.

Non-linear algorithms such as decision trees and support vector machines are frequently used to map the words in a text to its classification. There has been work applying these techniques to financial news [Koppel and Shtrimberg (2006)], [Genereux, Poibeau and Koppel (2008)]. This thesis expands on this work by exploring the avenues for research left open by [Genereux, Poibeau and Koppel (2008)]. In particular, I take into consideration risk-adjusted returns and trading costs when analyzing the trading strategies recommended by each algorithm.

In the finance community, two seminal papers [Tetlock (2007)], [Tetlock, Saar-Tsechansky and Macskassy (2007)] have recently been published exploring sentiment in financial news. The first of these papers explores the relation between the Dow Jones Industrial Index and and a pessimism index. This pessimism index is based on the number of negative words in the Wall Street Journal's *Abreast of the Market* column. The second paper looks at individual stock returns, and their relation to the number of negative words in stories from the Wall Street Journal and from the Dow Jones News Service.

The negative words used in these papers are not learned from the data. Instead, the authors use the General Inquirer Dictionary [Stone (1966)], a commonly used dictionary in sentiment analysis. The General Inquirer classifies words according to multiple categories, including positive or negative. This dictionary includes 1915 positive words and 2291 negative words. Each news story is summarized with one number **Neg**, the number of negative words that appear in the story. This is in contrast to the learning approach, where each news story is characterized by a binary vector.

The main original contribution of this thesis is to compare, contrast and combine these two approaches. Each community is unaware of the other's work. The NLP approach uses sophisticated

[1] Also referred to as machine learning in this thesis

classification techniques, but lacks an analysis of trading costs and risks. The finance approach builds on what is known about stock returns, but relies on using word counting to summarize a piece of news.

I evaluate both of these approaches on three questions that motivate this thesis. I find that machine learning techniques are better than the word counting approach for predicting whether a piece of news will be positive or negative. On the other hand, classification techniques that are trained on financial news cannot be applied successfully outside the financial domains.

The machine learning approach also outperforms the word counting approach at predicting future returns. Machine learning generate positive returns when trading after news are announced, but these profits may vanish if we account for trading costs. An investor with insider information that can trade before news are announced can use the machine learning methods to predict how the market will react. This investor can generate annualized returns exceeding 400%.

1.2 Structure of this thesis

This work is structured as follows: chapter 2 gives a primer on Efficient Markets and the Fama French Three Factor Model. It also explains the methodology used in the finance community. Chapter 3 introduces the machine learning methods used in the natural language processing community, as well as different measures of performance used. Chapter 4 compares the methods from chapters 2 and 3, and answers the three main questions motivating this thesis:

1. Can we learn to predict sentiment using financial news?
2. Can we develop trading strategies using our algorithms to predict sentiment? How risky and costly are these strategies?
3. Can we use our algorithms trained on financial news to predict sentiment in other domains, such as movie reviews?

Chapter 5 reviews related work, suggests future avenues of research, and provides a conclusion.

Chapter 2

The General Inquirer Approach

This chapter gives a primer on the Efficient Markets Hypothesis and the Fama French Three Factor Model, both of which are used throughout this thesis. In addition, I describe the approach to sentiment analysis that represents each piece of news by two numbers: the number of positive words and the number of negative words mentioned in the piece. These words are chosen from the General Inquirer dictionary [Stone (1966)], which lists 1915 positive words and 2291 negative words. All words that are not in these lists are ignored.

The other approach, described in chapters 3 and 4, represents a piece of news as a binary feature vector $(w_1, ..., w_n)$ where w_i is equal to 1 if word i is in the given piece of news. The features are learned automatically from the data, as opposed to from a list defined by a human expert.

The results in this chapter confirm what was shown in 2007 by Tetlock, Maytal Saar-Tsechansky, and Sofus Macskas [Tetlock, Saar-Tsechansky and Macskassy (2007)]. In particular, the proportion of negative and positive words in a text are highly correlated with stock returns.

However, most of the variance in abnormal stock returns is still unexplained by these measures of sentiment. The word counting method is limited because it captures the complexity of news using only two attributes: the number of positive words and the number of negative words in the text. I suggest at the end of this chapter, and show in chapter 4, that more sophisticated approaches give us better predictions of future abnormal returns.

2.1 The Efficient Markets Hypothesis and Risk Adjustment

2.1.1 The Efficient Markets Hypothesis

Assume you are an investor looking at ABC Inc.'s stock. ABC is currently trading at $100, but you have information that the firm is about to receive a lucrative government contract. You predict that when this information is revealed, ABC will be valued at $110. Thus, you place an order to buy the stock. If all sellers have access to this information and agree with your valuation, no one will want to sell you at any price lower than $110. Thus, you will have to move your bid up to that price, and the market will reflect this new piece of information. The Efficient Markets Hypothesis is a formalization of this fact: market prices immediately reflect all available information. There are three common versions of this hypothesis:*weak, semi-strong* and *strong*.

The weak form of the hypothesis states that no profit can be made by looking at transaction information: previous prices, bids and offers, and volume levels. This negates the possibility of developing profitable trading strategies based on the study of patterns in price and volume levels.

The semi-strong form of the hypothesis states that no profit can be made by looking at *publicly available* information. This includes price and volume levels, as well as balance sheet, production, and earnings information publicly announced by firms. People with inside knowledge that few others share can still develop profitable trading strategies. Obtaining this knowledge and using it for trading is either costly or illegal.

The strong form of the hypothesis states that the market immediately reflects all information, including *private* information. There are stylized stories about the strong version of the hypothesis. One story suggests that oil exploration workers, upon a new discovery, will call their brokers to trade the firms' stock before they call their managers. This should not happen in a world where there are legal obstacles to insider trading.

2.1.2 Abnormal Returns

The common thread throughout this thesis is labeling financial texts using market returns. We see a piece of news and assign it a positive or negative classification depending on how the market moves. But how do we know that the market is reacting to the information in the given piece of news. How do we know that a company is reacting to new information? Are we sure it is not being moved by the pressures of traders with interests unrelated to the news, or by general market trends?

In essence, we want a quantity that is independent of the risk in market movements and other information not related to the news event. This quantity is called the **abnormal return**. This is the amount of daily stock return that cannot be explained using traditional explanations.

But, what is a traditional explanation? What constitutes a normal return? The most simple model of stock returns is the Capital Asset Pricing Model (CAPM). This model states that the excess return in asset i can be decomposed in two components. One of these components captures all the risk from correlation with the market. The other component is independent of market movements or any other assets. It captures all the risk that is unique to the stock.

Mathematically, the CAPM can be expressed with the equation $R_{it} - R_f = \beta_i(R_{mt} - R_f) + \epsilon_{it}$, where R_{it} is the return of asset i during time period t, R_f is the riskless return and R_{mt} is the return of a market index during the same period.[1]

If we are allowed to sell short and borrow as much money as we want[2], we can create a synthetic asset with return ϵ_{it} by purchasing and shorting a suitable combination of asset i, the market index and treasuries.[3] If we assume the existence of arbitrage, the expected return of such a synthetic asset will have to equal zero, as I explain below.

---

[1]Note that we model excess return $R_i - R_f$. A manager that cannot obtain an average positive excess return is not worth her commission. It would be safer and more profitable to invest the money with the government.

[2]Recall that shorting a stock is the act of borrowing it from a broker and selling it, with the expectation that its price will go down. The position is closed by purchasing the stock and returning it to the broker. This requires paying interest to the broker and putting up collateral. I assume unlimited short selling.

[3]In particular, we can do this by purchasing one share of asset i, shorting β_i shares of the maket index and borrowing $(-1 - \beta_i)$ at the risk free rate. f we have a strategy to choose assets $1, ..., N$ so that $E[\epsilon_i] > 0$, then we can generate an arbitrage opportunity holding a portfolio. This requires no capital, since we can use the money we get from shorting and borrowing in order to purchase asset i.

Suppose we can choose assets $1, ..., N$ such that $E[\epsilon_i] > 0$ for all these assets. Suppose further, as the CAPM does, that ϵ_i is inherent to asset i, and hence independent of ϵ_j for $i \neq j$. If N is large enough, we can create an arbitrage opportunity by creating a portfolio that invests in these assets. The average return of this portfolio would be positive. On the other hand, the variance of this portfolio tends to zero as N becomes large, because we assumed independence between ϵ_i and ϵ_j. If we assume that arbitrage opportunities do not exist, then we must conclude that $E[\epsilon_i] = 0$.

The CAPM has been very successful, but it has its limitations. Fama and French [Fama and French (1992)] showed that investing in a portfolio of stocks with a high book to market ratio yielded significant abnormal returns beyond those explained by the CAPM. A portfolio of stocks with small market value yielded simliar results. If the CAPM were a true description of stock returns, buying any of these portfolios would provide investors with an arbitrage opportunity.

Thus, Fama and French developed an alternative to the CAPM. This is the three factor model, which uses the following three portfolios to capture the correlation of a security with the market:

1. The **Market** portfolio. The return of this portfolio on a given day is the average return of all securities in the market. This average may or may not be weighted by market value. In this thesis, I use an unweighted average computed over all stocks in the CRSP database.

2. The **Small Minus Big** (SMB) portfolio. This portfolio accounts for the fact that small firms seem to have larger excess returns than bigger firms. The return on this portfolio on any given day is obtained by buying a pre-determined list of small firms and shorting an analogous list of large firms.

3. The **High Minus Low** (HML) portfolio. This portfolio accounts for the fact that firms with a high book to market value (value stocks) have larger excess returns than firms with a small book to market value (growth stocks). The return on this portfolio on any given day is obtained by buying a pre-determined list of value firms and shorting an analogous list of growth firms.

In this model, asset i's excess return $R_i - R_f$ is a linear function of the market excess

return $R_{Market} - R_f$, the Small Minus Big (SMB) portfolio excess return $R_{SMB} - R_f$ and the High Minus Low (HML) portfolio excsess return $R_{HML} - R_f$.

$$R_i - R_f = \beta_{i1}(R_{Market} - R_f) + \beta_{i2}(R_{SMB} - R_f) + \beta_{i3}(R_{HML} - R_f) + \epsilon_i.$$

The Fama-French model assumes that all the undiversifiable risk of asset i is captured by the correlation of the asset with these three portfolios. The residual term ϵ_i is inherent to the asset, and uncorrelated with the movement of any other assets. It is assumed to have a normal distribution with mean zero and variance σ_i^2.

We can create a synthetic asset with return ϵ_i by purchasing or short selling an appropriate combination of asset i, the risk-free asset and the three market factors. If we have a strategy that chooses assets $i_1, ..., i_n$ so that $E[\epsilon_i] > 0$ and the correlation $Cor(\epsilon_i, \epsilon_j) = 0$ for $i \neq j$, then we can generate profitable strategies with very little risk. The variance of the return will tend to zero as the number of assets increases. This chapter, as well as chapter 4, explores what happens when we choose the assets based on sentiment analysis of financial news.

2.2 Data collection

I merge four sources of data: a corpus of news, a dataset of quarterly fundamentals, a dataset reporting earnings surprises, and a dataset of abnormal returns. These sources of data are described in detail in this section.

2.2.1 News Data

The first source is the Reuters Key Developments Corpus, which contains important news for individual stocks. The topics covered include, but are not limited to, earnings announcements, product unveilings, and lawsuit filings. The corpus ranges between the years 1998 and 2009. Only companies with more than 20 news stories are used. This raw dataset contains 3859 companies and 385620 pieces of news. The average number of news per company is close to 100.

Each piece of news contains four important elements: a stock ticker, a publication date, a headline and a body. If the piece of news appears after the market closes, then I move the publication date forward by one day. This assumption ensures that pieces of news are associated with market movements that happen after they are published.

For each stock, each day that a piece of news is published is considered an event date. Intuitively, this means that some new information about that stock is made available. I concatenate all the news for one event date and treat them as one big story. Suppose that Reuters releases two news stories for XYZ Inc. on April 1st. These two stories are "XYZ Inc. announces dividend." and "ABC Corp. files lawsuit against XYZ Inc." These two stories are concatenated and treated as one single story – "XYZ Inc. announces dividend. ABC Corp. files lawsuit against XYZ Inc." – because they happen on the same date.

I drop an event date for a stock if there are less than 20 words associated with the stock on that date. I also drop the event date if the number of positive and negative words adds up to a number less than 5. This is done in accordance with [Tetlock, Saar-Tsechansky and Macskassy (2007)]. [4]

Given a security i and an event date t, we know how many positive and negative words are mentioned for that security in the given date. We also know how many words are used in total for that security in the given event date. Using these numbers, we can compute the proportion of words that are positive and the proportion of words that are negative. Call these quantities **POS** and **NEG**.

For example, suppose that we get the piece of news "XYZ Inc. threatened with lawsuit." According to the General Inquirer dictionary, the only negative word in this piece of news is "threatened". There are 5 words in this piece of news. Hence, the variable **NEG** for this event is 0.2. Since

[4] I perform this pre-processing not only for the General Inquirer approach, but also for the learning approach described in the next chapters. For both approaches, I discard event dates that have less than 20 words associated with the event. I also discard the event dates whose positive and negative words add up to a number less than 5. I do this to ensure that each event has a sufficiently long text associated with it, and that the text has some notion of sentiment.

