Mining Unstructured Financial News to Forecast Intraday Stock Price Movements

Master Thesis

presented by
Simon Bacher
Matriculation Number 1306810
Supervisors: Dr. Johanna Völker, Markus Doumet

submitted to the
Lehrstuhl für Künstliche Intelligenz
Prof. Dr. Heiner Stuckenschmidt
University Mannheim

October 2012

*“We conclude that markets are very efficient,
but that rewards to the especially diligent, intelligent,
or creative may in fact be waiting.”*

Bodie et al. (1989), p. 371

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

The legendary investor Daniel Drew (1797–1879) was quoted as saying "Anybody who plays the stock market not as an insider is like a man buying cows in the moonlight" (Renehan, 2004). The efficient market hypothesis (EMH) introduced a century later by Fama (1965) seems to support this statement. The semi-strong form of the EMH states that it is impossible to earn excess profits from publicly available news and announcements. However, in the past decades strong evidence has been discovered that challenges the EMH. For instance, the empirical works of Tetlock (2007), Chan (2003), and Bernard & Thomas (1990) raise the claim that news articles and announcements are able to directly influence the market's supply-demand equilibrium (Munz, 2011). As a consequence, most professional investors rapidly adjust their trading behavior to the latest news. They typically have access to a newswire service.

With the emergence of information technology such as the World Wide Web, a large amount of data becomes available to us all. According to estimations, the information available worldwide doubles roughly every 20 months (Frawley et al., 1992). In particular, human investors have access to an increasing amount of digital financial news articles that potentially influence future security prices. Farhoomand & Drury (2002) confirm the information overload that market participants experience. Thus, their ability to make rational and fast trading decisions depends highly on the process of selecting information most relevant to them (Mitra & Mitra, 2011).

Recently, text mining has received growing attention as a means to analyze unstructured textual data. One of the main advantages of text mining is the ability to process large amounts of text quickly therefore leaving more demanding tasks to humans (Dörre et al., 1999). There are many promising attempts to tap the potential of text mining in practical applications. Examples include government agencies being able to detect terrorist networks by linking unstructured information or banks tracking their customers' spending behavior more efficiently (Fan et al., 2006). It seems natural to apply the text mining task within the financial domain to address the information overload problem while supporting investors' trading decisions.

In this thesis, we aim to develop a system that forecasts short-term stock price movements using text mining techniques. With regards to the forecasting problem we tackle the challenge of choosing input data and processing it accordingly. We perform adjustments of state of the art text mining techniques in order to maximize the prediction performance. We also address the issue of, from a financial point of view, evaluating the prediction performance.

The remainder of this introduction is structured as follows. First, we present the key research questions in this thesis (Section 1.1). Then we provide an overview of our contributions with respect to these research questions (Section 1.2). Last, we give an outline of the thesis (Section 1.3).

1.1 Research questions

1. In the introductory section, we argued that state of the art text mining techniques potentially provide the means to resolve the information overload problem faced by market participants today. This leads us to our first research question: Is it possible to forecast short term stock price movements using text mining techniques on unstructured financial news?

2. Text classification is a promising text mining task suitable for analyzing textual data. It includes building a function (classifier) from a set of labeled data in order to label previously unseen data. The success of such a classifier highly depends on the quality of the input data it operates on. There are different types of news that can serve as input data, such as regulated or unregulated news. News articles are published by different sources and their reliability and relevance for the stock price vary. In addition, the news articles themselves can be assumed to be not equally relevant for the stock price movements. For instance, a news article published at a weekend might have another effect on the stock prices than a news article published during a trading day. Similarly, news articles dealing with issues such as dividends or major product announcements might be particularly important for the stock price. This leads us to the following question: Which news data must be selected to provide a classifier with training data that is as relevant and noise free as possible?

3. An important prerequisite of building a classifier is to label the input data, in our case the news articles. An automatic labeling requires the alignment of the news articles with the relevant stock prices. Factors that are essential for this alignment process include the time window used to assess the news' price influence and the approach to calculate the price changes after publishing the news. As a result, these factors influence the characteristics of the training data set. They determine not only the quality of the news labeling, but also the amount of different labels and the distribution of the data with respect to the different labels. This in turn influences the expected prediction quality. The following question arises: How can the news data be automatically labeled to build a classifier that achieves a high prediction performance?

4. The ambitious goal of forecasting stock prices using text mining techniques requires achieving the highest possible prediction performance. This involves the challenge of extracting the characteristics (features) from the news articles that are essential for their semantics. Reducing the number of features using different metrics might as well influence the performance. Moreover, it is important to address the problem of a highly unbalanced training data set. Since different classifiers have shown varying performance in the past, making the right classifier choice suitable for our specific goal is necessary. An important question to ask is therefore: How do state of the art text mining techniques help to improve the classifier's prediction performance?

5. A goal highly relevant for the practical application is to achieve monetary profits based on the price forecasting. Thus, the predictions made by the classifier need to be translated into a trading strategy that can be executed in real markets. To ensure the profitability of such a strategy, it must be tested under real market conditions. Trading volume restrictions and trading costs need to be taken into account. The arising key question is: Do the predictions made by a classifier lead to significant excess profits based on realistic market assumptions?

1.2 Contributions

In this thesis, we develop a system that analyzes unstructured financial news using text classification in order to forecast stock price trends. We review similar systems to build on successful ideas and combine them with novel approaches (Question 1). We discuss the different types of news that are potentially relevant to the stock prices and choose news sources for the system accordingly. To eliminate irrelevant news, we present suitable filtering approaches such as the implementation of a rule-based thesaurus (Question 2). We develop an automatic labeling approach and compare it to a manual labeling approach. We evaluate the influence of different automatic labeling approaches on the prediction performance (Question 3). For the training, we introduce a set of features novel with respect to the price forecasting task. We compare different text mining techniques such as the feature vector dimensionality reduction and different classifiers (Question 4). To answer Question 5, we investigate the influence of trading costs on potential profits and run a market simulation that is able to support or reject the practical profitability of the system.

1.3 Outline

The remaining parts of this thesis are organized as follows.

In Chapter 2, we introduce the theoretical foundations to provide a background understanding of the following chapters of the thesis. In Section 2.1, we focus on the financial foundations. We explain basic characteristics of market trades and introduce the phenomenon of trading costs (Section 2.1.1). In Section 2.1.2, we discuss the different types of news articles and explain their potential influence on the stock prices from the viewpoint of the efficient market hypothesis. We explain the event study methodology as a means to test the efficient market hypothesis and to investigate the speed of stock price adjustments as a reaction to new information. In Section 2.2, we introduce the foundations of text mining. After an overview of the different ways to transform a document into a feature vector, we explain all necessary text preprocessing steps. We describe the process of optionally weighting the particular features and reducing the feature vector dimensionality by eliminating less relevant features. We compare different classifiers that can be used to predict the stock price influence of unlabeled news. Finally, we introduce the most common means to evaluate the prediction performance.

In Chapter 3, we provide a comprehensive review of relevant systems developed in the last two decades. We conclude this chapter with a comparison of the systems' key characteristics, and summarize key findings that might help to build on successful ideas and address existing gaps using new approaches.

In Chapter 4, we present the development of our trading system. After providing an overview of the system design (Section 4.1), we describe the system implementation in detail (Section 4.2). We explain the process of preparing the data (Section 4.2.1), which includes acquiring the news and price data, extracting the news contents, filtering out irrelevant news, and converting the news into a proper format to make them ready for the subsequent steps. We then describe the process related to training the data (Section 4.2.2). This includes automatically labeling the news using the stock price data, extracting a set of features, applying methods for handling unbalanced data, and finally training the classifier to predict the labels of unseen news articles.

In Chapter 5, we describe how we evaluate the performance of the system. After introducing the evaluation settings and the evaluation methodology (Sections 5.1 and 5.2), we present the results of the classifier evaluation separated for two different data sets, each resulting from a different labeling approach. For both data sets, we present a parameter tuning in order to increase the performance (Section 5.3). Finally, we evaluate the performance of the system financially by performing a trading simulation (Section 5.4).

In Chapter 6, we provide a summary of the thesis and point out ideas for potential future work.

Chapter 2

Theoretical foundations

In this chapter, we describe the foundations that will be used to design a price forecasting trading system. In Section 2.1, we provide an overview of the financial basics that are required when trading with the goal of short-term profits. In Section 2.2, we introduce the fundamentals of text mining and describe the process of training a large amount of text data to classify an unseen test data set.

2.1 Financial background

In this section, we describe financial foundations that we will utilize when designing a trading system and when evaluating this system financially. In Section 2.1.1, we explain fundamentals of trading at stock markets. In Section 2.1.2, we discuss the role of news articles and their potential influence on the stock prices.

2.1.1 Trading fundamentals

The characteristics of markets can determine whether short term stock trading decisions are profitable. In this section we explain how securities are typically traded on the market. We then give an overview about the costs of trading to create realistic assumptions on the profits that can be earned by a trading system.