In this way, the General Inquirer Dictionary is an input to both approaches. The main difference is that the General Inquirer approach uses the dictionary blindly, without looking at the relationship of the words with the data. The learning approach uses the data to figure out which words are important. It is not necessary that these words be mentioned in the General Inquirer Dictionary.

there are no positive words the variable **POS** is zero.

Tetlock et al. suggest normalizing these variables to control for changes in editorial tone and style. I follow this approach in order to maintain my results comparable to theirs. I compute the mean μ_{pos} and standard deviation σ_{pos} of **POS** for the calendar year previous to the event. The normalized variable is $\mathbf{pos} = \frac{\mathbf{POS} - \mu_{pos}}{\sigma_{pos}}$. I do a similar calculation for **NEG**.

In the above example, assume that the piece of news "XYZ Inc. threatened with lawsuit" is published in 2005. Assume further that the mean value for the variable **NEG** in 2004 was 0.3 and that the standard deviation was 0.5. For this event, the value of the variable **neg** is equal to $\frac{0.2 - (0.3)}{0.5} = -0.2$. Note that this normalizing procedure can yield negative values for **neg** and **pos**.

2.2.2 Economic Data

The second source of data consists of fundamental variables for each company and each fiscal quarter. These are obtained from the Compustat tool in the Wharton Research Data Services website [WRDS]. For each company and each quarter, I retrieved the total assets and liabilities, the number of shares outstanding, the number of shares traded and the closing price for the quarter. The following variables were derived from this data:

1. Market Value: Number of shares outstanding · Closing Price
2. Book Value: Total Assets - Total Liabilities
3. Share Turnover: $\frac{\text{Shares Traded}}{\text{Shares Outstanding}}$

The third source of data is the Eventus Tool from the Wharton Research Data Services website [WRDS]. Given a security and an event date, this tool computes the abnormal return associated with the event. I specify the Fama French three-factor model as a benchmark. This model is estimated using a period before the event date. This estimation period starts a year before the event, and ends 31 trading days before the event.

This source provides many important variables. The cumulative abnormal returns $\mathbf{CAR_{+1}}$, $\mathbf{CAR_0}$, $\mathbf{CAR_{-1}}$, $\mathbf{CAR_{-2}}$, $\mathbf{CAR(-30,-3)}$ correspond to periods after, during and before the event. If the event happened on day 0, then $\mathbf{CAR_{+t}}$ represents the abnormal return t trading days after the event and $\mathbf{CAR_{-t}}$ represents the abnormal return $-t$ trading days before the event. The variable $\mathbf{CAR_{(-30,-3)}}$ represents the cumulative abnormal return between 30 and 3 days before the event.

The Eventus tool also provides us with the coefficient α of a security in the Fama-French three factor model. This reflects the risk-adjusted return that the security provided in the year previous to the event date.

The fourth source of data is the Institutional Brokers Estimate System database (I/B/E/S) [WRDS]. From this source I obtain earnings forecasts, actual earnings and earnings announcement dates for each quarter. Earnings surprises are calculated by subtracting actual earnings from earnings forecasts.

2.3 Experiments and results

2.3.1 Explaining returns

The experiments in this section explain abnormal return as a function of market sentiment. I use two different measures of abnormal return as my dependent variables. In the first set of experiments, I use the abnormal return on the day after a given piece of news is published $\mathbf{CAR}_1$. In the second set of experiments, I use the abnormal return on the day the news is published $\mathbf{CAR}_0$.

Note that returns on day zero are computed by comparing the closing price on day zero to the closing price on day -1. To obtain a return equal to $\mathbf{CAR_0}$, one would need to trade the night *before* news is revealed. Unless one has insider information, this is impossible. The best return one can hope for is $\mathbf{CAR_{(+1,+t)}}$. This return is obtained by trading the asset when the market closes the day the news is announced, and closing the position t days later. Nevertheless, the measure $\mathbf{CAR_0}$ is very important to analyze. It gives us a notion of the difference in security price *before* and *after* a piece of news is announced. It is also useful as an upper bound on the returns generated

by an investor that has instant access to news and can trade immediately.

If we believe the efficient markets hypothesis, we would expect the return on day zero to be highly correlated with news sentiment, but the return on day 1 to be insignificantly correlated. The experiments show that news sentiment is highly correlated with returns on the day the news is published. The effect of sentiment decreases on the day after the news is published, but it is still significant.

I add the variables $\log(\mathbf{share\_turnover})$, $\log(\frac{\mathbf{book\_value}}{\mathbf{market\_value}})$, $\log(\mathbf{market\_equity})$ and **unexpected_earnings** as regressors because they are known to explain abnormal returns beyond the Fama-French three factor model. Furthermore, I add the variables $\mathbf{CAR}_{-1}$, $\mathbf{CAR}_{-2}$, $\mathbf{CAR}_{(-30,-3)}$ and α. These variables control for an autocorrelation effect that may be causing the security to deviate from the Fama-French model. For example, the security may be experiencing momentum from a new trend that started last week. These are the same regressors as those used in [Tetlock, Saar-Tsechansky and Macskassy (2007)]. The only addition is **pos**, the proportion of positive words.

Table 2.1 shows the result of using $\mathbf{CAR}_1$ as the dependent variable. The instances are clustered by date to allow correlation among events that happen on the same date, and to allow different variances on different dates. One immediate thing to note is the significantly positive correlation between abnormal returns and positive words, as well as the significantly negative correlation between abnormal returns and negative words. This confirms one of the expectations motivating this thesis: sentiment can be used to explain returns. This explanatory power still holds when we use other economic variables.

This shows that the variables **pos** and **neg** are suitable explanations of returns on day 1. However, do they provide any additional explanatory power beyond fundamental economic variables? Can this method explain something about stock return variance that was unexplained before? If we use R^2 as a measure of explanatory power, we see that the answer is no. Table 2.2 shows that the R^2 statistic when sentiment is not taken into account is the same as the R^2 statistic when sentiment variables are included as regressors. Both equal 0.005. Hence, adding positive and negative sentiment as regressors does not give any extra explanation for the variance in returns the day *after* the news

is published.

Table 2.1: Dependent Variable : $\mathbf{CAR_{+1}}$

Variable	**Coefficient**	**(Std. Err.)**	**(t-statistic)**
pos	0.033***	(0.010)	(3.467)
neg	-0.034***	(0.010)	(-3.481)
earningSurprise	0.014	(0.034)	(0.418)
log(**marketValue**)	-0.017***	(0.005)	(-3.374)
$\log(\frac{\mathbf{bookValue}}{\mathbf{marketValue}})$	-0.002	(0.006)	(-0.280)
log(**turnover**)	-0.043***	(0.013)	(-3.387)
α	-89.360***	(7.502)	(-11.912)
$\mathbf{FFCAR_0}$	0.004	(0.005)	(0.864)
$\mathbf{FFCAR_{-1}}$	-0.035***	(0.006)	(-5.853)
$\mathbf{FFCAR_{-2}}$	-0.019***	(0.006)	(-3.517)
$\mathbf{FFCAR_{(-30,-3)}}$	-0.002	(0.001)	(-1.689)
Constant	0.706***	(0.170)	(4.156)
R-squared	0.005		
N	144451		

$*p < 0.05, **p < 0.01, ***p < 0.001$

Table 2.2: Dependent Variable : $\mathbf{CAR_{+1}}$ without using sentiment

Variable	**Coefficient**	**(Std. Err.)**	**(t-statistic)**
earningSurprise	0.015	(0.034)	(0.437)
log(**marketValue**)	-0.017***	(0.005)	(-3.448)
$\log(\frac{\mathbf{bookValue}}{\mathbf{marketValue}})$	-0.002	(0.006)	(-0.303)
log(**turnover**)	-0.042**	(0.013)	(-3.275)
α	-88.846***	(7.507)	(-11.835)
$\mathbf{FFCAR_0}$	0.005	(0.005)	(0.978)
$\mathbf{FFCAR_{-1}}$	-0.035***	(0.006)	(-5.829)
$\mathbf{FFCAR_{-2}}$	-0.019***	(0.006)	(-3.505)
$\mathbf{FFCAR_{(-30,-3)}}$	-0.002	(0.001)	(-1.655)
Constant	0.687***	(0.170)	(4.042)
R-squared	0.005		
N	144451		

$*p < 0.05, **p < 0.01, ***p < 0.001$

The situation is different if we look at the returns on day zero. Tables 2.3 and 2.4 show that positive and negative sentiment do help explain the variance in these returns. In particular, the R^2 in the model with sentiment variables is twice as much as the R^2 in the model without sentiment variables.

Note that the coefficients for **pos** and **neg** are at least 5 times larger than in the previous regression. In fact, these are the variables that are most significantly correlated with returns,

Table 2.3: Dependent Variable : $\mathbf{CAR_0}$

Variable	**Coefficient**	**(Std. Err.)**	**(t-statistic)**
pos	0.262***	(0.016)	(15.944)
neg	-0.298***	(0.016)	(-18.186)
earningSurprise	0.145***	(0.038)	(3.802)
log(**marketValue**)	-0.116***	(0.009)	(-13.119)
$\log(\frac{\mathbf{bookValue}}{\mathbf{marketValue}})$	-0.039**	(0.012)	(-3.288)
log(**turnover**)	-0.103***	(0.020)	(-5.131)
α	-112.589***	(13.392)	(-8.407)
$\mathbf{FFCAR_{-1}}$	-0.007	(0.011)	(-0.590)
$\mathbf{FFCAR_{-2}}$	-0.017	(0.009)	(-1.940)
$\mathbf{FFCAR_{(-30,-3)}}$	-0.003	(0.002)	(-1.554)
Constant	2.297***	(0.267)	(8.593)
R-squared	0.008		
N	144482		

$*p < 0.05, **p < 0.01, ***p < 0.001$

Table 2.4: Dependent Variable : $\mathbf{CAR_0}$ without using sentiment

Variable	**Coefficient**	**(Std. Err.)**	**(t-statistic)**
earningSurprise	0.152***	(0.039)	(3.936)
log(**marketValue**)	-0.119***	(0.009)	(-13.496)
$\log(\frac{\mathbf{bookValue}}{\mathbf{marketValue}})$	-0.040***	(0.012)	(-3.408)
log(**turnover**)	-0.091***	(0.020)	(-4.544)
α	-108.514***	(13.349)	(-8.129)
$\mathbf{FFCAR_{-1}}$	-0.005	(0.011)	(-0.484)
$\mathbf{FFCAR_{-2}}$	-0.017	(0.009)	(-1.875)
$\mathbf{FFCAR_{(-30,-3)}}$	-0.002	(0.002)	(-1.383)
Constant	2.146***	(0.268)	(8.015)
R-squared	0.004		
N	144482		

$*p < 0.05, **p < 0.01, ***p < 0.001$

according to the t-statistics. These shows that the variables are good at explaining returns. However, the variables are not very good at *predicting* future returns. If we look at R^2, we see that more than 90% of the variance in abnormal returns is unexplained by these variables.

2.3.2 Creating Portfolios

To explore further how information is absorbed, I create four portfolios: two positive, two negative. To create the portfolios, I calculate the quartiles for the variables **pos**, **neg** on the calendar year before the event. Thus, if the event happens in 2005, I use the quartiles from 2004. The first positive portfolio contains all securities where the variable **pos** is in the *highest* quartile. The second positive portfolio contains all securities where the variable **neg** is in the *lowest* quartile. The negative portfolios are constructed analogously.

Figure 2.1 shows the compounded abnormal returns for each portfolio. For each portfolio, we know the average abnormal risk-adjusted return $\overline{CAR}_t$ where t ranges from 30 days prior to the event to 30 days after the event. Thus, the x axis ranges from $t = -30$ to $t = 30$. The associated y value is the cumulative log return: $\sum_{i=-30}^{t} \log(1+\overline{CAR}_i)$. This allows us to explore how information affects returns beyond the first and second day after the news is published.

From the graphs we can see four effects:

1. The effect of the news announcement is concentrated on the event day. A trader who does not have access to inside information or cannot trade immediately may not be able to take advantage of this strategy. I explore this more in chapter 4.

2. There is a reversal to a (generally negative) trend. There may be a spike at the event date, but after this information is absorbed the abnormal return goes back to its normal trend. Note that this should provide an arbitrage opportunity in the medium term: when you see a piece of news published as a Reuters Key Development, short the stock. If the market absorbed all information, the trend should be flat. I do not explore this inefficiency in this thesis.