In principle, most basics described in this section hold not only for stocks, but for any kind of financial securities such as treasury notes, bonds and federal agency securities. However, since this thesis aims to predict stock prices, we limit our description to stocks. Furthermore, since we are interested in short-term price movements, we do not take dividends into account, which is cash returned by a company to its stakeholders typically quarterly as part of its payout policy (Brealey et al., 2011).

Trading mechanics

Whenever an investor wishes to buy or sell a stock, he or she must file a market order specifying the following: The stock's issuer and company name, the order purpose (buy or sell), the order size (number of shares) and the *order type*. Individual investors typically use intermediates, so called *brokers*, to place the order on the market.

The following four types of orders can be distinguished (Elton et al., 2011a):

- *Market order*: An order to buy or sell at the best price currently possible. For instance, suppose there are investors on the market willing to pay 30$ for a single Microsoft stock and at the same time there are investors willing to sell Microsoft for 30.50$. Then Microsoft would be quoted at 30$ *bid* and 30.50$ *ask*. If an investor is placing a market order to buy Microsoft, he or she would pay 30.50$, which is currently the best price available. Similarly, if an investor is placing a market order to sell Microsoft, the order price would be 30$.

- *Limit order*: An order to buy (sell) at a fixed maximum (minimum) price. For instance, assume an investor places a limit order on the market to sell Microsoft stocks for the minimum price of 31$. Then the order is executed as soon as another investor is willing to pay 31$ or above for the Microsoft stock. Since this might not happen at all, limit orders are not always executed. Therefore, the investor is required to specify a period of time when placing a limit order. If the order has not been executed within this time, the limit order is canceled automatically. This type of limit orders is called *standing limit order* (Harris, 2003).

- *Short sale*: An order to sell a stock the investor does not own and buy it back at a later point of time. For instance, an investor borrows a Microsoft stock for 30$ from a brokerage firm and sells the stock on the market. Later he or she buys the stock back and returns it to the brokerage. If the stock trades at this time at 20$, the investor has earned a profit of 10$. The most common reason for investors to short-sell a stock is to expect the stock price to decline in the future.

- *Stop order*: An order that will be converted to a market order when the stock price reaches a fixed limit. Stop orders are usually used to lock in a profit. For instance, an investor owns a Microsoft stock bought at 20$ that now deals at 35$. In order to save a part of the earned profit, the investor could place a stop sell (or *stop loss*) order at 30$. As soon as the price drops to 30$, the order becomes a market sell order.

In practice, stocks are mainly traded on *stock markets* such as the New York stock exchange (NYSE) in the United States or the Frankfurt stock exchange (Frankfurter Wertpapierbörse, FWB) in Germany.

Trading costs

Trading costs are an important aspect of trading markets, since they determine how large the expected mispricing of a stock has to be before an investor will place an order. There are three main sources of trading costs: Direct costs, costs caused by the bid-ask-spread (Elton et al., 2011a) and price impact costs (Edelen et al., 2007).[1]

The direct costs include commissions, fees and taxes. Commissions are paid to market brokers in order to execute the trade. They are known in advance of the trade, but vary from broker to broker. Similarly, commissions charged by a single broker may vary depending on the difficulty of the trade to be executed. Commissions are calculated based on the total price of stock traded. Fees are paid by the brokers and they typically pass the bills to their customers, the investors. Fees include ticket charges for floor brokers, clearing and settlement costs, exchange fees, and in the U.S. SEC (Securities and Exchange Commission) transaction fees, which recover governmental costs for supervising the markets (SEC, 2007). Taxes are charged based on realized earnings. They are known in advance, but depend on the total transaction size (Kissell, 2006).

The costs caused by the bid-ask-spread were first analyzed by Demsetz (1968) and can be understood by considering a continuously trading market on a stock exchange. All standing limit orders placed by investors are filed in the so called *limit order book*, usually maintained by the exchange. An example of a limit order book is illustrated in Figure 2.1. The best bid price for a Deutsche Bank stock is 23.795 € (1,494 shares), and the best ask price is 23.805 € (313 shares). Without any new orders, no trade would be executed. The difference between best bid and best ask price is called *bid-ask spread* and would be 23.805 €-23.795 € = 0.01 €. The best price estimate of the current stock value is the average of best bid and best ask price (23.795 €+23.805 €)/2 = 23.800 €. If an investor is placing now a new market buy order, he or she would have to pay the best price of 23.805 € and thus paying an extra amount of 0.005 €, which is half of the bid-ask-spread (Harris, 2003). However, there are more sophisticated methods to estimate the costs of the bid-ask-spread by taking market realities into account, such as the possibility for traders to hide the order size (e.g. Bessembinder & Venkataraman, 2010). The

[1]Another cost factor is the cost of acquiring information to support the investor's trading decision. Since this is hoped to be achieved in this thesis, the costs are neglected here.

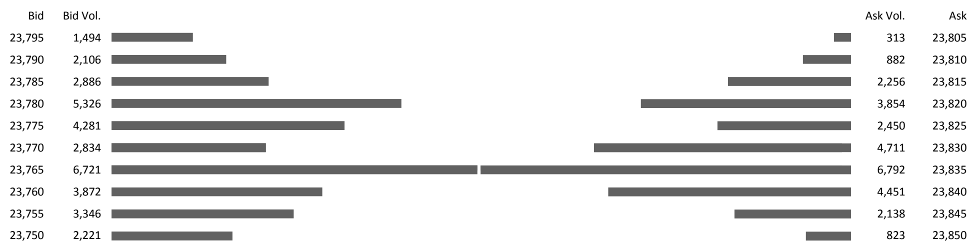


Figure 2.1: Limit order book for Deutsche Bank AG at 14:09:53 MEZ on 23rd July 2012 on Xetra, the FWB trading system (Deutsche Börse, 2012)

more illiquid the stock is traded, the higher the costs caused by the bid-ask-spread are likely to be (Elton et al., 2011a).

The price impact (or market impact) costs are caused by either liquidity demands of an investor or the information content of an order. Liquidity demands require paying a premium for attracting new buyers or sellers. In the example above (see Figure 2.1), an investor might want to buy 2,000 shares of Deutsche Bank stocks. However, only 313 shares are available for the best ask price of 23.805 €. The investor would have to pay a premium of 0.005 € each share for additional 882 shares and a premium of 0.010 € each share for the remaining 805 shares. The information content of an order is a signal to the market that the security traded is likely to be mispriced. For instance, assume that an investor places an order to buy 250,000 shares of Deutsche Bank stock. Once this information is released to the market, other investors currently owning the stock might conclude that the stock is undervalued and adjust their prices upwards (Kissell, 2006).

The problem of correctly estimating trading costs is a challenging one and remains widely discussed (e.g. Odean, 1999; Snell & Tonks, 2003). Trading costs are negatively correlated with the stock's market capitalization (Edelen et al., 2007), since large stocks tend to be more liquid and trade with more volume than middle-size or small stocks, which results in less bid-ask-spread and price impact costs. In the following, large cap stocks are referred to as stocks with more than 8 billion $ market capitalization (equal to 70% of the U.S. market capitalization), mid cap stocks as stocks with a 2 to 8 billion $ market capitalization (20%) and small cap stocks as stocks with less than 2 billion $ market capitalization (10%) (Morningstar, 2011). Edelen et al. (2007) analyze trading costs for mutual funds on the U.S. market and report average one-way trading costs of 76 (bps, 1 basis point = $1/100$ of a percentage point) or 152 bps per round trip.[2] For large cap stocks, av-

[2]One way trading costs are costs for a single trade, whereas round trip trading costs are costs for buying and selling an equity.

erage one-way trading costs are 11 bps for trading commissions, 7 bps for bid-ask-spread costs and 26 bps for price impact costs, which sums up to around 45 bps total trading costs. The trading costs for mid-caps (84 bps) and small-caps (146 bps) are substantially higher. Investment Technology Group (ITG), a multinational research broker firm, quarterly estimates the average trading costs. In the U.S. (first quarter 2012), they estimate total one-way trading costs for large cap stocks to be 34.4 bps, for mid cap stocks 67.7 bps and for small cap stocks 117.3 bps. They report a significant decline of trading costs compared to the first quarter 2009, when large cap stocks trading costs were 73.9 bps (ITG, 2012). This tendency is consistent with the findings of French (2008), who reports a 92% reduction of trading costs from 1980 to 2006 in the U.S. market. It can be explained with factors such as the development of electronic trading networks and regulations by SEC to increase market transparency and liquidity (French, 2008). Li et al. (2011) assume average round trip costs of only 30 bps for their stock price forecasting system.

2.1.2 Trading on financial news

In this section, we give an overview about financial news as a main source of information influencing the stock price. We elaborate on the efficient market hypothesis that deals with the question whether and how fast new information is reflected in the stock price. We discuss the event study methodology, which is a widely used method to empirically test the efficient market hypothesis.

Financial news

Financial news can be divided into two different types: regulated news and unregulated news (Munz, 2011). In the U.S., the term *regulated news* is defined by the Securities and Exchange Commission (SEC). Its Regulation Full Disclosure (FC) forces since 2000 all publicly traded U.S. companies to simultaneously disclose “material” and “nonpublic” information to all investors (SEC, 2000). A piece of information is “nonpublic” if it has not been made public to all investors yet. The SEC defines “material” information as information that is highly likely for reasonable investors to be considered important for trading decisions. The following examples for material information are given, which are supposed to be reviewed to decide whether they are material (SEC, 2000, sec. II B.2):

1. “earnings information”
2. “mergers, acquisitions, tender offers, joint ventures, or changes in assets”
3. “new products or discoveries, or developments regarding customers or suppliers (e.g., the acquisition or loss of a contract)”

4. “changes in control or in management”
5. “change in auditors or auditor notification that the issuer may no longer rely on an auditor’s audit report”
6. “events regarding the issuer’s securities – e.g., defaults on senior securities, calls of securities for redemption, repurchase plans, stock splits or changes in dividends, changes to the rights of security holders, public or private sales of additional securities”
7. “bankruptcies or receiverships”

The information must be made publicly available on one or more of the following channels (SEC, 2000, sec. II B.24):

- *Form-8K Disclosures*, which are report templates specifically filed for the purpose of disclosure
- Press releases disseminated by a newswire service
- Conference calls received by telephonic or electronic means
- The company’s website. This disclosure channel was added by SEC (2008).

Until recently, the NYSE demanded companies to use press releases to fulfill their disclosure obligation. In 2009, the rules were changed and allow companies to use any of the channels specified by SEC (2000, sec. II B.24). However, the SEC assumes that many companies continue to use press releases in the future (SEC, 2009). Press releases are typically distributed by newswire services. The services PR Newswire and Business Wire are market leaders and together share 58.6% of the newswire market (UBM, 2011). However, their competitor newswire services such as Reuters, Dow Jones or Bloomberg are so-called *Editorial Newswires* that manually edit incoming press releases, which may be time consuming and reduces the likelihood of making profits by trading on these news. Since PR Newswire and Business Wire publish news identical with the texts originally submitted their market share for publishing press releases enforced by the Regulation Full Disclosure is much higher (Mittermayer, 2006).

The term *unregulated news* is referred to as news not considered as material information, such as analyst opinion or rumors, which can be distributed by various channels like traditional news reporting, blogs or social media. This makes unregulated news a potential source of noise caused by irrelevant information or editorial errors (Munz, 2011).

Efficient market hypothesis

The efficient market hypothesis (EMH) was originally proposed by Fama (1965) and states that security prices always fully reflect all available information in efficient capital markets. In other words, it is impossible to earn profits by trading based on any available information (Jensen, 1978). However, this strong hypothesis would mean that investors need incentives to trade until the prices in fact reflect all information. Thus, the costs of information acquisition and transaction costs would need to be zero, which is clearly not the case (Grossman & Stiglitz, 1980). A less restricted definition of EMH seems to be more realistic, which states that security prices only reflect all information only until marginal trading benefits (profits) exceed the marginal trading costs (Fama, 1991).

Tests of EMH typically deal with the question *how fast* the information is reflected in security prices. There are three different categories of EMH tests, each considering a different subset of information (Elton et al., 2011b; Fama, 1970):

- *weak form* tests: Is all information contained in historical prices fully reflected in the current price? These tests include examining seasonal patterns such as high returns on January or returns predicted from past data.
- *semi-strong* tests: Is all publicly available information fully reflected in the current price? These tests deal with the question whether investors can earn excess profits based on public news and announcements that change price expectations.
- *strong form* tests: Is all information available, whether public or private, fully reflected in the correct prices? These tests examine whether any type of investors can make excess profits, even if they possess insider information.

In this thesis, the semi-strong form of EMH is particularly interesting, since a trading system earning excess returns based on analyzing financial news will need to test this semi-strong form. A method widely used in financial research to test semi-strong form of EMH is called *event study* and is described in following section.

Intraday event study methodology

Event studies have the purpose to test whether markets are efficient and in particular, how fast new information is incorporated in the price (Elton et al., 2011b). For instance, an event such as a company merger or an earnings announcement might be reflected in the stock price within a few minutes, days or even weeks. Since we aim to forecast short term stock price movements in this thesis, we focus on event studies examining price reactions within less than a day, so-called *intraday event studies*.

Event studies are typically structured as follows (MacKinlay, 1997). First, an event of interest and an *event window* are defined. The event window represents the period over which the stock prices of the companies involved in this event will be analyzed. Second, selection criteria are introduced deciding which companies are included in the event study. For instance, one might include only companies contained in the S&P 500 index. Third, a measurement of the abnormal return $AR_{i\tau}$ is determined, which is for the stock of company i and the event date τ defined as

$$AR_{i\tau} = R_{i\tau} - E_{i\tau}.$$

$R_{i\tau}$ is the actual ex post return over the event window and $E_{i\tau}$ is the normal return, meaning the returns that are expected using a *normal return model*. Next, an *estimation window* is defined, which is the time period used to estimate the normal price given the normal return model. Usually, the estimation window is chosen to be a time period prior to the event window. Based on the estimated normal returns, the abnormal returns can be calculated to gain insides about the effects caused by the event under study.

The following main issues need to be taken account when dealing with intraday data (Mucklow, 1994):

- Calculating returns: The most common ways of calculating the returns of any given stock at time t are the *proportional return* $P_t/P_{t-1}-1$ and the *logarithmic return* $\ln(P_t/P_{t-1})$. However, using the proportional return introduces an upward bias of stock returns caused by the bid-ask-spread. Mucklow (1991) finds that using the logarithmic return eliminates this bias.

- Noncontinuous trading: There might be no stock trade at all in a given time interval. Assume a stock trading at time t at the price P_t and stops trading until the time $t+n, n \geq 2$, when it trades at price P_{t+n}. The return defined by P_t and P_{t+n} can be described by three different ways. First, the equilibrium price is assumed to remain stable during the non-trading period and thus the return is assumed to be zero (realization method). Second, the return is allocated evenly throughout the non-trading period (quasiaccural method). Third, the non-trading period is treated as undefined and excluded from the sample (consecutive returns method).

- Estimating the normal return: There are three ways of estimating the normal return in order to determine the abnormal return of a stock. First, the return is assumed to be zero (raw returns model). Second, the return is assumed to be equal to the mean return at the same time interval and stock at other days in the estimation period (mean-adjusted returns model). Third, the return is adjusted to market movements and risk using methods such as the capital asset pricing model (CAPM) proposed by Lintner (1965); Sharpe (1964) (market adjusted models). Mucklow (1994) finds that for time periods less than 60 minutes, the raw returns model is sufficient for well-specified statistical tests.

Mittermayer (2006) compiled the results of various intraday event studies within two decades. He concluded that stock prices start to adjust a few seconds after a news arrival and end to adjust usually 5-30 minutes later.

2.2 Text mining

Text mining can be defined as an application of data mining. Data mining is referred to as the process of discovering meaningful patterns in usually well structured data with the goal of gaining a (typically economic) advantage (Terada & Tokunaga, 2003; Witten et al., 2011, Chapter 1.1). However, in the case of text mining, the patterns are extracted from unstructured textual data in document collections (Feldman & Sanger, 2006). Sebastiani (2002) states that these document collections have large quantities. Witten et al. (2004) defines a wide range of text mining activities that particularly include document retrieval, text classification, language identification and extracting entities such as names and dates.

In this section, we describe the activities necessary to extract patterns from financial news that might be responsible for significant stock price movements. First, we explain the different methods to represent a text document by identifying features that contain essential information about a text and how to transform the document into a compact feature vector (Section 2.2.1). We then discuss the process of actually extracting the features from the text (Section 2.2.2), calculating the weights of the feature vectors (Section 2.2.3) and successively reducing the overall number of features to increase performance (Section 2.2.4). In Section 2.2.5, we compare different classifiers that transform the information contained in the extracted features into a model that is able to automatically classify unseen text documents. Last, we review suitable methods to evaluate the performance of the created model (Section 2.2.6).

2.2.1 Feature vector representation

The process of scanning text documents and extract useful information is called *information extraction* (IE) (Hobbs & Riloff, 2010). Many IE techniques have recently been adapted to the research area of text classification and are used to extract *features* that contribute to the semantics of the text document (Sebastiani, 2002). The following main types of features are particularly important in this thesis:

- *n-grams*
- *Named entities*
- *Part of speech (POS) tags*
- *Sentiment*

n-grams are sequences of n words in a row. The simplest form of n-grams are unigrams ($n = 1$), also referred to as bag-of-words. An n-gram with $n = 2$ is referred to as bigram. The word "price" is an example for a unigram, the phrase "company acquisition" is an example for a bigram.