3. There is a positive spike around the event date for the high positive, low negative and low positive portfolios. The spike begins before the event date, which suggests that the market has

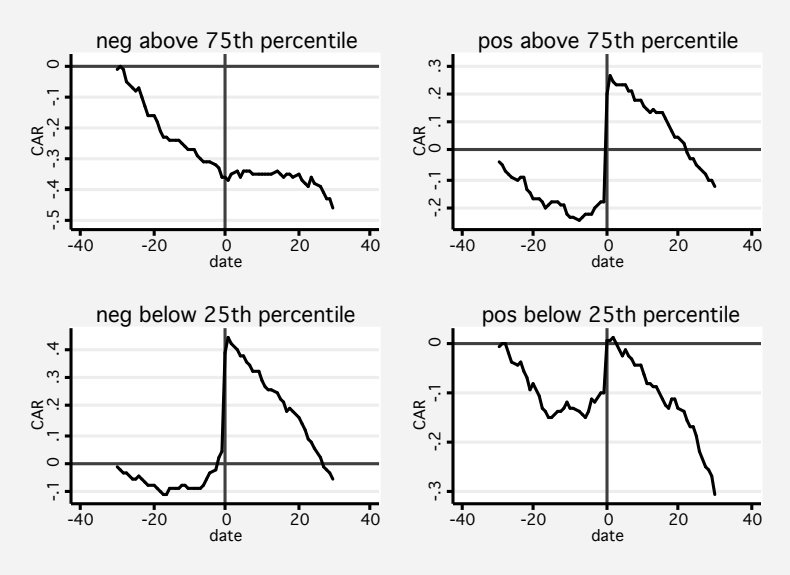

Figure 2.1: Cumulative Returns for different Portfolios

information about upcoming news. The only surprise here is that the low positive portfolio (which is supposed to be a negative portfolio) shows a positive spike. This may be explained by the slogan "no news is good news". The absence of positive words in a piece of news does not imply it is negative. Note that the spike for news with a low number of positive words is much smaller than the spike for the two positive portfolios. Note also that these news are preceded by a series of underperforming returns. The average cumulative abnormal return never crosses above zero for these events.

4. Highly negative news are highly anticipated by the markets. Before the event date, the cumulative log return is below -0.035. This is compared with all the other portfolios, which do not reach a value below -0.025. After negative news are published, cumulative returns stabilize. This may be due to value investors purchasing a security they perceive as oversold. After this period of stabilization, the negative trend continues.

2.4 Limitations

This chapter showed the usefulness of sentiment in financial news. The measures of sentiment we use are simple, but show significant correlation with stock returns. These methods can help us separate events into positive and negative portfolios that have good risk-adjusted excess returns.

This partially answers the second question that motivates this thesis. We can use sentiment analysis to explain and predict market returns. However, it tells us very little about the other questions that motivate this thesis. Can we engineer a method to distinguish between positive and negative texts using market information? Can we learn which words and phrases are important in separating positive and negative news. How do non-linear learning methods compare to word counting?

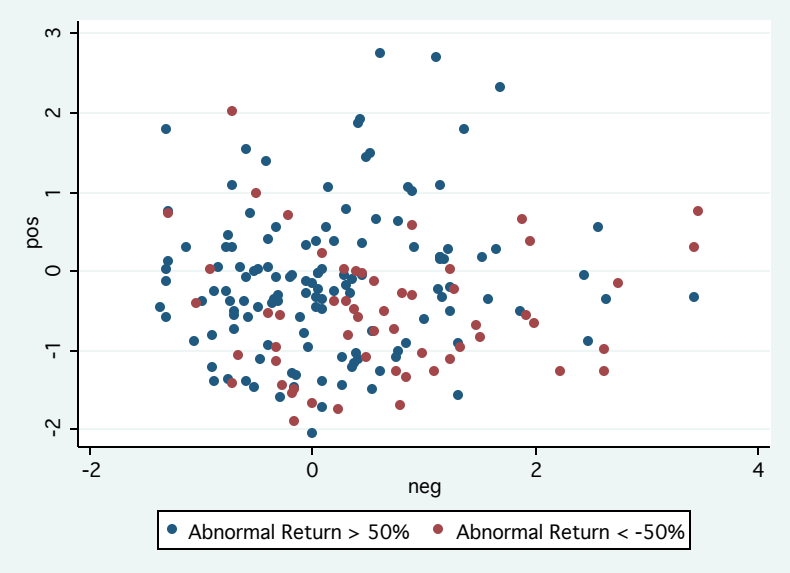

Figure 2.2: The variables **pos** and **neg** for different texts. Colors indicate direction of stock movement.

Figure 2.2 shows that there may be better approaches than word counting. Each point in this figure represents a news item. The x axis is the normalized proportion of negative words (the variable **neg**), the y axis is the normalized proportion of positive words (the variable **pos**). Blue points have a very high abnormal return. Red points have a very low abnormal return. It is hard

to see this picture and come up with a good rule to separate the red points from the blue points. By considering different representations of texts, we may get a better rule to separate positive news from negative news.

The following two chapters explore more sophisticated ways of detecting sentiment. Attribute selection algorithms will help us select words that are meaningful when trying to predict financial sentiment. I will also introduce non-linear methods that map words to sentiment. Chapter 3 provides a brief introduction to these techniques.

These methods are frequently used to perform sentiment analysis in the natural language processing community. I apply them to the financial domain in chapter 4 and show that they are better at predicting future returns than the word counting methods described in this chapter.

Chapter 3

The Natural Language Processing Approach

As we saw in the previous chapter, there is a limitation to the General Inquirer approach. In the first place, it does not reveal which words are important for sentiment classification, since the list of relevant words is determined beforehand. I have also shown that the connection between a news piece and the associated stock return cannot be completely captured just by looking at the number of positive/negative words in the text.

We need a more sophisticated model, which I introduce in this chapter. I test this model in the next chapter.

3.1 The vector model

In chapter 2, I summarized texts using two numbers, the proportion of positive and negative words in each text. In this chapter and the next, I will represent texts as binary vectors. Assume that we are given an ordered list of words $W_1, ..., W_N$ that can be related to sentiment. This may be a pre-selected list of words, like the General Inquirer, or it may be a general list including every word seen in our corpus of news. I represent a given text τ as a vector $(w_1, ..., w_N)$, where $w_i = 1$ if

τ mentions the i^{th} word in our ordered list, and 0 if τ never mentions this word.[1]

As an example, consider how we encode these three documents: "Alice said hello", "Bob said hi", "Carol yelled hello". There are 7 words in this corpus: Alice, said, hello, Bob, hi, Carol and yelled. Let's take this as the order of the words. Then the vectors that correspond to these three documents are $(1,1,1,0,0,0,0), (0,1,0,1,1,0,0)$ and $(0,0,1,0,0,1,1)$.

This process has some problems. First of all, it completely ignores position of words, as well as phrases, collocations and grammatical purpose. The word "good" as an adjective is a much stronger indicator of sentiment than the word "good" as a noun. Many of these limitations can be fixed by adding more features. One example would be adding pairs of words as features called bigrams. The document "Alice said hello" would be represented by 5 bigrams: "Alice", "said,", "hello", "Alice said", and "said hello". The presence/absence of each bigram would be a feature.

Similarly, the word "good" could be split into two features: "$good_{Noun}$" is a feature that equals one when the word appears as a noun and "$good_{Adj}$" is a feature that equals one when the word appears as an adjective. Previous work [Pang, Lee, and Vaithyanathan (2002)], [Genereux, Poibeau and Koppel (2008)] shows that these methods do not provide much improvement in sentiment classification.

There is a reason why adding new features is not always better. The more features we have, the harder it is for our algorithms to learn which features are relevant for classification and which ones are not. For example, assume we are trying to learn a linear function $y = f(x_1, ..., x_n) = \sum_{i=1}^{n} \beta_i x_i$. We can do this by solving a linear system of equations where the β_i are the variables to be solved for and each instance $y_\tau = f(x_{1\tau}, ..., x_{n\tau})$ represents an equation. If we have less instances (equations) than features (variables), our system is underdetermined and we cannot find correct values for all the β_i. Thus, the number of instances required to learn grows at least linearly with the number of

[1]There are alternative schemes for constructing a vector out of a text. The most obvious one is assigning $w_i = n(\tau, i)$: the number of times that the text τ mentions word i. This is called Text Frequency (TF). Another used measure is Inverse Document Frequency (IDF). This is an inverse function of D_i, the number of documents where word i is mentioned. The idea behind IDF is that frequently used words are not as important in classifying documents as infrequently used ones. It is common to combine these measures by multiplying, thus yielding $w_i = TF \times IDF$ [Chakrabarti (2003)]. Studies in sentiment analysis [Pang, Lee, and Vaithyanathan (2002)], [Koppel and Shtrimberg (2006)] have shown that results are not very different if these alternatives are used. I choose the presence/absence model out of simplicity.

features. This also applies when we try to learn non-linear functions. Thus, it is good to reduce the number of features.

To reduce the number of features, I apply stemming. Consider the two texts "The investors reacted positively" and "Investors showed a positive reaction". These texts are not very different, but they only agree on one word: "investors". However, we should consider the words "reacted" and "reaction" as derivatives of the same word: "react". Similarly, "positive" and "positively" both derive from the root "positiv". Reducing words to their roots is a procedure called stemming. It is useful because it allows us to ignore differences in gender, number and conjugation that we do not care about for sentiment classification. On the other hand, stemming introduces noise in our data by making words like "university" and "universal" both reduce to the root "univers".

Another way in which I process these texts is by removing stopwords. Stopwords are frequent words that appear in texts regardless of sentiment. These include words like "a" and "the". Removing these words reduces the number of attributes. This helps our algorithms make better decisions.

Finally, I only keep the features that appear in four or more texts. Features that appear in less than a given number of texts should not be considered useful for prediction, since they are very infrequent.

This is the basic process by which I turn texts into vectors. I implement it using the Natural Language Toolkit [NLTK (2002)]. This process allows me to feed the texts into algorithms that learn what distinguishes a positive piece of news from a negative piece of news. These algorithms can then see new texts and output classifications for them.

3.2 Classification and Regression

In machine learning, we are given a set of features $\vec{x_i} = (x_{1i}, ..., x_{Ni})$ and a corresponding target y_i. The learning task consists of finding a function $y = f(\vec{x})$ that fits this data.

This framework encompasses both regression and classification. In regression, the learned function f is continuous and the output y lies on a scale. In chapter 2, I used regression to explain

stock returns as a function of news sentiment, earnings surprises, share turnover and other variables.

In classification, the learned function f outputs a discrete classification. For example, we can classify a piece of news as positive or negative. Another example is classifying a mole as cancerous or non-cancerous. Classification is useful when the desired outputs do not fit naturally in an ordered scale. In chapter 2, I used a very simple classification method (following [Tetlock, Saar-Tsechansky and Macskassy (2007)]) to classify news. News with a high number of negative words were negative. News with a low number of negative words were positive. This classification method was a good predictor of stock returns. We also classified news based on the number of positive words, but this was not such a good predictor of stock returns. From now on, I focus on counting only negative words.

Figure 3.1 shows the main idea behind classification. In this case, we have two attributes x and y. Blue points are positive, red points are negative. These points are linearly separable. That is, there is a line separating the domain into two half-planes such that all points on one half-plane are positive and all points on the other half-plane are negative. If we can learn the parameters of this line from the given labelled points, we can predict the classification of a new point. This concept can be extended to many dimensions (using a hyper-plane instead of a line). This kind of classifier is called a **linear separator**.

Note that in general the learned classifier will only approximate the true classifier. In the above example, the true classifier is the function that outputs positive if and only if $x + y \geq 1$. The learned classifier may be any function $\alpha x + \beta y \geq \gamma$. We want to choose the parameters so as to minimize the classification error.

Note that the word counting method from chapter 2 is a linear separator with only one attribute. If there are more than θ negative words in a piece of news, it is negative. Otherwise, it is positive.

The motivation for using classification methods in this thesis is simple. All of the questions I am trying to answer are of a discrete nature. Classifying a text as positive or negative is naturally discrete. Predicting future returns may seem like a continuous task, but when it comes time to make

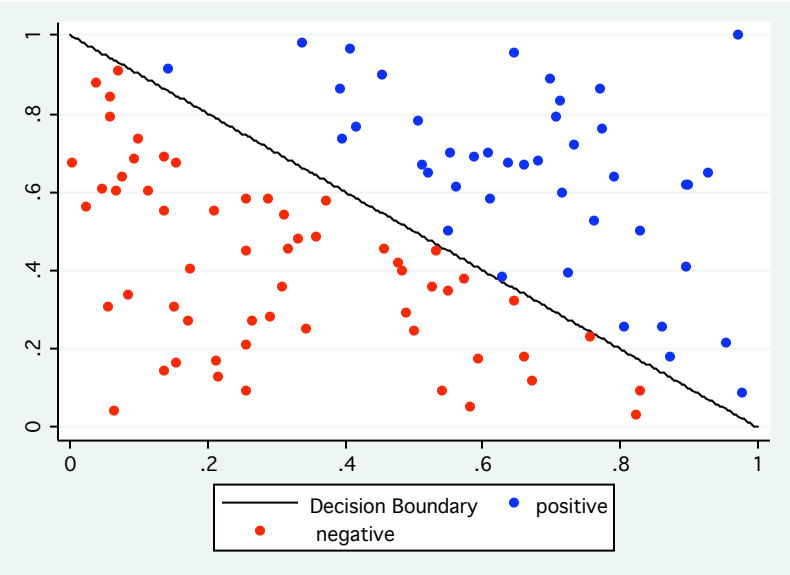

Figure 3.1: An example of a linear separator

trading recommendations, there is only a finite number of choices between strong sell and strong buy. Finally, predicting the polarity of texts outside finance is also a discrete task.