Named entities are proper names, i.e. names of particular things or classes (Sekine & Nobata, 2003). Named entities can be categorized into the following widely accepted classes proposed at the Seventh Message Understanding Conference (MUC-7) (Chinchor, 1998): unique identifiers of entities (organizations, persons and locations), times (dates and times) and quantities (monetary values and percentages). The process of identifying named entities in a text document is referred to as *named entity recognition and classification* (NERC). NERC is typically realized using either supervised learning techniques or handcrafted rule-based algorithms. Supervised learning techniques are based on the idea of automatically inferring rules or sequencing labeling algorithms based on a large amount of prelabeled training examples. Supervised learning techniques do not require system developers to have prior expert knowledge in linguistics and have recently become increasingly popular (e.g. Asahara & Matsumoto, 2003; Borthwick et al., 1998). However, rule-based approaches are superior when only a few training examples are available (Nadeau & Sekine, 2007).

A widely used and freely available system performing NERC is the language engineering framework GATE (Cunningham et al., 2011) along with the information extraction system ANNIE (Cunningham et al., 2002). ANNIE consists of different processing resources that can be successively used to extract named entities. The processing resource *Gazetteer* contains plain text lists containing widely known examples for named entities such as "Europe" or "New Taiwan dollar" categorized by named entity types such as persons or locations. However, using only a gazetteer is often not enough, since there might be words belonging to two or more different categories. For instance, "Washington" could be a surname or a state, "Philip Morris" could be a person or a company (Mikheev et al., 1999).[3] To address this problem, the ANNIE processing resources *part of speech (POS) Tagger*

[3]For other challenges of NERC using gazetteers see Nadeau et al. (2006)

(Hepple, 2000) and *Semantic Tagger* can be used. The POS Tagger creates annotations for each word or symbol, each representing one of 36 grammatical categories also known as the *Penn Treebank tag set* (Marcus et al., 1993). The complete tag set can be found in Appendix A.C. Subsequently, the *Semantic Tagger* can be used to produce the final named entity annotations. This is done by applying rules written in the JAPE (Java Annotations Pattern Engine) language (Cunningham et al., 2000). For instance, the following rule could be designed: If the words "lives in" are followed by a word that was annotated as NNP (singular proper noun) by POS Tagger, annotate this noun as "Location" (Cunningham et al., 2011). Marrero et al. (2009) found that ANNIE performs well with respect to NERC compared to similar systems.

As mentioned above, *part of speech (POS) tags* are word annotations representing different grammatical categories. Words of a particular subset of categories such as verbs or nouns might contribute more to the semantic of the text than others. Gonçalves & Quaresma (2005) report that choosing such a word subset as features leads to a learner performing equally well as a learner using all words. Similarly, the POS tags themselves can serve as features rather than the annotated words. This method has shown limited success in the past (Moschitti & Basili, 2004), but has rarely been applied in the financial domain yet.

The *sentiment* of a text document represents the overall opinion towards its subject (Pang et al., 2002); for instance, whether a financial article is good or bad news for the according company. Davis et al. (2012) report that the language sentiment (optimistic or pessimistic tone) in earnings press releases reflects the future performance of the company at the market. Henry (2008) finds that investors' trading decisions are influenced by the tone in earnings press releases and reports higher abnormal returns after press releases using a positive tone. The sentiment can be automatically extracted using either a text classification or a lexicon-based approach (Taboada et al., 2011). The text classification approach includes labeling text examples with their sentiment and trains a classifier to successively label unseen text examples (Pang et al., 2002). The lexicon-based approach includes calculating the sentiment of a document by using the sentiment of the words or phrases contained in this document (Turney, 2002). In the first step, a dictionary containing words or word phrases along with their sentiment values is created automatically (e.g. Hatzivassiloglou & McKeown, 1997; Turney, 2002) or manually (e.g. Taboada et al., 2011; Tong, 2001). Then all words or phrases in the dictionary can be extracted from a text and their according sentiment values can be aggregated, resulting in a single sentiment score for each text (Taboada et al., 2011). In this thesis, we will use the manually created sentiment list provided by Taboada et al. (2011).

With the exception of the sentiment feature, all features described consist of words or word phrases (also referred to as *terms*). However, classifier algorithms are not able to handle textual data. Thus, text documents need to be converted into a numerical vector of features, uniformly for training and test data. Let T be a set of features that occur at least once in the whole document corpus. For each document d_j every feature in T is given a weight $0 \leq w_{ij} \leq 1$, resulting in a feature vector $\vec{d_j} = \langle w_{1j}, \ldots, w_{|T|j} \rangle$ (Sebastiani, 2002).

In the following, we will describe the process of text preprocessing (Section 2.2.2) and calculating the weights of the feature vector (Section 2.2.3) in more detail.

2.2.2 Text preprocessing

Tokenization

The first step of text preprocessing is called tokenization, which is the process of dividing the text into *tokens*. A token is a useful semantic unit consisting of a character sequence in a document. This character sequence might be a word, a number, a symbol or a punctuation mark. Tokenization can simply be realized by using white spaces as word separators and cutting off all numbers and symbols (Manning et al., 2008, Chapter 2). However, different problems become obvious in practice (Manning & Schütze, 1999):[4]

- A period following a word can cause ambiguities: The period can either mark the end of a sentence or an abbreviation. For instance, *Washington* is often referred to as *Wash.*, which can be confused with the verb *wash*. To make matters worse, abbreviations such as *etc.* often occur at the end of the sentence. Here, one period serves both functions simultaneously.

- Contractions such as *I'll* and *isn't* present the problem whether or not and where to split them into single words. Phrases like *dog's* may stand for *dog is*, *dog has* or can be considered as possessive case of *dog*. However, there are valid examples that attach the *'s* to the last word of a noun phrase, e.g. *the house I rented yesterdays' garden*. Therefore, it is not clear how to treat these phrases.

- Hyphens that occur between words often make it difficult to determine how to tokenize these words. For instance, *e-mail* might clearly be considered as one word, whereas word groups such as *text-based* or *90-cent-an-hour raise*

[4]Manning et al. (2008, Chapter 2) emphasizes the difference of tokenization issues in different languages. However, since this thesis deals with English speaking news, we focus on this language.

should be separated into single words. Hyphens are also commonly used at the end of a line to divide words into two parts in order to improve justification of text. These line-breaking hyphens can be confused with hyphens used naturally.

To address these problems, it is recommended to customize the tokenizer (Weiss et al., 2005) taking the domain into account.

Stop Words

Very common words such as *is* and *the* are not very helpful for determining the correct document class. Eliminating these words can highly increase the runtime of classification by reducing the number of tokens up to 40% (Navarro & Ziviani, 2011). They are called *stop words* and are typically stored in *stop lists*, which are often hand-crafted using domain specific knowledge. An example used in this thesis can be found in Appendix B.A.

Stemming and lemmatization

Stemming and lemmatization both have the goal to reduce the extracted words to their common base form, since it often seems to be useful to match words with related ones. For instance, the words *is*, *are*, *am* and *be* would be transformed into *be*, the words *cat*, *cat's*, *cats* and *cats'* would be transformed into *cat*. However, stemming and lemmatization have subtle differences. Stemming refers to the heuristic of reducing a word to its stem by cutting off the word's end in order to achieve, in most cases, acceptable results. In contrast, lemmatization is the process of reducing a word to its *lemma*, which is the canonical form of the word. This is done by figuring out and removing inflectional endings using morphological analysis. For instance, a stemmer would most likely transform the word *saw* into *s*, whereas a lemmatizer would try to identify whether the token is a verb or a noun in order to decide whether to output *saw* or *see* (Manning et al., 2008, Chapter 2).

One of the most well-known stemming algorithms is the *Porter stemmer* developed by Porter (1980). It is based on the idea of removing the longest possible affixes of words. This is done by applying hand-crafted rules. The sequence $C?(VC)mV?$ represents a word, where C is a list of consonants and V is a list of vocals. Then $C?$ is an optional consonant sequence in the beginning and $V?$ an optional vocal sequence in the end of the word. The sequences VC are repeated m times in between. For instance, the words *free* and *why* correspond to $m = 0$, the words *prologue* and *compute* to $m = 2$. In the next step, rules satisfying certain conditions are defined: The temporal stemming rule $(m > 0)EED \rightarrow EE$

transforms *agreed* to *agree* and leaves *feed* unchanged since it satisfies $m = 0$ (Weissmann, 2004).

Stemming has the disadvantage of causing ambiguities between words. For instance, the Porter stemmer would reduce the words *operational*, *operative* and *operating* to *oper*. This would cause the phrases *operational research*, *operating system* and *operative dentistry* to lose part of their meaning. Lemmatization can partly solve this problem by removing only inflectional endings. However, *operating system* would be transformed to *operate system*, which is still a bad match (Manning et al., 2008, Chapter 2). Krovetz (1993) and Hull (1998) report small but consistent improvements in retrieval effectiveness achieved through stemming (Xu & Croft, 1998). Stemming has been successfully used in practice, reducing the number of terms by about 40% (Witten et al., 1999a, p. 147).