In chapter 4, I will compare the word counting classification method from chapter 2 to other methods used in machine learning. These are the methods that I introduce in the current chapter.

3.3 Feature Selection

There are tens of thousands of words in the Reuters Key Developments Corpus, and hundreds of thousands of words in the English language. Feature selection methods find which of these words are relevant to sentiment and which are irrelevant. An excess of features would lead to our algorithms discovering random patterns. This is why feature selection is necessary.

3.3.1 Information Gain

Let D be a dataset of labelled texts. Let p_D represent the probability that a random text from D is classified as positive. If the set is highly skewed towards positive or negative instances, then classification should be relatively easy. On the other hand, if the set is very disorderly with equal probability of positive and negative instances, then classification is hard. The disorder in the set D is measured by its entropy:

$$H_2(D) = -p_D \log p_D - (1 - p_D) \log(1 - p_D).$$

This can be interpreted as the average number of bits I need to communicate the classification of each item in the dataset. If all the texts are positive ($p_D = 1$), then $H_2(D) = 0$ (assuming $\log 0 = 0$). I need 1 bit of information to tell you all the texts are positive, which translates to an average $\frac{1}{|D|}$ bits per text. This is very close to zero if $|D|$ is large. On the other hand, if half the texts are positive and half are negative ($p_D = \frac{1}{2}$) then I have no choice but to tell you the classification of each text separately. This is $H_2(D) = -\frac{1}{2}(\log_2 \frac{1}{2}) - \frac{1}{2}(\log_2 \frac{1}{2}) = 1$. I need one bit per instance to tell you the classification.

We would like to choose relevant features that help us classify the set D. In the Information Gain framework, a feature is useful if it helps us reduce the disorder in the dataset. We want to choose features that reduce disorder the most.

If we choose a feature x, the dataset is split between instances where x is zero and instances where x is 1. Call these subsets D_0 and D_1. If both of these sets are relatively orderly, then we have reduced disorder. Quantitatively, we have that information gain is measured by

$$IG(D, x) = H_2(D) - \frac{|D_0|}{|D|} H_2(D_0) - \frac{|D_1|}{|D|} H_2(D_1).$$

This is the difference between the entropy in the original dataset D, and the average entropy of the sets D_0, D_1.

The Information Gain Criterion chooses the features $x_1, ..., x_k$ that maximize $IG(D, x)$. It

chooses one feature at a time.

3.3.2 Correlation-based Feature Selection

Mark Hall introduced Correlation-based Feature Selection (CFS) [Hall,1998]. The motivation behind this method is choosing features from $(w_1, ..., w_N)$ that are highly correlated with the classification label, but have low correlation between each other. For convenience of notation, let S be a set of features and let y be the classification variable. Let $\rho_{S,y} = \{|\rho(s, y)| : s \in S\}$ be the set of magnitudes of correlations between the elements of S and the classification label y. Let $\rho_{S,S} = \{|\rho(s, t)| : s, t \in S\}$ be the set of magnitudes correlations between pairwise different elements of t.

The method chooses a subset S of features that maximizes a function $\phi(\rho_{S,y}, \rho S, S)$. Recall that we want the correlation of S with the classification to be high and we want the correlation of S with itself to be low. Thus, the function ϕ should be increasing for variables in $\rho_{S,y}$ and decreasing for variables in $\rho S, S$. The set S that (approximately) maximizes the function ϕ is obtained by local search.

In practice, the function ϕ used is

$$\phi(S, y) = \frac{k\overline{\rho(S, y)}}{\sqrt{k + k(k-1)\overline{\rho(S, S)}}}$$

where k is the number of elements in S, $\overline{\rho(S, y)}$ is the average correlation of elements of S with the class label y and $\overline{\rho(S, S)}$ is the average correlation between elements of S.

This feature selection method provides consistently good results, as I will show in chapter 4.

3.3.3 General Inquirer

For consistency in exposition, I should note the General Inquirer feature selection method. This method simply ignores all the words that are not catalogued as positive or negative by the

General Inquirer dictionary.

3.4 Classification Methods

I will explain how three different classification methods work using five attributes from the movie review domain. I will apply these classifiers to the financial domain in the next chapter. For briefness, I only describe how an algorithm takes an input $(x_1, ..., x_N)$ and outputs a classification y. I do not explain how the parameters of the classifiers are learned. This is analogous to saying that linear regression models y as a linear function of $x_1, ..., x_N$ without mentioning the least squares method. The interested reader can refer to [Russell and Norvig (2003)] for details of the algorithms. To implement the algorithms, I use the open source software Weka 3.5.8 [Witten and Frank, 2005].

3.4.1 Decision Trees

Recall that a text can be represented as a binary vector, each position corresponding to an attribute. In this simplified example, we will consider instances represented as 5-dimensional vectors. The first component will be equal to 1 if the text contains the word "wasted", and will be equal to 0 otherwise. The other 4 components depend on the presence/absence of the words "worst", "bad", "awful", "stupid".

Thus, the text "This is the worst movie I have ever seen. My time was wasted" can be represented as the vector $(1, 1, 0, 0, 0)$. The positive text "I was expecting this movie to be awful, but it turned out not to be bad at all" is represented as the vector $(0, 0, 1, 1, 0)$.

One of the classifiers that I apply to financial news is a decision tree. Figure 3.2 is an example of a decision tree obtained when learning to classify movie reviews based on the five above words.

To classify an instance, we start at the root. The root attribute in this case is the presence/absence of the word "wasted". Instances where this word is absent are classified by following the left branch of the tree. Instances where this word is present are classified by following the right branch of the tree. In this tree, the left branch leads to another rule, while the right branch leads

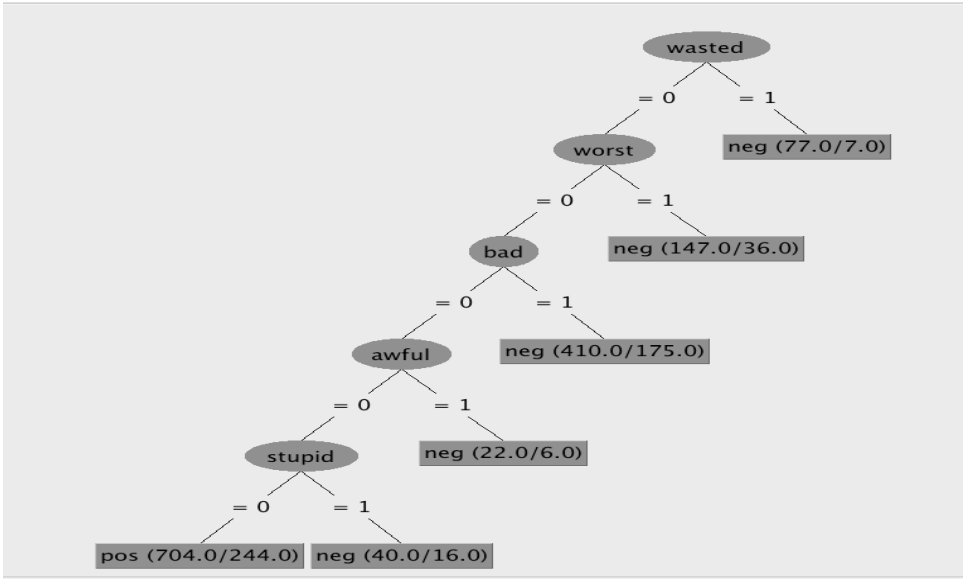

Figure 3.2: An example decision tree

to an immediate classification: the movie review is negative. Once we follow enough branches, we get to a terminal node, which is called a leaf. Each leaf specifies how the text should be classified. The numbers printed in the leaf say how many texts that reached that leaf were classified correctly and how many incorrectly.

If we try to classify the instance "This is the worst movie I have ever seen. My time was wasted," we start at the root and follow the right branch. This leads to an immediate negative classification, which seems accurate in this case. 77 texts that followed this path were classified correctly. 7 were classified incorrectly.

If we try to classify the instance "I was expecting this movie to be awful, but it turned out not to be bad at all," we start at the root and follow the left branch (the word "wasted" is not mentioned). The next rule asks us if the text mentions the word "bad". Since the text mentions this word, we follow the right branch associated with this rule and classify the instance as negative.

This tree is slightly misleading because it contains no positive words. Rather, a positive text is indicated by the *absence* of any negative words. This gives the tree a special structure. If a negative word is mentioned, you automatically classify the instance as negative. If a negative word is

not mentioned, you check for the next negative word. If none of these negative words are mentioned, the instance is positive. In general, decision trees do not need to be unbalanced but this will be a general pattern with the news we analyze. Positive instances are characterized by the absence of any negative words.

The main advantage of using decision trees is that they are easy to interpret. Their main disadvantage is that they are not good at handling co-predictors. These are pairs of features that provide no information about the classification when used separately, but are very informative when combined. Nevertheless, as we will se in the next chapter, Decision Trees perform very well when classifying sentiment in financial news.

3.4.2 Naive Bayes Model

We continue with our example where we have five attributes $X_1, ..., X_5$ corresponding to the presence or absence of 5 different words. I will notate the corresponding classification with the letter C. Naive Bayes and other probabilistic models treat both the attributes and the classification of an instance as random variables. This model of the world assumes that variables are linked to each other, but only probabilistically. If the text mentions the word "disaster", there is a high probability that the overall sentiment is negative. But there is also a slight chance that the sentiment is positive ("disaster was averted"). More importantly, the presence or absence of given words allows us to update our beliefs about the sentiment of the text. Probabilistic methods have three key components:

1. A prior belief about a text's classification. This is represented via a probability distribution $P(C)$.

2. A model that tells us the probability of seeing an instance $X_1 = x_1, ..., X_n = x_n$ given that the class is $C = c$. This model is expressed as a conditional probability distribution $P(X_1 = x_1, ..., X_n = x_n | C = c)$.

3. We can use Bayes' Rule to update our beliefs about a text's classification given the words that it contains: $P(C|X_1, ..., X_n) = \frac{P(X_1,...,X_n|C)P(C)}{P(X_1,...,X_n)}$

The predicted class $\tilde{c}$ is the class with the highest probability of occurring given the attributes.

The problem with the above framework is that we need an absurdly large number of parameters if we do not impose any other conditions. If the attributes $X_1, ..., X_5$ are binary, we have $2^5 = 32$ possible instances. Each instance $x_1, ..., x_5$ corresponds to two parameters of the model: one specifying $P(X_1 = x_1, ..., X_5 = x_5|C = 0)$ and one specifying $P(X_1 = x_1, ..., X_5 = x_5|C = 1)$. The number of examples we need to learn these parameters grows at least linearly with the number of parameters. Thus, if we had 266 variables, we would need more examples than there are atoms in the universe to learn the parameters. This is impossible.

What we need are assumptions that simplify the model. The Naive Bayes assumption suggests that all attributes are independent given the classification. That is, we can write $P(X_1 = x_1, ..., X_5 = x_5|C = 1)$ as $P(X_1 = x_1|C = 1) \cdot P(X_2 = x_2|C = 1) \cdot ... \cdot P(X_5 = x_5|C = 1)$. Using this assumption, the number of parameters is linear in the number of variables. For variable X_i we need to know the values $P(X_i = x_i|C = c)$ where $x_i, c \in \{0, 1\}$. Since x_i and c can only take binary values, we only need to know four parameters.[2] In total, this gives us 20 parameters. If we had 266 variables, we would need to know $266 \cdot 4 = 1064$ parameters. This is much smaller than the 10^{80} parameters we would have without the Naive Bayes assumption. With this number of parameters, it is much easier to learn from examples.

Naive Bayes has proved itself to be a very useful algorithm, even in situations where the assumption of conditional independence does not completely hold. Text categorization is one of these situations.

3.4.3 Support Vector Machines

Support Vector Machines are generalizations of linear classifiers. Their power relies on reducing non-linear classification problems to linear ones. This brief section will only give the most important details. Interested readers can refer to [Cristianini and Shawe-Taylor (2000)] for more details.

[2]For the purists, this is over-determined since we know $P(X_i = 1|C = c) = 1 - P(X_i = 0|C = c)$.

Not every classification problem can be solved by applying a linear separator. Figure 3.3 shows a set of points that is classified using the circle $x^2 + y^2 = 1$. Points outside this circle are positive, points inside are negative. Clearly, there is no line that separates the negative from the positive points. However, we can apply a mapping $(x, y) \rightarrow (z_1, z_2, z_3) = (x^2, y^2, x \cdot y)$. In this case, there is a linear function of z_1, z_2, z_3 that separates the data. It is $z_1 + z_2 = 1$. Thus, there are many cases where we can reduce learning non-linear separators to learning linear separators in higher dimensional space. These mappings are called **kernels**. In particular, the mapping $(x, y) \rightarrow (z_1, z_2, z_3) = (x^2, y^2, x \cdot y)$ is a quadratic polynomial kernel.