2.2.3 Feature weighting

The feature vector to used to represent a document described in Section 2.2.1 contains a weight for each feature. The most commonly used approaches to calculate these weights, are *binary weighting* and *TF-IDF weighting*.

Binary weighting simply assigns a feature occurring in a document the weight 1 and the weight 0 otherwise. TF-IDF weighting (Salton & Buckley, 1988) relies on two different measurements, the *document frequency* (DF) and the *term frequency* (TF). DF_i refers to the number of documents that contain a term i, $\text{TF}_{i,j}$ refers to the number of occurrences of term i in the document j (Baeza-Yates & Ribeiro-Neto, 2011). Let N be the number of documents in the whole corpus. Then DF_i/N is the document frequency relative to the corpus. Thus, $\text{IDF}_i = \log N/\text{DF}_i$ can be referred to as the *inverted document frequency* of term i, which decreases with the number of documents containing term i. The TF-IDF weighting scheme is defined as

$$w_{i,j} = \text{TF}_{i,j} \times \text{IDF}_i,$$

where $w_{i,j}$ is the weight associated with term i and document j. TF-IDF exists in different variants as discussed in Witten et al. (1999a). These introduce extensions such as varying parameters to increase or reduce the influence of $\text{TF}_{i,j}$ or using the log function to increase the weight (Baeza-Yates & Ribeiro-Neto, 2011).

2.2.4 Dimensionality reduction

Before feeding the classifier with the extracted features, it might be useful to select the most important ones. This process is referred to as *dimensionality reduction* and has the following two main reasons (Manning et al., 2008, Chapter 13): First, it reduces the number of so-called *noise features*. These are features that contribute to high misclassification error. For instance, a rare word might be not important for a document, but still happens to be present in this document. Then a classifier might draw the wrong conclusion of a strong relationship between this word and the document. This problem is called *overfitting* and will be explained in detail in Section 2.2.5. Second, since large documents might contain thousands of features, reducing their number will reduce the time needed for the successive classifier training. In the following, the most important feature selection methods will be reviewed.[5]

Frequency based methods

A popular approach to reduce feature dimension is the TF-IDF weighting scheme described in Section 2.2.3. Once the terms are TF-IDF weighted, the ones with the weights $w_{i,j}$ laying above a predefined threshold can be selected as features (Gonçalves, 2011). Another option is to use document frequency (DF) on its own as a ranking to reduce features (Luhn, 1957) and is based on the idea that rare terms are either noise features (not helpful for the classifier training) or do not influence the global performance (Yang & Pedersen, 1997). Those features can therefore be eliminated to reduce training time. Very common words considered to be noise either and are assumed to be filtered out in advance by the use of a stop list (see Section 2.2.2).

Term strength (TS)

Term strength, sometimes also referred to as *word strength*, ranks terms based the probability of a term t being existent in a document that is similar to any document containing t. More formally this can be written as

$$s(t) = Pr(t \text{ contained in } y | t \text{ contained in } x),$$

where x and y is an arbitrary pair of *similar documents* (x, y). Two documents are considered to be similar, if their cosine similarity is above a certain threshold.[6] Then the term strength of t can be approximated by dividing the number of document pairs (x, y) in which t occurs in both x and y by the number of documents where t occurs in x (Wilbur & Sirotkin, 1992; Yang & Wilbur, 1996).

[5]For further methods see Sebastiani (2002, p.14 ff.)

[6]A detailed explanation of cosine similarity can be found e.g. in Salton & McGill (1986, p. 201 ff.)

Information gain (IG)

Information gain (Quinlan, 1986) measures the degree of information available for the category prediction based on the presence of absence of a term. It can be written as

$$\begin{aligned}\text{Gain}(t) = &-\sum_{i=1}^{m} Pr(c_i) log_2 Pr(c_i)\\ &+Pr(t)\sum_{i=1}^{m} Pr(c_i|t) log_2 Pr(c_i|t)\\ &+Pr(\bar{t})\sum_{i=1}^{m} Pr(c_i|\bar{t}) log_2 Pr(c_i|\bar{t}).\end{aligned}$$

In this functional, $\{c_i\}_{i=1}^{m}$ is the set of categories and t is a term (Yang & Pedersen, 1997). The part $-\sum_{i=1}^{m} Pr(c_i)log_2Pr(c_i)$ is also referred to as *entropy* $E(t)$ of a term t (Manning et al., 2008, Chapter 13). All terms below a predefined threshold can be discarded (Yang & Pedersen, 1997).

Mutual information (MI)

The mutual information criterion measures how much influence the presence or absence of a term has in correctly deciding the class of a document (Manning et al., 2008, Chapter 13). Suppose a term t and a class c and let $Pr(A)$ be the a priori probability of A. Then is the mutual information defined as

$$I(t, c) = \log \frac{Pr(t \wedge c)}{Pr(t) \times Pr(c)}. \quad (2.1)$$

Consider the two way contingency table (Table 2.1). A is the number of co-occurrences of t and c and D is the number of times both t and c are missing. B is the number of occurrences of t without c and C is the number of times c occurs without t. Further let N be the number of all documents in the corpus. Then

	c	$\neg c$
t	A	B
$\neg t$	C	D

Table 2.1: Contingency table

Equation 2.1 can be estimated as

$$I(t,c) \approx \log \frac{A \times N}{(A+C)(A+B)}$$

(Yang & Pedersen, 1997). If t and c are independent, their mutual information value is zero, meaning that information about one of the variables does not tell anything about the other variable (Gonçalves, 2011). In order to rank the terms regarding their global mutual information, the following two variants can be formulated (Yang & Pedersen, 1997):

$$I_{avg}(t) = \sum_{i=1}^{m} Pr(c_i) I(t, c_i)$$

$$I_{max}(t) = \max_{i=1}^{m} I(t, c_i),$$

where m is the number of classes. Features are selected if $I_{avg}(t)$ (or alternatively $I_{avg}(t)$) is above a certain threshold. All other features are filtered out (Gonçalves, 2011).

Despite its popularity, MI has the disadvantage of being influenced by the a priori probabilities by the terms. Two terms with identical conditional probability $P_r(t|r)$ have different MI values if one term is more frequent than the other. This makes the MI scores non-comparable when having highly varying term frequencies (Yang & Pedersen, 1997).

χ^2 statistic (CHI)

χ^2 measures the independence of class and term occurrence, where independence is defined as $Pr(AB) = Pr(A)Pr(B)$ (Manning et al., 2008, Chapter 13). Again suppose the contingency table (Table 2.1) and let N be the number of all documents in the corpus, then χ^2 is defined as

$$\chi^2(t,c) = \frac{N \times (AD - CB)^2}{(A+C) \times (B+D) \times (A+B) \times (C+D)}.$$

Notice that $\chi^2(t,c)$ is zero if t and c are independent and increases the less independent t and c become. A global χ^2 term ranking can be determined using one of the alternative versions

$$\chi^2_{avg}(t) = \sum_{i=1}^{m} Pr(c_i) \chi^2(t, c_i)$$

$$\chi^2_{max}(t) = \max_{i=1}^{m} \chi^2(t, c_i)$$

(Yang & Pedersen, 1997). Similarly to MI, features are selected if $\chi^2_{avg}(t)$ (alternatively $\chi^2_{max}(t)$) is higher than a defined threshold (Gonçalves, 2011).

Compared to MI, CHI has the advantage of being comparable across terms with varying term frequencies as it is a normalized value (Yang & Pedersen, 1997). Dunning (1993) shows, however, that CHI only performs well on large corpora or on corpora restricted to the most common words. Despite their differences, MI and CHI seem to perform similarly well in most text classification applications (Manning et al., 2008, Chapter 13).

2.2.5 Classification

Generally, one can distinguish between *supervised* and *unsupervised* learning methods (Duda & Hart, 1973). In unsupervised learning, one tries to discover patterns from unlabeled examples. A common unsupervised learning task is clustering, a task of finding a finite set of clusters in order to describe the data (Fayyad et al., 1996). Supervised learning involves learning a function from prelabeled examples. A typical supervised learning task is the *classification*. In this thesis, we focus on a well-known instance of classification, the text classification (or text categorization). Assume a document domain D, an initial corpus $\Omega = \{d_1 \dots d_{|\Omega|}\} \subset D$ of documents, and a set of labels (classes) $C = \{c_1 \dots c_{|C|}\}$. Let $\breve{\Phi} : D \times C \rightarrow \{T, F\}$ be a function that assigns either the value T or F to each pair $\langle d_j, c_i \rangle \in D \times C$. T means the document d_j belongs to the class c_i, F means d_j does not belong to c_i. The function $\breve{\Phi}$ is unknown and describes how the documents are supposed to be classified. The task of text classification then is to create a function $\Phi : D \times C \rightarrow \{T, F\}$ in such a way that Φ and $\breve{\Phi}$ correspond as much as possible. The function Φ is also referred to as *classifier* or *model* (Sebastiani, 2002).

The process of text classification can be divided into two different phases, the *learning phase* and the *application phase*. The learning phase involves a *text classification algorithm* that learns the model from text documents with given labels using extracted features described in Section 2.2.1. In the application phase, these patterns can be used to assign labels (classes) to yet unseen documents.

Some text classification algorithms (including support vector machines) cannot naturally handle more than two classes. Although for most of these classification algorithms exist variants handling multiclass problems, these variants might be difficult to implement or not be fast enough (Witten et al., 2011). Alternatively, the multiclass problem can be transformed into different binary class problem using one of the following main approaches (Witten et al., 2011):

- *One-versus-rest* approach: For each class of the multiclass problem, all instances of the original data set are copied but the class value is changed to yes (instance has this class) or no (instance has any other class). The result is one binary class problem for each original class. After building the binary models, each test instance is classified according to the class that is predicted *yes* with the highest confidence.
- *Pairwise classification* approach: For each pair of classes in the multiclass problem, a new binary problem is built using only instances associated with one class of the pair. After building the binary models, test instances are assigned the class with the most votes.

In the following, we will present the most important text classification algorithms.

k nearest neighbors

The k nearest neighbors method is an instance-based learning method (Cover & Hart, 1967). That means, instead of building an explicit target function, all training examples are stored. When a new example arrives, it is classified according to its relationship to the existing training examples (Mitchell, 1997, Chapter 8).

k nearest neighbors (k-NN) works as follows (Sebastiani, 2002). Suppose the initial corpus of documents $\Omega = \{d_1 \ldots d_{|\Omega|}\}$, where d are documents classified under the classes $C = \{c_1, \ldots, c_{|C|}\}$. Assume a document $d_j \in c_i$. If a large proportion of documents similar to d_j are in c_i as well, a positive decision is taken, otherwise a negative. The similarity of documents can be defined as their distance to each other (Mitchell, 1997, Chapter 8). Therefore, classifying d_j using k-NN can be formulated as

$$\text{CSV}_i(d_j) = \sum_{d_z \in \text{Tr}_k(d_j)} \text{RSV}(d_j, d_z) \cdot [\![\breve{\Phi}(d_z, c_i)]\!].$$

In this functional, $\breve{\Phi} : D \times C \to \{T, F\}$ is the function describing how documents are supposed to be classified. Furthermore,

$$[\![\breve{\Phi}(d_z, c_i)]\!] = \begin{cases} 1 & \text{if } \breve{\Phi} = T \\ 0 & \text{if } \breve{\Phi} = F \end{cases}$$

holds. $\text{Tr}_k(d_j)$ depicts a set of k documents d_z maximizing the function $\text{RSV}(d_j, d_z)$, which is the semantic relatedness between d_j (training document) and d_z (test document).

The k-NN approach is illustrated for $k = 5$ in Figure 2.2 (Duda et al., 2001). The white and black points are training documents d_z classified as c_1 and c_2 re-

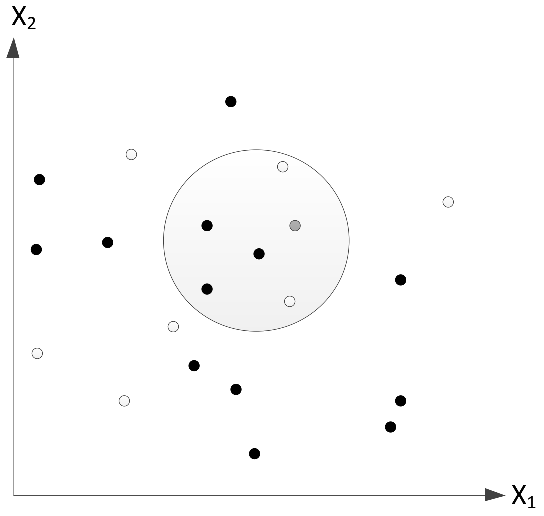


Figure 2.2: k nearest neighbors (based on Duda et al., 2001, p. 183)

spectively. The gray point represents a test document d_j. Starting with d_j, k-NN increases a spherical area until it contains $k = 5$ training documents. In this case, the majority of these documents are members of the black class c_2. Therefore, d_j would be classified into c_2.

Results of k-NN can differ when choosing different values for k. Mittermayer & Knolmayer (2006a) use $k = 10$, Yang (1994, 1999) finds that the values $30 \leq k \leq 45$ have proven to be successful. An increasing k does not significantly reduce performance (Sebastiani, 2002).

Since the data does not need to be divided linearly in a document space, the main advantage of k-NN is its good performance even if classes are highly diverse or overlapping (Gerstl et al., 2002). In the training phase, nothing needs to be calculated apart from the number of nearest neighbors, which can be done in a very short time (Mittermayer, 2006). However, the training phase is much more time consuming, since the similarity of all documents in the entire training set with the test document must be calculated one by one. This inefficiency is the major drawback of k-NN (Sebastiani, 2002).

Decision Trees

Decision trees are a very simple but yet effective approach to classify data. A decision tree represents a function that takes a vector of values as input and returns a single value as output, which represents a classification decision. The tree consists

	Features				
No.	Outlook	Temperature	Humidity	Windy	Class
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P
11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N

Table 2.2: Example training set (Quinlan, 1986)

of inner nodes representing a question asking for the value of the input feature and branches labeled with the possible feature values. The tree's leafs correspond to the classification decisions (Russell & Norvig, 2010).

For instance, suppose the training set illustrated in Table 2.2. Assume each example represents a Sunday morning and the features are Outlook = {sunny, overcast, rain}, Temperature = {cool, mild, hot}, Humidity = {high, normal}, Windy = {true, false}. A family wants to decide whether to go on a bike trip depending on the current weather. This decision is represented by the class each example is associated with (P = go, N = do not go). For the sake of simplicity, only two classes are assumed in this example, but it could be easily extended to more classes. The decision tree illustrated in Figure 2.3 correctly specifies each training example in Table 2.2. Whenever a new example arrives, it can be classified as follows. Starting at the root node, its feature value is checked and the according branch is taken. This process continues until a leaf node is reached and the example can be classified appropriately. Under the assumption of adequate features, it is always possible to construct a decision tree that specifies all examples correctly. Moreover, there often are several possible correct decision trees for one dataset (Quinlan, 1986).

A popular decision tree algorithm was proposed by Quinlan (1986) and is called ID3 (iterative dichotomiser 3). The goal of ID3 is to generate a simple

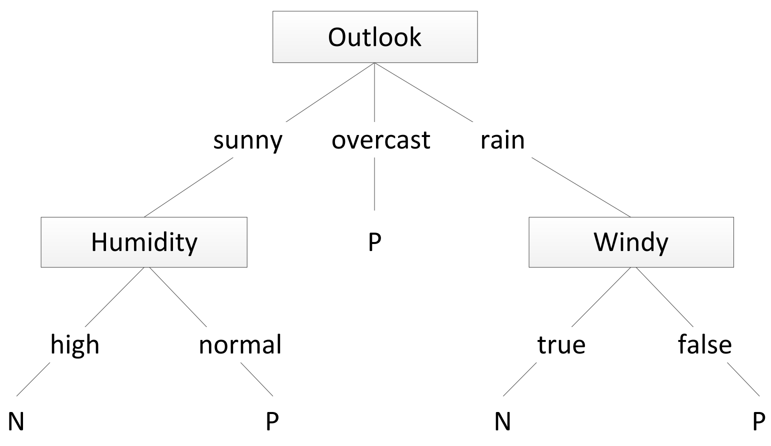


Figure 2.3: Example decision tree (Quinlan, 1986)

decision tree for a large amount of training examples and features using low computation time. ID3 works as follows. First, a feature corresponding to the root node is determined. For every value of the feature, a different branch and child node is created. This process is repeated recursively for each child node using only the instances reaching that child node. If all examples have the same classification, the algorithm stops at this part of the tree (Witten et al., 2011, Chapter 4.3).

To determine the best feature to be used as the next node, each feature is evaluated with regard of its purity. Take the example of the weather data in Table 2.2. Each of the features Outlook, Temperature, Humidity and Windy could be used as the root node of the decision tree. However, if one feature separates the examples more purely into the classes P or N than the others, then choosing this feature as node will result in a simpler decision tree. This purity can be measured by the information gain of the features, which was defined in Section 2.2.4. We rewrite the functional to

$$\text{Gain}(S, F) = E(S) - \sum_{v \in \text{Values}(F)} \frac{|S_v|}{|S|} E(S_v).$$

In this functional, F represents a term, S is the set of all examples and S_v is the set of examples with term F having the value v. More formally, $S_v = \{s \in S | F(s) = v\}$ holds. $E(S)$ is the entropy

$$E(S) = -\sum_{i=1}^{|C|} Pr(i) log_2 Pr(i),$$

where $Pr(i)$ are fractions of the examples that are classified as $c_i \in C$, where C is the set of all classes possible (Mitchell, 1997).