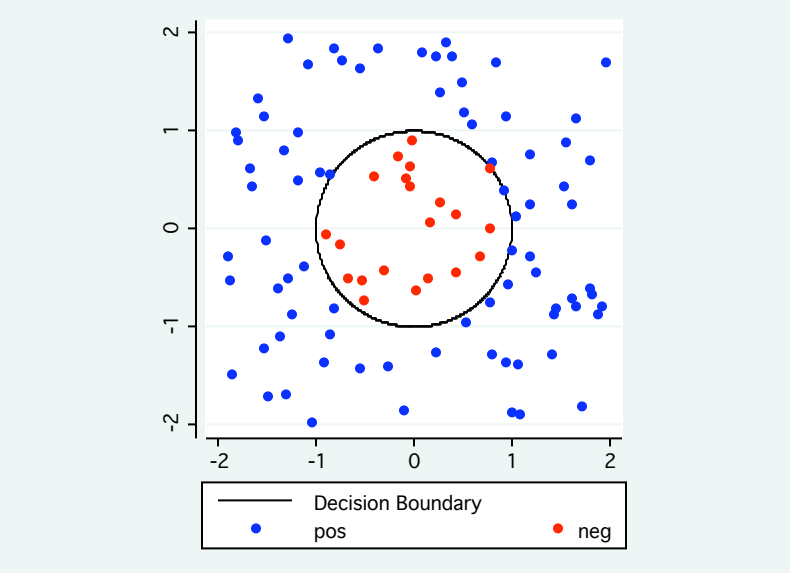

Figure 3.3: A non-linear separator. Examples are positive if $x^2 + y^2 > 1$.

As I mentioned before, adding more variables brings some problems. The most important problem is overfitting. If we have many more features than examples, it is possible that we memorize the data instead of learning a meaningful pattern. Our classifier does very well on the data that we provide it, but poorly on new, unseen data when we release it into the real world.

When we have linearly separable data, there are uncountably many linear separators that we can learn. Some of these linear separators are very close to the positive examples as in figure 3.5. Some of these are very close to the negative examples. Some of these are in the middle as in

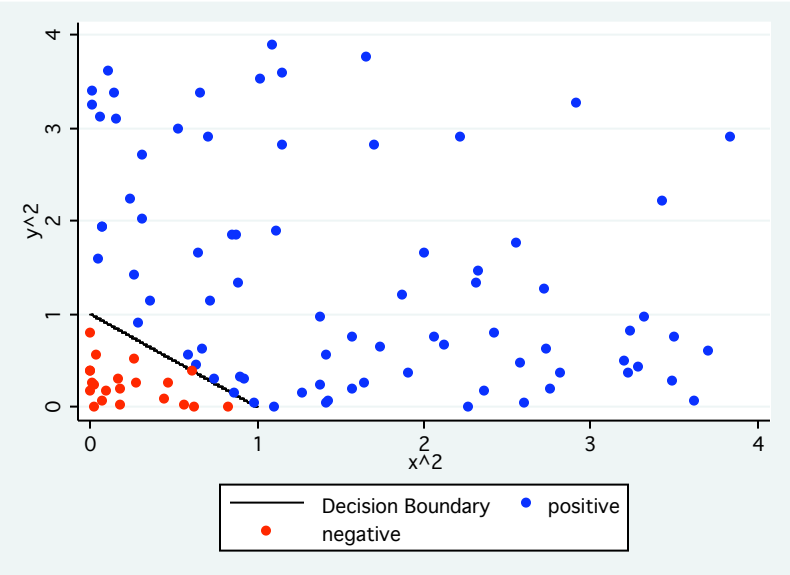

Figure 3.4: Same classifier after we map to the higher dimensional space $(x, y) \rightarrow (x^2, y^2, x \cdot y)$

figure 3.6. We want to find the linear separators that are equidistant from the positive and negative examples. This will help us prevent overfitting.

Figure 3.5 shows the problem with choosing a linear separator that is too tight. This linear separator classifies instances as positive if $x + y \geq 1.5$, while the true linear separator classifies instances as positive if $x + y \geq 1$.

The blue points represent positive training examples, the red points represent negative training examples. The line is the learned linear separator, which is consistent with the training set. Now we get two new points, which are colored orange. The correct classification of these points is positive because they satisfy $x + y \geq 1$. However, because our linear separator is too close to the points from the training set, these points are misclassified as negative.

This does not happen when we choose a linear separator that is equidistant from the positive and negative training examples, as is shown in figure 3.6.

I have only discussed separators that can be expressed as a polynomial function of the attributes. These are captured by polynomial kernels, but there are other types of kernels that capture different types of separators. In this thesis, I only use polynomial kernels. The degree of

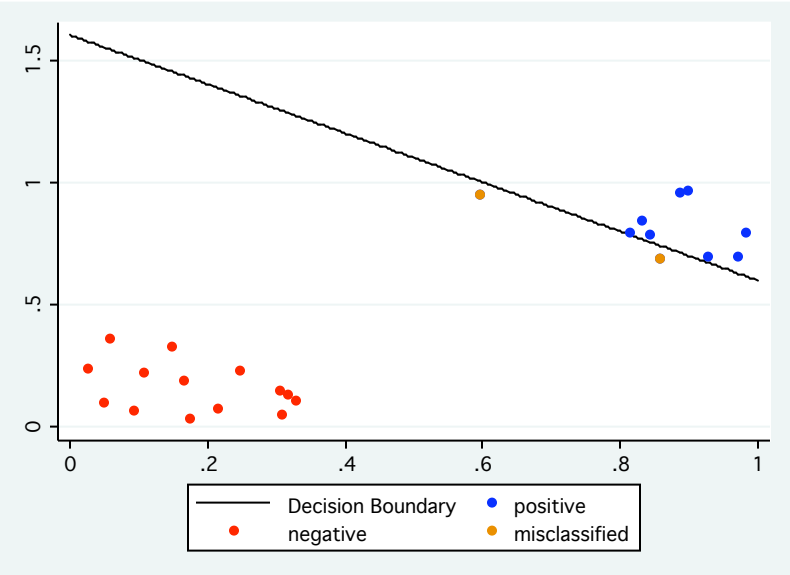

Figure 3.5: A linear separator too close to the positive points. Risk of false negatives (shown in orange)

these kernels is no higher than 2.

3.5 Training and Testing

Given some labelled news, we want to train a model that tells us whether the news are positive or negative. But how do we know our model is working? How do we know if it will generalize to unseen instances?

The key to doing this is to split the data into a training and a testing set. We learn models using the **training set**. Their performance is measured using the **testing set**. This allows us to measure how well the algorithm generalizes to unseen instances. The main measure of performance for a classifier is **accuracy**. This is the proportion of instances in the test set that are classified correctly. If the testing set has 100 instances of which 76 are classified correctly by the algorithm, its accuracy is 76 %.

One may be tempted to train many classifiers and choose the one that performs best on the testing set. Doing this does not guarantee good performance on unseen events. This is a

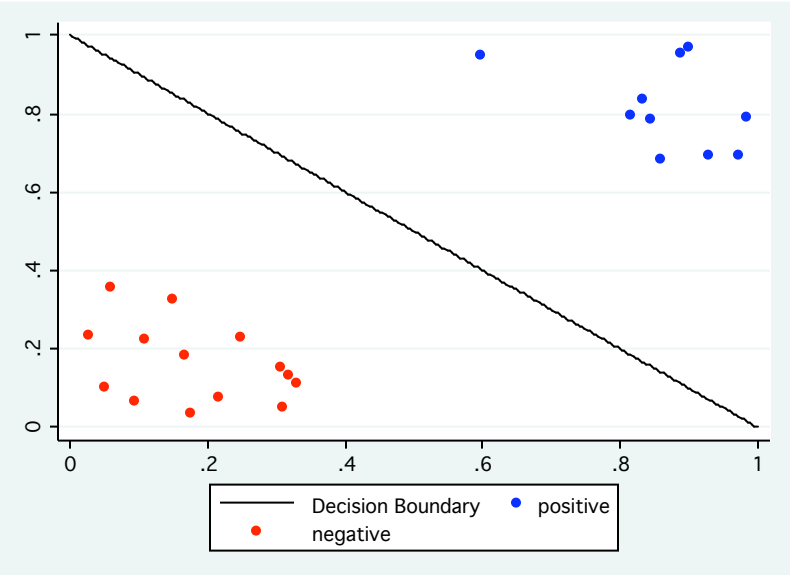

Figure 3.6: A linear separator that maximizes the margin. This has less risk of false negatives or false positives

mistake called **overfitting to the domain** and is analogous to survivorship bias in Finance. A malicious firm could create ten random portfolios, asking friends to invest in them. Nine of them may show mediocre results, and one could show very good results, just by chance. Then, the nine underperforming portfolios are closed and the surviving random portfolio is peddled to investors, based on its previous amazing returns. In this case, it is very obvious that past results are not indicative of future performance.

Similarly in the machine learning case, one could train ten classifiers and evaluate them on a test set, cherry picking the one that performs best. This classifier just may have good results by chance, and its performance may not generalize to unseen instances.

To solve this problem, we need a validation step between our training and testing steps. The data is split into training, validation and testing sets. We train the ten models on the training set, and then measure their performance on the validation set. We choose the classifier that performs best on the validation set. Up to this point, no algorithm has seen the test set, and it is safe to assume that the learned models are independent of the testing instances. Now we can take the chosen classifier and measure its performance on the testing set. Because the classifier was trained

and chosen *without* using the testing data, we can use this as a measure of how well the classifier will perform on unseen instances.

The tradeoff is that we need more instances in our dataset to perform validation. An alternative to this is to use the training set itself for validation. The way to do this is to split the training set into k folds and train the algorithm k times. The first time, the algorithm is trained on the last $k-1$ folds and validated on the first fold. The second time, the algorithm is validated on the second fold and trained on all other $k-1$ folds. We repeat this process k times, validating on each fold exactly once. Thus, each point in the training set is used for validation exactly once. The proportion of points that were classified correctly is the validation accuracy of the algorithm. This process is known as **k-fold cross validation**.

This discussion has been practical and informal, and is all that is needed for the current work. All these notions are formalized by Computational Learning Theory. For many classifiers, there are theorems guaranteeing that if we choose a large number of examples, the classifiers will generalize well to unseen instances with very high probability. The interested reader should check [Kearns and Vazirani (1994)] for more details.

3.6 Conclusion

This section introduced the vector model to represent texts, feature selection methods, classification methods and measures of performance. I will apply all these concepts in the next chapter to the problem of learning to detect sentiment from financial texts and market data.

Chapter 4

A comparison of different methods

In chapter 2, I partially answered whether we can use sentiment in text to predict future returns. We can if we summarize texts using the number of positive and negative words in each text. This is a partial answer because the relation between these numbers and stock returns is fuzzy. The problem did not seem amenable to either classification or regression. We want to know if there are better ways to recommend buying or selling stocks based on language. Also, we want to know if we can learn something about what makes a text positive or negative by looking at market returns associated with news. The General Inquirer approach does not help us solve this question.

This led me to introduce the vector model as well as non-linear classifiers in chapter 3. These methods have been applied frequently to classify texts in natural language processing. In particular, they have been very useful to detect sentiment in movie reviews.

In this chapter, I apply these methods to detect sentiment in financial news. In the process of doing this, I answer the three questions that motivate this thesis:

1. Can we use automatically labelled financial news to learn which news are positive and which news are negative?

2. Can we use what we learned about language to predict future returns? Can we use what we learned to trade? Which methods are best?

3. Do our models extend to other domains beyond finance? If we train a model to detect positive and negative financial news, how well can it detect positive and negative movie reviews?

The answer to the first question is positive. Using Support Vector Machines with linear kernels, we can achieve a classification performance comparable to humans. This is surprising, since one would expect humans to perform better at emotion detection than computers.

The answer to the third question is negative. When we train classifiers using financial news, no method performs better than random guessing when predicting sentiment in movie reviews. In the other direction, when we train classifiers using movie reviews, no method performs better than random guessing when predicting sentiment in financial news.

The second question has a more complicated answer, which depends on whether we consider outliers and whether we can trade before information is revealed to the public. In brief, the semi-strong version of the efficient markets hypothesis is confirmed. A trader with insider information can make amazing returns, but a trader with only public information loses money once we account for transaction costs.

The rest of the chapter shows how I answer these questions. The next chapter provides a conclusion, and suggests avenues for future work.

4.1 Data Collection

The first step towards answering these question is using stock returns to label financial news as positive, negative or neutral. This is analogous to using stars or ratings to label movie reviews. I assume that these labels represent the true sentiment of the news. This assumption may not always be valid. A common market proverb states: "Buy on the rumor, sell on the news". If a positive piece of news is not as positive as was expected, it may provoke a sharp decline. Nevertheless, this automatic labeling method is robust enough to provide significant results in the experiments below.

To measure stock returns, I use the Abnormal Return over the Fama-French three factor model defined in chapter 2. I do this to account for the natural correlations between the security

and the market. Recall that in this model risk is captured by looking at the correlation of a given security with three portfolios:

1. The Market portfolio. The return of this portfolio on a given day is the average return of all securities in the market. This average may or may not be weighted by market value. In this thesis, I use an unweighted average computed over all stocks in the CRSP database.

2. The Small Minus Big (SMB) portfolio. This portfolio accounts for the fact that small firms seem to have larger excess returns than bigger firms. The return on this portfolio on any given day is obtained by buying a pre-determined list of small firms and shorting an analogous list of large firms.

3. The High Minus Low (HML) portfolio. This portfolio accounts for the fact that firms with a high book to market value (value stocks) have larger excess returns than firms with a small book to market value (growth stocks). The return on this portfolio on any given day is obtained by buying a pre-determined list of value firms and shorting an analogous list of growth firms.

In this model, asset i's excess return $R_i - R_f$ is a linear function of the market excess return $R_{Market} - R_f$, the Small Minus Big (SMB) portfolio excess return $R_{SMB} - R_f$ and the High Minus Low (HML) portfolio excsess return $R_{HML} - R_f$.

$$R_i - R_f = \beta_{i1}(R_{Market} - R_f) + \beta_{i2}(R_{SMB} - R_f) + \beta_{i3}(R_{HML} - R_f) + \epsilon_i.$$

ϵ_i represents the difference between the return predicted by the linear model and the actual return. This is the abnormal return.

To label the news, I select two thresholds *low* and *high*. Any piece of news whose associated abnormal return is lower than *low* is labelled negative. Any piece of news whose abnormal return is higher than *high* is labelled positive. Anything in the middle is labelled neutral.

This provides the classification for each piece of news. Choosing the attributes is more complicated. Consider a matrix whose rows represent pieces of news, and whose columns represent words in the english language. The (i, j) entry of this matrix is 1 if news piece i mentions word j, and

is zero otherwise. This matrix has two important properties. First, it has many more columns than rows. Each column represents a word in the English language, and there are hundreds of thousands of these. Second, the matrix is very sparse. Each text mentions only a handful of words.

Because of these properties, inference is hard unless we use feature selection methods. One of these methods is ignoring any words that are not in the list of positive or negative words of the General Inquirer. Another method is to select the words that best predict the news piece's sentiment. The techniques I use for automatic feature selection are Correlation-Based Feature Subset Selection (CFS) [Hall,1998] and Information Gain. They are described in chapter 3.

This process transforms each piece of news n_i into a feature vector $\mathbf{v}_i$ and a classification c_i. To test how well computers can learn to detect sentiment in financial news, I use the data to train different classifier functions. These functions take as input the feature vector $\mathbf{v}_i$ and return a predicted classification $\tilde{c}_i$. The classification is accurate if $c_i = \tilde{c}_i$.

This gives us a natural model with which to answer the questions that motivate this thesis:

1. Can computers learn to detect positive and negative news? To answer this question, I select a set of highly positive and highly negative news. I separate this set into training and testing sets. I train different classifiers on the training set, and choose the one that performs optimally using 10-fold cross validation. To guarantee that I am not cherry picking, and that the chosen classifier performs well on unseen instances, I check its accuracy on a testing set that is disjoint from the training set

2. Can we use the models to predict future returns? To answer this question I separate the data by year. Each model is trained on the data for year t, and makes buy/sell/do nothing recommendations every time it sees a piece of news during year $t+1$. Each stock is bought or shorted for a day. The models are evaluated on the strength of the strategies they recommend. This simulates the development of a real trading strategy, where we learn from what happened last year to trade this year.

3. Can we use the learned models to predict positive/negative documents in other domains? To answer this question, I take the trained classifiers from experiment 1 and test them on the

set of labeled movie reviews used in [Pang, Lee, and Vaithyanathan (2002)]. This dataset is publicly accessible (`http://www.cs.cornell.edu/people/pabo/movie-review-data/`).

I will compare the ability of five learning algorithms in answering these questions:

1. Decision Trees with Pruning
2. Support Vector Machines with polynomial kernel of degree 1
3. Support Vector Machines with polynomial kernel of degree 2
4. Naive Bayes
5. Counting negative words (henceforth referred to as Word Counting).

The first four of these methods have been described in the previous chapter. The last method, word counting, is used in [Tetlock, Saar-Tsechansky and Macskassy (2007)] to construct positive and negative portfolios. Remember that for each piece of news, the variable **neg** measures the proportion of words in the piece that are negative. The algorithm then adds this quantity to a list. When it is finished scanning the instances, it computes the median [1] of all elements in this list. When faced with a new instance, the algorithm computes the proportion of negative words. If it is higher than the the median, then the news is classified as negative. If it is lower than the median, it is classified as positive.

Note that for word counting, I only use the words from the General Inquirer as features. I do not apply automated feature selection with this approach, because these methods do not indicate which words are positive and negative. They just indicate which words are relevant.

[1] I use the median instead of the high and low quartiles because I want the algorithm to output a classification for *all* news in the test set. If the algorithm refused to classify some news, it would be hard to compare to the machine learning algorithms.

4.2 Experiments and Results

4.2.1 Distinguishing between positive and negative

The first question requires us to learn relevant models that predict positive or negative news. We are not required to learn about what makes a piece of news "neutral". Previous literature [Genereux, Poibeau and Koppel (2008)] shows that three-class classification in sentiment analysis is much harder than two-class classification. Furthermore, the performance of learning algorithms improves when there is a wide separation between positive instances and negative instances. For these reasons, my first experiment ignores news classified as neutral. I choose the thresholds for positive and negative classification to be very high. A piece of news is positive if it produces an abnormal return larger than 15%. A piece of news is negative if it prouces an abnormal return lower than -15%.

Classifier	CFS	Gen Inq.	Info. Gain
Word Counting	N/A	61.99 %	N/A
SVM-linear	74.04 %	73.26 %	**75.55 %**
SVM-quadratic	74.56 %	73.49 %	70.88 %
DecisionTree	74.6 %	67.71 %	73.38 %
NaiveBayes	72.26 %	70.63 %	69.53 %

$N = 5166$

Table 4.1: Classifier accuracies with different att. selection methods

Table 4.2.1 shows the accuracy for all combinations of classifier and feature selection methods. Each row is indexed by a classifier, and each column by a feature selection method. The accuracy is computed using 10-fold cross validation. As we can see from this table, the optimal classifier is a linear SVM using the Information Gain Criterion. This is shown in bold.

There are some secondary results to observe from this table. First, all machine learning methods have comparable accuracies, and they all outperform word counting by a wide margin. Second, Correlation-based Feature Subset selection produces overall good results (above 70% accuracy), always outperforming the General Inquirer method of feature selection, but not by much. This suggests that choosing features via the General Inquirer may be a good idea, but aggregating

the 4000 words in this dictionary into a single feature significantly decreases performance.

Information gain does not produce uniform results. It gives the best performance of all methods when used with linear Support Vector Machines, but performs worse than CFS when used with any other classifier function.

We have selected the classifier that performs best using cross validation, but we still run the risk of survivorship bias. Since we trained so many classifiers, linear SVM's may perform well just by sheer luck. How does this model generalize to unseen instances? To answer this question, I look at the performance of the classifier on an unseen testing set. Table 4.2 shows that the model still has a performance that is close to that attained with cross validation.

Classifier	**Accuracy**
SVM – Linear	74.83 %

$N = 1133$

Table 4.2: Testing the optimal classifier

How good or bad is this performance? For comparison purposes, consider the simplest possible classifier: majority rule. This classifier always outputs the most popular class in the training set. In this case, the most popular class is positive. The algorithm that always outputs "positive" regardless of news context achieves an accuracy of 53.3% on the testing set. A sophisticated algorithm should give better results than this basic strategy. All our algorithms clearly outperform this benchmark.

Is there an upper limit on how good a machine trained algorithm can be? Are there news that are simply too confusing to be classified correctly? It is safe to assume that humans are better than computers at detecting sentiment in language. Thus, our algorithms should not perform better than humans at classification. After all, humans can detect many more subtleties in the texts than machines can. Thus, the accuracy of human classifiers can be thought of as an upper bound to the accuracy of our algorithms. To see what this upper bound is, I asked humans to classify 201 randomly selected elements from the testing set using the Amazon Mechanical Turk service[2]. The

[2]This is becoming common in computational linguistics

human classifiers used their knowledge of English to decide whether the given pieces of news moved the market up or down. Each piece of news was classified by two groups.

The first group of people achieved an accuracy of 72%. The second group achieved an accuracy of 69%. If we only consider results where the two groups agree, we get an accuracy of 82.5%. Using a linear SVM, I get an accuracy of 74.8%. This is higher than each group by itself, and only slightly lower than both groups combined. Thus, we can conclude that machines can learn to distinguish positive and negative news from financial data almost as well as humans can. Furthermore, Linear Support Vector Machines are the best method, significantly outperforming Word Counting.

So far, this is not very transparent. Yes, the algorithms can predict positive and negative news with high accuracy, but what have they learned about language? Table 4.3 shows the thirty most relevant words learned with each feature selection method.

The first column shows words learned from the movie review domain using the Correlation-based Feature Subset selection criterion. The second through fourth columns show words in the financial domain using different feature selection methods: CFS, General Inquirer, and Information Gain. Because there is a limited number of words I can show, only the 30 most relevant words are shown for each method. In particular, the General Inquirer method suggests around 4000 features, but only the 30 most relevant are shown. In this case, relevance is ranked using the information gain criterion *only on the words that appear in the General Inquirer dictionary.* This is in contrast to the fourth column, which uses this criterion on all words in the corpus.

I should note that, when I use the term "word", I mean a root of a word. Thus, the words "negative" and "negation" get reduced to the stem "neg". Similarly, the words, "expected" and "expectations" get reduced to the stem "expect". I should also note that the word $\langle number \rangle$ represents the occurrence of any number.

CFS - Movies	CFS - Finance	GeneralInquirer - Finance	InfoGain - Finance
also	analyst	lower	lower
awful	auditor	low	quarter
bad	declin	owe	expect
boring	delay	fal	analyst
despite	due	repo	earn
dull	earn	danc	report
dumb	fall	even	estim
especially	fell	even	guidanc
expect	impact	guid	revenu
extremely	loss	short	shortfal
job	lower	compani	compani
lame	neg	gui	share
looks	quarter	har	fall
mess	rais	share	loss
others	report	fall	multex
performance	result	art	⟨*number*⟩
pointless	shortfal	loss	fell
quite	slowdown	delay	neg
relationship	suffici	agreement	delay
ridiculous	suspend	aver	impact
sometimes	weak	econom	fiscal
stupid	armi	declin	weak
supposed	cdma	neg	due
taylor	dna	educ	result
terrible	ebix	charg	averag
true	netzero	reason	agreement
unfunny	orbit	ill	rang
uninteresting	premium	numb	fourth
waste	rose	primarili	reuter
wasted	settl	continu	wall

Table 4.3: Most relevant words for each feature selection method

A brief analysis shows that the methods are learning important words. Both CFS and Information Gain recognize important actors in the market: analysts, auditors and companies. Negative words play a significant role in all methods. Some of the most relevant words are "lower", "weak", "decline", "shortfall", "suspend" and "delay". There are important words that are strong but convey no emotion, such as "impact". For CFS and Information gain, we start getting spurious words around the 25th position. These are words like "Reuters", "DNA" and "netzero".

Another conclusion is that movie reviews are much more opinionated than financial news. Words relevant to movie reviews are much closer to "stupid", "ridiculous" and "boring" than to "lower", "weak" and "decline". The first set of words is more subjective and judgmental. The second set has a more objective tone.

Table 4.4 shows the intersection of the sets of words learned with different methods. The first words in this table belong to the intersection of all three feature selection methods when used on financial news. That is, words in the General Inquirer dictionary that are deemed relevant by both CFS and the Information Gain Criterion. This requires us to limit ourselves to words in the General Inquirer dictionary. The bottom of he table expands the list using words deemed relevant by both CFS and the Information Gain Criterion, ignoring whether they belong to the General Inquirer. It is hard to argue that any of these words are unrelated to market sentiment.

4.2.2 Predicting Future Returns

So far I have been ignoring neutral news, focusing on highly positive or highly negative news. Using highly positive and negative news helps us train classifiers that predict positive and negative news with relatively high accuracy. However, ignoring neutral news is not a realistic choice. In the real world, our algorithm cannot know if a piece of news will produce an abnormal return higher than 15% or lower than -15%. It has to deal with the possibility that the return will be between those values.

My second experiment accounts for this. The setup is as follows: every calendar year, I observe some news and the associated excess returns. I create a training set out of these news by

Words common to all three selection methods
delay
loss
lower
fall
neg
Words common to CFS and the Information Gain Criterion
impact
earn
shortfal
weak
due
fell
result
report
quarter
analyst

Table 4.4: Intersection of relevant words selected by different methods

choosing texts that lead to a very negative abnormal return or a very positive abnormal return. I train the algorithms using this data. I use each algorithm to construct a trading strategy: assume we get a piece of news about stock i. If the piece of news is classified as positive, buy the stock; if it is classified as negative, short the stock.[3] Close the trade the following day. I test the algorithms using *all* news from the following year. I repeat this for all years for which I have data.