In the weather example, the information gain of the feature *Outlook* would be therefore calculated as follows:

$$\begin{aligned}\text{Gain}(S, \text{Outlook}) &= E(S) - (\frac{5}{14}E(S_{\text{sunny}}) + \frac{5}{14}E(S_{\text{overcast}}) + \frac{5}{14}E(S_{\text{rain}}) \\ &= 0.940 - 0.694 = 0.246,\end{aligned}$$

with

$$\begin{aligned}E(S) &= -(\frac{9}{14}log_2\frac{9}{14} + \frac{5}{14}log_2\frac{5}{14}) = 0.940 \\ E(S_{\text{sunny}}) &= -(\frac{2}{5}log_2\frac{2}{5} + \frac{3}{5}log_2\frac{3}{5}) = 0.9710 \\ E(S_{\text{overcast}}) &= -(\frac{4}{4}log_2\frac{4}{4} + \frac{0}{4}log_2\frac{0}{4}) = 0 \\ E(S_{\text{rain}}) &= -(\frac{3}{5}log_2\frac{3}{5} + \frac{2}{5}log_2\frac{2}{5}) = 0.971.\end{aligned}$$

Similarly, the other features are calculated as $\text{Gain}(S, \text{Temperature}) = 0.029$, $\text{Gain}(S, \text{Humidity}) = 0.152$ and $\text{Gain}(S, \text{Windy}) = 0.048$. As *Outlook* has the highest information gain, it is selected as the root node of the decision tree. Now the process is continued at the branch *Outlook = sunny*: The information gain of the remaining features *Temperature*, *Humidity* and *Windy* is calculated and the one with the highest information gain is taken as child node of *Outlook*. In the end, the decision tree in Figure 2.3 is generated (Mitchell, 1997; Witten et al., 2011, Chapter 4.3).

Compared to other classification approaches, the main advantage of decision trees is the transparency of its classification results. Thus, even persons not familiar with model details or domain are able to interpret the decisions taken (Brücher et al., 2002). On the other hand, decision trees suffer from a problem called *overfitting*, meaning that a complex tree is created without patterns that are supposedly meaningful. For instance, suppose an experiment is performed where a die is rolled 100 times and it is checked whether the die shows a 6. Suppose various features are reported, such as the time of the dice roll or the color of the weight. The correct decision tree for a fair dice would have just two branches, P (6) and N (no 6). However, if there are two examples where a 6 occurs with Time = afternoon and Color = red, ID3 would create another branch for this case, i.e. it overfits the data. An approach to reduce overfitting is called *pruning*. Pruning is not described in detail here, but it is based on the idea of eliminating nodes corresponding to significantly irrelevant features (Russell & Norvig, 2010). Pruning is used by C4.5, which is the successor of ID3 and was introduced by Quinlan (1993). The source code of C4.5 is published and used by various machine learning software packages.

Bayesian approaches

Bayesian approaches are based on the idea of creating a probabilistic model using the training documents. Newly arriving test documents are assigned to the class that is most likely to be correct based on this model (Gerstl et al., 2002). A very common kind of Bayesian classifiers is known as *naïve Bayes* (Michie et al., 1994) and builds on the assumption that all features are equally important and independent of one another if the class is known. Even though these assumptions are not realistic, naïve Bayes performs surprisingly well (Witten et al., 2011, Chapter 4.2). However, there are extension of naïve Bayes do not depend on these assumptions (e.g. Lam et al., 1997), which are not described in detail here.

The naïve Bayes classifier is based on Bayes' rule of conditional probability:

$$Pr(H|E) = \frac{Pr(E|H)Pr(H)}{Pr(E)},$$

where $Pr(A)$ is the probability of event A and $Pr(A|B)$ is the probability of A given event B. H denotes the hypothesis of an example being assigned to a certain class. E denotes the evidence, meaning an example consisting of a specific combination of features (Witten et al., 2011, Chapter 4.2).

For instance, take the weather data in Table 2.2. The probability $Pr(p)$ is the a priori probability that a new example classified as P, which is the number of positive examples divided through the number of all examples, in this case $9/14$. Suppose a new example with *Outlook* = *sunny*, *Temperature* = *cool*, *Humidity* = *high* and *Windy* = *true* arrives that needs to be classified. This example is the evidence E and can be divided into the four pieces of evidence E_1, E_2, E_3, E_4. $Pr(E_1|p)$ is then the probability of *Outlook* = *sunny* given the example to be positive, which is $2/9$. Since all features are assumed to be independent given the class, $Pr(E|p)$ can be calculated as follows:

$$Pr(E|p) = Pr(E_1|p) \cdot Pr(E_2|p) \cdot Pr(E_3|p) \cdot Pr(E_4|p) = \frac{2}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{3}{9}.$$

The calculated values can be plugged into Bayes' rule:

$$Pr(p|E) = \frac{Pr(E|p) \cdot Pr(p)}{Pr(E)} = \frac{0.0053}{Pr(E)}. \tag{2.2}$$

Similarly, $Pr(n|E) = 0.0206$ can be calculated. Normalizing both probabilities so that they sum up to one leads to the results $Pr_n(p|E) = 0.0053/(0.0053 + 0.0206) = 20.5\%$ and $Pr_n(n|E) = 0.0206/(0.0053 + 0.0206) = 79.5\%$. The denominator in Equation 2.2 is eliminated during the normalization step (Witten et al., 2011, Chapter 4.2).

Document classification is an important domain for the naïve Bayes approach (Witten et al., 2011, Chapter 4.2). When using the bag-of-word representation of a document, Equation 2.2 can be adapted to

$$Pr(C_i|d) = \frac{\left(\prod_{w \in d} Pr(w|C_i)\right) \cdot Pr(C_i)}{Pr(d)},$$

where d is a document, w is a word and C_i is a document class. Then the following problem becomes obvious. Suppose w does not exist in a document with class C_i. w would get the value 0, and thus the probability of classifying any document d with C_i would be zero, when d is missing w (Agrawal et al., 2000). To address this issue, $Pr(w|C_i)$ can be calculated using Lidstone's law of succession:

$$Pr(w|C_i) = \frac{n(C_i, w) + \lambda}{n(C_i) + \lambda |V|},$$

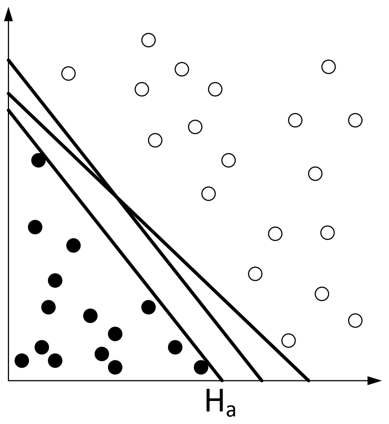
where $n(C_i, w)$ denotes the number of occurrences of w in C_i and $n(C_i) = \sum_w n(C_i, w)$ denotes the total number words in C_i. $|V|$ is the total number of words in the vocabulary. (Herrmann, 2002). Agrawal et al. (2000) find that choosing λ in a range between 0.2 and 0.01 yields the best prediction performance.

Even though the naïve Bayes classifier is comparatively simple, it often performs better or equally well as more sophisticated approaches (Witten et al., 2011, Chapter 4.2). Moreover, it scales up well when classifying new documents (Herrmann, 2002). However, due to its assumption of feature independence, the performance decreases when using datasets with high dependencies between features (Witten et al., 2011, Chapter 4.2). In the document classification domain, research shows that naïve Bayes gets worse results with large vocabularies (McCallum & Nigam, 1998).

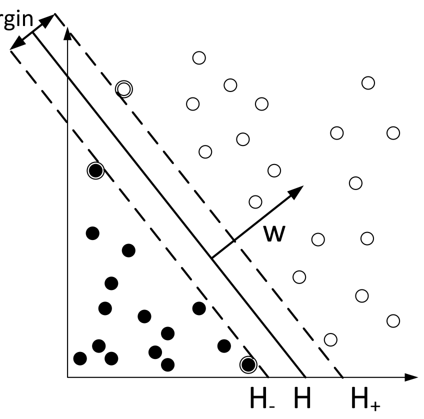
Support vector machines

Support vector machines (SVMs) were first introduced by Boser et al. (1992); Vapnik (1995). Applied to the domain of text categorization, they are reported to perform better than all methods described earlier (Joachims, 1998). The exceptional performance of SVMs has the following reasons (Joachims, 1998; Russell & Norvig, 2010, Chapter 18.9):

- SMVs use the *large margin classifier* to distinguish example points with the highest distance possible. Therefore, SVMs generalize well.
- SMVs use the so-called *kernel trick*: Data that is not linearly separable is mapped to a higher dimensional space and separated by a linear separator (*hyperplane*). After that, the data is transformed back into the original



(a) Three possible hyperplanes



(b) Optimal hyperplane H

Figure 2.4: Hyperplanes for a two class data set (based on Burges (1998); Russell & Norvig (2010, p. 745))

space. The resulting hyperplane is nonlinear in the original space. Hence, this method is a great improvement compared to methods restricted to linear representations.