This strategy has undiversifiable risk due to the correlation of stocks with general market movements. Suppose we get good news about IBM in a day when the general market drops 10%. We buy IBM stock but we lose money. Why? Becuase IBM is part of the general economy. When the market goes down, it drags IBM down with it. However, its price decreases by less than 10%, because the information from the good news is factored in. In this case, simply buying IBM is a losing proposition. On the other hand, if we short the market portfolio and use the money we get from the sale to buy IBM stock, we get a positive return. If the situation was reversed, and we obtained bad news about IBM on a really good day, the strategy would short IBM and use the

[3]Recall that shorting a stock is the act of borrowing it from a broker and selling it, with the expectation that its price will go down. The position is closed by purchasing the stock and returning it to the broker. This requires paying interest to the broker and putting up collateral. I assume unlimited short selling.

money from this sale to invest in the market portfolio.

In my strategy, I offset the risk due to correlation with the market by taking a counterbalancing position in treasuries and the Fama French portfolios as described in chapter 2. This creates a synthetic asset with return ϵ_i: the abnormal return of asset i. If our strategy allows us to create synthetic assets $\epsilon_{i_1}, ..., \epsilon_{i_N}$ with $E[\epsilon_i] > 0$ where ϵ_i and ϵ_j are independent, then we can create a portfolio with a positive mean return and risk that tends to zero when N is large.

Recall from our discussion in chapter 2 that we are interested in two quantities. The abnormal return on day zero (the day a piece of news is announced) represents the difference between stock prices *before* and after the news event. It is a good measure of how much immediate impact a piece of news has on an asset. Algorithms that can generate a significantly positive return on day zero would provide a good explanation of what information moves the markets, but would not necessarily imply a profit or an arbitrage opportunity.

For an investor to obtain this return, she should be able to buy the asset the day before the piece of news is announced. This is infeasible without insider information. Thus, I also look at the abnormal return on day one, the day *after* the piece of news is announced. An investor can generate this return by looking at all the news that happen in a given day, putting all her orders in before the market closes, and closing her positions the following day. Algorithms that can generate a significantly positive mean abnormal return would provide evidence against the semi-strong form of the efficient markets hypothesis. The market will not have absorbed all information immediately, and thus a profit can be made.

Labeling

As in experiment 1, the algorithms are trained only on examples that are very negative ($< -5\%$ abnormal return on day zero or -2% abnormal return on day one) or very non-negative ($> 5\%$ abnormal return on day zero or $+2\%$ abnormal return on day one). I select these lower-magnitude thresholds because I am training a new model each year. In experiment 1, I could afford a large distance between the positive and the negative class because there were more than 6000

instances with absolute return greater than 15%. The parameter 5% (or 2% if we are training with returns on day one) gives an abundant number of news per year (around 4000), but not so abundant as to make training very onerous.

There is one more reason for choosing a smaller threshold. Since we are testing on *all* news, a smaller threshold implies that our training set will be closer to reality. Ideally, we would want to make this threshold zero, but preliminary tests (with other data) show such a small threshold confuses the learner. It is outside the scope of this work to fit for the best threshold.

I only use CFS for feature selection. The only exception is the word counting strategy, where I use the words from the General Inquirer list.[4]

Results

Figure 4.1 shows the mean return for different classifiers. The training set consists of news labelled with the abnormal return for the day *after* news are published.

Each line represents a different model. The blue line represents Decision Trees, the red line represents Quadratic Kernel SVM's, the green line represents Naive Bayes and the orange line represents Word Counting. This figure shows that there is no one algorithm that consistently outperforms the others. Word counting outperforms the machine learning algorithms between 2000 and 2005, but decreases in performance after 2006.

This is surprising. The machine learning algorithms outperform word counting when learning to distinguish very positive news from very negative news (this is experiment 1, described in the previous section). However, when we use them to trade they seem to perform no better, and even worse than word counting for many years.

One explanation is that word counting performs better on outliers. If we look at the median return instead of the mean return (figure 4.2), we see that word counting frequently underperforms the machine learning algorithms.

From this analysis, it seems that machine learning methods perform well in most cases,

[4]Note that I use around 4000 features for the General Inquirer approach, instead of just the 30 most relevant that I showed in the previous section.

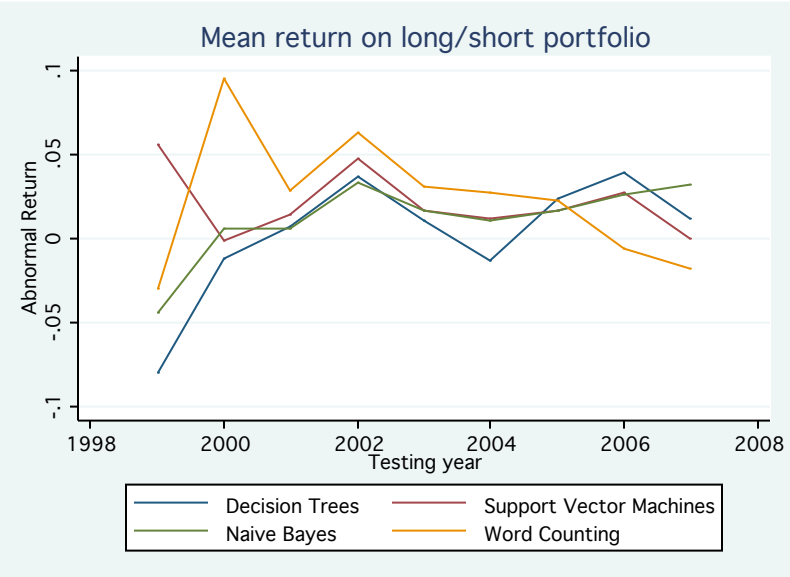

Figure 4.1: Mean returns for different classifiers trained and tested on day one

but are making very costly mistakes when classifying outliers. One explanation for this is training set noise. Recall that the learning algorithms take as input a training set. This training set is a list of news pieces with their corresponding classifications. The classifications depend on the abnormal return of the associated stock on the day after news are announced. The algorithms learn patterns relating the text to the classification.

There is no guarantee that the classification label will be at all related to the text. In fact, the efficient market hypothesis suggests that the classification should be independent from the text: why should I be able to predict today's returns with news that came out yesterday? Even if the efficient markets hypothesis is not completely true, the amount of noise in the labels may be enough to make our algorithms learn random patterns. These random patterns do not generalize to the testing set, and make bad recommendations.

The word counting approach will not have the problem of learning random patterns due to noise in the classification. The reason is that this approach derives all its information from the *texts* in the training set, and none from the classification labels. If we changed the classification labels, the word counting strategy would still make the same recommendations: buy on news that are less

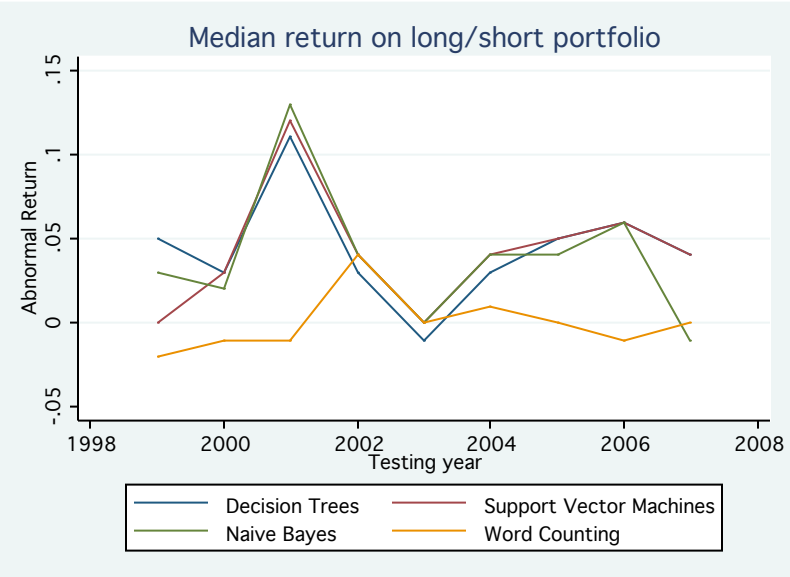

Figure 4.2: **Median** returns for different classifiers trained and tested on day **one**

negative than last year's news; sell on news that are more negative than last year's news.

Figure 4.3 and table 4.5 give evidence that the machine learning algorithms are being confused by noisy classification labels. They show what happens when we *train* using abnormal returns from day zero and *invest* on day one. Note that no insider information is required to train using abnormal returns from day zero, because we always train on past data. These strategies are feasible, because positions are opened only after news are announced. Furthermore, they seem to generate positive returns.

Learner	Mean	Std. Dev.	95 % confidence interval	Annualized
Decision Tree	0.04 %	0.009 %	[0.022 %, 0.057 %]	10.5 %
Quadratic SVM	0.038 %	0.009 %	[0.021 %, 0.056 %]	9.9 %
Naive Bayes	0.032 %	0.009 %	[0.014 % , 0.049 %]	8.3 %
Word Counting	0.026 %	0.009 %	[0.009%, 0.044%]	6.7 %

Table 4.5: Returns for a trader without inside information

Table 4.5 confirms that the strategies generate significantly positive returns. In fact, the machine learning strategies slightly outperform word counting. Decision Trees are the best strategy, with an annualized return of 10.5%. In contrast, the Word Counting strategy yields an annualized

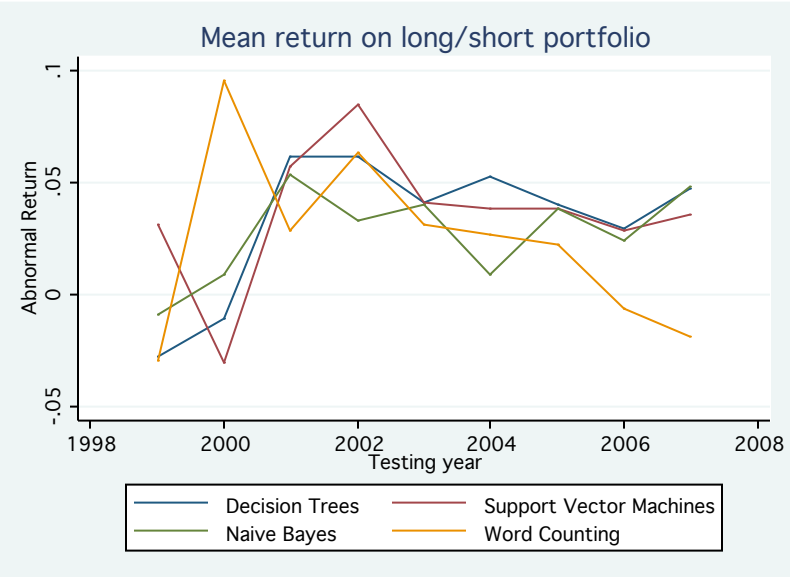

Figure 4.3: Mean return for classifiers trained on day zero, tested on day one

return of 6.7%. However, we have to take into account the consequences of trading costs. Even if the average trading cost was 0.005% per trade, our strategies would not be profitable. This seems to confirm a form of semi-strong efficiency. You cannot generate profits based on public information, once you account for trading costs.

Can we overcome the limitation of trading costs if we have insider information? Figure 4.4 is evidence for this. This figure shows the mean return that is achieved when we open a position the day *before* a piece of news is announced, and close it on the day news is announced. As mentioned before, this strategy is infeasible. However, it gives us a measure of what a trader with insider information can achieve. Machine learning methods significantly outperform word counting. Decision Trees again seem to be the best choice.

Table 4.6 shows the average performance of each method for traders with insider information. The results are significantly positive. We can see that the 95% confidence intervals are far away from zero. For the machine learning strategies not to be profitable, the average trading cost should be over 60 basis points per transaction.

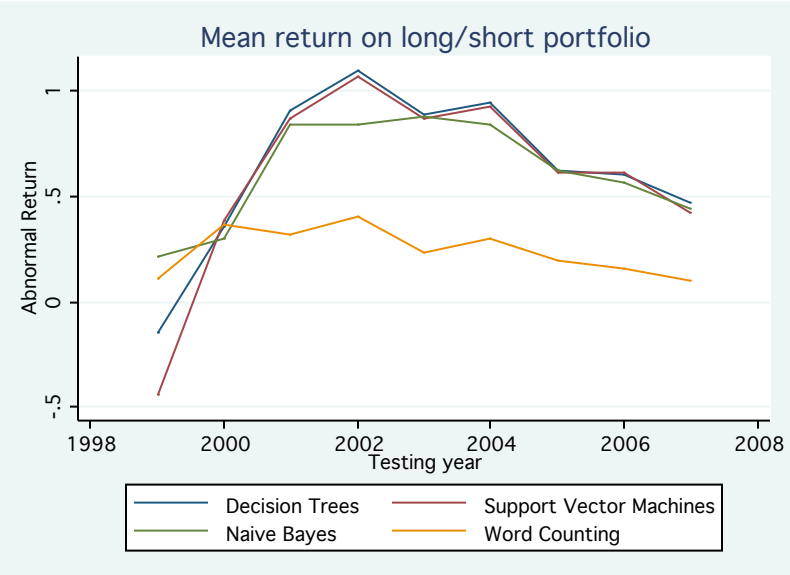

Figure 4.4: Mean returns for different classifiers trained and tested on day **zero**

Learner	Mean	Std. Dev.	95 % confidence interval	Annualized
Decision Tree	0.72 %	0.015 %	[0.693 %, 0.752 %]	501 %
Quadratic SVM	0.700 %	0.015 %	[0.671 %, 0.7294 %]	472 %
Naive Bayes	0.661 %	0.015 %	[0.632 % , 0.690 %]	419 %
Word Counting	0.253 %	0.015 %	[0.224%, 0.282%]	88 %

Table 4.6: Returns for a trader with inside information

4.2.3 Applying what we learned to other domains

My third experiment answers whether emotional language learned from the domain of financial news can be applied to other domains. I train classifiers on financial news and test them on movie reviews. Conversely, I train classifiers on movie reviews and apply them on financial news. The accuracy of these classifiers is a measure of how well emotional language translates across domains.

As table 4.7 shows, classifiers learned on financial news do not translate well to other domains. The accuracy of all trained classifiers on the movie review dataset is indistinguishable from 50%, which is the accuracy one would expect to get by flipping a random coin. Similarly, table 4.8 shows that training classifiers on movie reviews gives no insight on whether financial news are

positive or negative.

Classifier	Accuracy
SVM – linear	49.21%
SVM – quadratic	47.78 %
DecisionTree	49.57 %
NaiveBayes	50.14 %

Table 4.7: Training on financial news, testing on movie reviews

Classifier	Accuracy
SVM – linear	43.46%
SVM – quadratic	45.11 %
DecisionTree	53.72 %
NaiveBayes	46.66. %

Table 4.8: Training on financial news, testing on movie reviews

These results are understandable if we look at the words that are learned by our feature selection algorithms. The only word that the movie review domain has in common with the financial news domain is "expect".

4.3 Conclusion

Can we learn something about language from financial news? The answer is a qualified yes. On the one hand, machine learning classifiers are almost as good as humans when predicting which news will lead to highly positive market returns and which news will lead to highly negative market returns. The words learned include "impact", "fall", "shortfall", "neg", and "weak", among others. These words clearly convey information about a piece of news' polarity and strength, especially in the negative direction.

On the other hand, the learned classifiers are not useful outside the domain of finance. When tested against the movie review dataset, the classifiers did not perform better than random guessing. All of the learn words are specific to the field of finance, with the exception of "expect".

Can we predict future market movements using what we learned about language? If an

investor has insider information and is able to trade before news are announced, she can multiply her money five times each year by using machine learning methods. But if the investor is limited to trading after news are announced, then no strategy provides positive returns once we account for transaction costs. Machine learning algorithms always perform better than word counting.

These results answer the questions I set out to answer at the beginning of this thesis. The next chapter provides a conclusion, and suggests avenues for future work.

Chapter 5

Conclusion and Future Work

5.1 Summary

This thesis compared different tools for sentiment analysis in finance. In the process of comparing these tools, I showed that:

1. Counting the number of positive and negative words in a piece of news is a useful technique for explaining the market's reaction to new information. However, methods from machine learning are better at predicting future returns.

2. We can use automatically labelled financial news to train classifiers that distinguish between positive and negative news. Counting the number of positive and negative words performs better than random guessing at this task. Machine learning classifiers perform significantly better than word counting. The performance of Support Vector Machines is comparable to human performance.

3. Machine learning feature selection methods are good at detecting negative words such as "fall", "decline" and "weak", as well financial actors like "analyst", "auditor" and "company". These methods are not so good for detecting positive words.

4. Traders with insider information can make significant returns by buying before positive news are announced and selling before negative news are announced. Traders with this information can get annualized returns higher than 400% if they use machine learning methods. Using word counting methods still yields an impressive 88% annualized return.

5. Investors who do not have insider information *cannot* make significant returns by trading after news are announced. A strategy based on Decision Trees provides a return of 10.5%, but this return is not significant if we account for trading costs. This supports the semi-strong version of the efficient markets hypothesis. One cannot make a profit by trading on publicly available information once we account for trading costs.

6. We cannot use algorithms trained on financial news to predict sentiment in movie reviews. Conversely, we cannot use algorithms trained on movie reviews to predict sentiment in financial news.

5.2 Related and Future Work

There have been many contributions to the field of language and finance that are outside of the scope of this thesis. Most of them come from the natural language processing community. Sanjiv Das and Mike Chen propose an algorithm that looks at stock message boards, relating sentiment in these posts to price level in a technology stock index [Das and Chen (2007)]. It would be interesting to see the interaction between message board postings and news. Which news get people talking? Is this surge in popular interest related to changes in market prices?

Another avenue for research would use more sophisticated linguistic concepts. In this work, I only used words as features and ignored word position, synonymity, grammatical function and other characteristics of the text. Knowles [Knowles (1996)] performed an analysis of news in the Financial times and showed that health metaphors are frequently used to describe the state of the markets. Morris, Sheldon, Ames and Young distinguish between news that describe market prices as animate agents and news that describe market prices as inanimate objects that follow a trajectory [Morris,

Sheldon, Ames and Young (2007)]. Koppel et. al. evaluated the use of health and agent metaphors as features in sentiment detection [Genereux, Poibeau and Koppel (2008)]. The authors observed that these metaphors were not good features for predicting sentiment.

Stephen Chang, Dimitrios Antos, Shrenik Shah and I [Antos, Azar, Chang and Shah (2008)] attempted to train algorithms that learn whether pieces of news are relevant or not. This work used volume instead of stock returns to label news, and did not obtain meaningful results.

Apart from these directions, I believe that there are two very promising lines of future work.

The first research direction consists of testing existing models using language data. There exist many models that attempt to explain investor under-reaction and over-reaction to news. One of these models is proposed by Barberis, Shleifer and Vishny [Barberis, Shleifer and Vishny (1998)]. In this model, investors react to the strength of the information they are presented, but pay no attention to its statistical weight. It would be interesting to test this assumption by developing notions of strength and weight that are based on language. If we can obtain these measures, we can verify how strong words with little statistical weight influence prices.

We can also attempt to analyze sentiment on an intraday time scale. If we have access to news data with extremely accurate time-stamps, we can see how the order books react to news announcements. Baker and Stein [Baker and Stein (2004)] have a model where investor sentiment explains liquidity, as measured by low bid-ask spreads and high trading volume. If we can get news data at this level of detail, it would be interesting to see how liquidity behaves before and after news are announced.

The second avenue takes into account that English is only one of the world's languages. Can we use our methods to detect positive and negative news in other languages? What is the relation between Colombian news and the price of coffee? What can machines learn about the Arabic language by linking news releases to the price of oil? Machine learning methods would be key to answering these questions without having access to a team of international linguists.

5.3 Conclusion: Expanding the Economics Toolbox

My motivation for writing this thesis is exploring the power of computer science tools to solve problems in economics. My evidence and results are limited to the domain of finance and news, but the ideas in this thesis should suggest potential for future research. As rational actors, we use language every day. Our interactions, whether at home, at the workplace or in the markets, cannot be fully understood without a grasp of how we use language. I hope I have convinced the reader that economics can make progress on important questions by borrowing some tools from computer science and linguistics.

Bibliography

[Agrawal and Kamakura (1995)] "The Economic Worth of Celebrity Endorsers: An Event Study Analysis". J Agrawal, WA Kamakura. *Journal of Marketing*. 1995.

[Antos, Azar, Chang and Shah (2008)] "Relevant or Irrelevant? Let the market decide". Dimitrios Antos, Pablo Azar, Stephen Chang and Shrenik Shah. *Manuscript*. 2008.

[Baker and Stein (2004)] "Market liquidity as a sentiment indicator". Malcom Baker and Jeremy C. Stein. *Journal of Financial Markets*, Vol. 7. 2004.

[Barberis, Shleifer and Vishny (1998)] "A model of investor sentiment". Nicholas Barberis, Andrei Shleifer and Robert Vishny. *Journal of Financial Economics*, Vol. 49. 1998.

[Bernard (1993)] "Stock Price Reactions to Earnings Announcements: A Summary of Recent Anomalous Evidence and Possible Explanations", Victor L. Bernard in *Advances in Behavioral Finance*. Ed: Richard H. Thaler. Russel Sage Foundation. 1993.

[Cutler, Poterba and Summers (1993)] "What moves Stock Prices?", David M. Cutler, James M. Poterba and Lawrence H. Summers in *Advances in Behavioral Finance*. Ed: Richard H. Thaler. Russel Sage Foundation. 1993.

[Campbell, Lo, MacKinlay (1997)] *The Econometrics of Financial Markets*. John Y. Campbell, Andrew W. Lo and A. Craig MacKinlay. Princeton University Press. Princeton, NJ. 2007

[Chakrabarti (2003)] *Mining the web: Discovering Konowledge from Hypertext data*. Soumen Chakrabarti. Morgan Kauffman Publishers. San Francisco, CA. 2003.

[Cristianini and Shawe-Taylor (2000)] *An Introduction to Support Vector Machines and OtherKernel-based Learning Methods*. Nello Cristianini and John Shawe-Taylor. Cambridge University Press. Cambridge, UK. 2000.

[Das and Chen (2007)] "Yahoo! for Amazon. Sentiment Extraction from Small Talk on the Web". Sanjiv Das and Mike Chen. *Management Science*. Vol. 53, No. 9, September 2007.

[Dellavigna and Pollet (2008)] "Investor Inattention and Friday Earnings Announcements". Stefano Dellavigna and Joshua Pollet *Journal of Finance*. Forthcoming. 2008.

[Doyle and Magilke (2008)] "The Timing of Earnings Announcements: An examination of the Strategic Disclosure Hypothesis". Jeffrey Doyle and Matthew Magilke. *The Accounting Review*. Vol 84, No. 1. 2009.

[Fama and French (1992)] "The Cross-Section of Expected Stock Returns". Eugene Fama and Kenneth French. *Journal of Finance*. Vol 47, No. 2. 1992.

[Genereux, Poibeau and Koppel (2008)] "Sentiment analysis using automatically labeled financial news". M Genereux, T Poibeau, M Koppel. In *LREC 2008 Workshop on Sentiment Analysis*. Marrakesh, Morocco. 2008.

[Hall,1998] *Correlation-based Feature Subset Selection for Machine Learning*. M. A. Hall. Ph. D. Thesis. Hamilton, New Zealand.

[Kearns and Vazirani (1994)] *An Introduction to Computational Learning Theory*. Michael Kearns and Umesh Vazirani. MIT Press. Cambridge, MA. 1994.

[Knowles (1996)] "Lexicographical aspects of health metaphors in financial texts". Francis Knoles. In *Proceedings Part II of Euralex*. 1996.

[Koppel and Shtrimberg (2006)] "Good News or Bad News? Let the Market Decide", Moshe Koppel and Itai Shtrimberg in *Computing Attitude and Affect in Text: Theory and Applications*. Eds: James Shanahan, Yan Qu, Janyce Wiebe. Springer, Netherlands.

[Morris, Sheldon, Ames and Young (2007)] "Metaphors and the market: Consequences and preconditions of agent and object metaphors in stock market commentary". Michael W. Morris, Oliver J. Sheldon, Daniel R. Ames and Maia J. Young. *Journal of Organizational Behavior and Human Decision Processes*. 2007.

[NLTK (2002)] "NLTK: The Natural Language Toolkit", Edward Loper and Steven Bird. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics - Volume* . 2002.

[Pang, Lee, and Vaithyanathan (2002)] "Thumbs up? Sentiment Classification using Machine Learning Techniques. Lillian Lee, Bo Pang and Shivakumar Vaithyanathan. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. Philadelphia, PA. 2002.

[Russell and Norvig (2003)] *Artificial Intelligence: A Modern Approach*. Stuart Russell and Peter Norvig. Prentice Hall Series in Artificial Intelligence. Second Edition. 2003.

[Stone (1966)] *The General Inquirer: A computer approach to content analysis*. P.J. Stone. MIT Press. Cambridge, MA. 1966.

[Tetlock (2007)] "Giving Content to Investor Sentiment: The Role of Media in the Stock Market". Paul Tetlock. *Journal of Finance*. Vol. 62, 1139-1168. 2007

[Tetlock, Saar-Tsechansky and Macskassy (2007)] "More than words: Quantifying Language to Measure Firms' Fundamentals". Paul Tetlock, Maytal Saar-Tsechansky, and Sofus Macskassy *Journal of Finance*. Forthcoming.

[Witten and Frank, 2005] *Data Mining: Practical Machine Learning Tools and Technique*, Ian H. Witten and Eibe Frank, 2nd Edition. Morgan Kaufmann. San Francisco, CA. 2005. `http://www.cs.waikato.ac.nz/ml/weka/`.

[WRDS] *Wharton Research Data Services Website*. `https://wrds.wharton.upenn.edu`.