- SVMs use an overfitting protection not necessarily depending on the number of features. This makes them well-suited to the problem of text mining, which typically produces a large number of highly relevant features.

SVMs are based on the following idea (Russell & Norvig, 2010). Assume that the points in Figure 2.4(a) are training examples, the black points negative ones and the white points positive ones. All three lines (hyperplanes) linearly separate both example spaces without error. However, the hyperplanes differ in terms of separation quality. Hyperplane H_a is very close to four of the positive examples. Assuming that new points are drawn from the same probability distribution as the present points, it is likely that some of those points will be placed on the other side of the line. By maximizing the distance of the hyperplane to the present points, the likelihood of wrong new points is minimized.

In the following, the concept of SVMs will be described in more detail (Burges, 1998; Vapnik, 1995). Suppose two training classes are labeled as -1 and $+1$. The training data can be written as

$$(x_1, y_1), \ldots, (y_l, y_l), x_i \in \mathbb{R}^n, y_i \in \{+1, -1\} \text{ for } i = 1, \ldots, l.$$

The training data can be separated by a hyperplane H:

$$H : w \cdot x + d = 0.$$

For $n = 2$, the hyperplane H is illustrated in Figure 2.4(b). The vectors closest to H are called *support vectors* and are circled in the figure. Suppose the shortest distance between H and the closest positive vector is d_+, the distance between H and the closest negative vector is d_-. Then $d = d_+ + d_-$ is defined as *margin* of H. The optimal hyperplane separates the set of vectors without error and minimizes the margin. This can be described as follows:

$$w \cdot x_i + b \geq 1 \quad \text{if } y_i = 1 \tag{2.3}$$

$$w \cdot x_i + b \leq -1 \quad \text{if } y_i = -1. \tag{2.4}$$

These inequalities can be combined to the following formula:

$$y_i(w \cdot x_i + b) \geq 1, i = 1, \ldots, l. \tag{2.5}$$

All points fulfilling the Condition 2.3 lay now on the hyperplane $H_+ : w \cdot x + d = 1$. The perpendicular distance from H_+ to the origin is $|1 - b| / \|w\|$ with $\|w\|$ being the Euclidean norm of w. Analogously, all points fulfilling the Condition 2.4 lay on the hyperplane $H_- : w \cdot x + d = -1$, having a perpendicular distance to the origin of $| - 1 - b| / \|w\|$. Therefore, the shortest distance between the separating hyperplane H and the positive support vector d_+ is $1/ \|w\|$ and the margin $d = d_+ + d_- = 2/ \|w\|$. The optimal hyperplane can thus be found by minimizing $\|w\|^2$ with respect to vector w and scalar b while satisfying the Condition 2.5.

If the training data are inherently noisy, as it is often the case in the domain of text mining, it may be useful to not map a linear hyperplane into a high-dimensional space. In order to reflect the reality of the noisy data, it might be preferable to separate the training example using a soft margin classifier that allows the examples to be classified into the wrong side of the separator. However, wrong examples are assigned a penalty parameter proportional to the distance necessary to move them back into the correct side (Russell & Norvig, 2010).

This method changes the optimization problem described earlier as follows (Cortes & Vapnik, 1995). Let $\xi \geq 0, i = 1, \ldots, l$ be non-negative variables. For sufficiently small $\sigma > 0$, the functional

$$F_\sigma(\xi) = \sum_{i=1}^{l} \xi_i^\sigma$$

describes the number of training errors and can be minimized under the constraints

$$w \cdot x_i + b \geq 1 - \xi \quad \text{with } i = 1, \ldots, l \tag{2.6}$$

$$\xi \geq 0 \quad \text{with } i = 1, \ldots, l. \tag{2.7}$$

For computational reasons, the case $\sigma = 1$ is considered. The idea of introducing a penalty for wrong examples can be formalized as the following minimization problem (Vapnik, 1995):

$$\Phi(w, \xi) = \frac{1}{2}w^2 + C\left(\sum_{i=1}^{l} \xi_i\right)$$

with the Constraints 2.6 and 2.7, where C is a given constant (penalty parameter). This minimization problem can be transformed into the maximization problem

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j K(x_i, x_j)$$

subject to the constraints

$$0 \leq \alpha_i \leq C, i = 1, \ldots, l$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0,$$

where $K(x_i, x_j)$ is called the *kernel function*.[7] The following kernels are most commonly used in practice (Hsu et al., 2010):

- linear: $K(x_i, x_j) = x_i^T x_j$
- polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$
- radial basis function (RBF): $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0$
- sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$,

where γ, r and d are kernel parameters. The most widely used software packages implementing the SVM approach are LIBSVM (Chang & Lin, 2011) and SVM$^{\text{light}}$ (Joachims, 1999), which perform similarly well (Zanni et al., 2006). Both packages are free of charge and provide a variety of additional features allowing advanced parameter tuning and evaluation.

2.2.6 Evaluation

The performance of text classification can be measured in terms of effectiveness (taking the right classification decisions) and efficiency (system run time). In this thesis, we will focus on the effectiveness, which is usually the more important aspect (Sebastiani, 2002).

[7] For a detailed proof, see Vapnik (1995, p. 129 ff.)

		correct label	
		yes	no
predicted label	yes	TP_i	FP_i
	no	FN_i	TN_i

Table 2.3: Confusion matrix for category c_i

Evaluation Metrics

Evaluation metrics are based on the idea of quantifying the different kinds of possible classification errors, which are illustrated in a so-called *confusion matrix* (Massy, 1965) as shown in Table 2.3. For instance, if the classifier decides an example to be in class c_i and it is in c_i in reality, this example is a *true positive*. If the classifier decides an example to be in c_i, but in reality it is not in c_i, this example is a *false positive*. *True negatives* and *false negatives* are defined analogously. The terms TP_i, FP_i, TN_i and FN_i depict the according number of examples, e.g. TP_i is the number of true positives for class c_i (Sebastiani, 2002).

The two most commonly used standard metrics are precision (π) and recall (ρ). π is the number of examples correctly classified divided by the total number of examples classified as positive. ρ is the number of examples correctly classified divided by the total number of examples that are positive in reality. Formally this can be written as

$$\pi_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}$$

$$\rho_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i}.$$

Using two different numbers for evaluating a classifier has the advantage that one number might be more important than the other in certain applications. A measurement to express the trade-off between precision and recall is called *F-measure* (F) and is defined as

$$F_i = \frac{1}{\alpha\frac{1}{\pi_i} + (1-\alpha)\frac{1}{\rho_i}} = \frac{(\beta^2+1)\pi_i\rho_i}{\beta^2\pi_i + \rho_i},$$

where $\beta^2 = (1-\alpha)/\alpha$ and $\alpha \in [0,1]$. β changes the relative importance of precision and recall in the formula (Gonçalves, 2011; Manning et al., 2008, Chapter 8). The form most frequently used in practice is the so-called *balanced F-measure* (F_1) introduced by van Rijsbergen (1979), which combines precision and recall using equal weights ($\alpha = 1/2$ or $\beta = 1$) (Yang & Liu, 1999):

$$F_{1i} = \frac{2\pi_i \rho_i}{\pi_i + \rho_i}.$$

The metrics described so far evaluated only a single class. There are two standard methods that can be used to evaluate a classifier across different classes, *macro-averaging* (M) and *micro-averaging* (μ). Macro-averaging is considered to be a per-class average, since it is calculated by first determining the values of the per-class confusion matrix and then compute the global score by averaging this per-class confusion matrix. In contrast, micro-averaging is calculated by first building a global confusion matrix with cells containing the sums of the corresponding per-class confusion matrices and then computing the global score based on this global confusion matrix (Yang, 1999). Global precision and recall can therefore be computed as

$$\pi^M = \frac{\sum_{i=1}^{|C|} \pi_i}{|C|}, \rho^M = \frac{\sum_{i=1}^{|C|} \rho_i}{|C|},$$

using macro-averaging, where $|C|$ is the total number of classes. Similarly, global precision and recall can be computed as

$$\pi^\mu = \frac{TP}{TP + FP} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|}(TP_i + FP_i)},$$

$$\rho^\mu = \frac{TP}{TP + FN} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|}(TP_i + FN_i)},$$

using micro-averaging (Sebastiani, 2002). The global version of the balanced F-measure can be calculated as

$$F_1^M = \frac{\sum_{i=1}^{|C|} F_{1i}}{|C|} \text{ and } F_1^\mu = \frac{2\pi^\mu \rho^\mu}{\pi^\mu + \rho^\mu}.$$

F_1^M gives every single class the same importance, whereas F_1^μ gives every single document the same importance (Gonçalves, 2011).

Other common metrics to measure the classifier effectiveness are *accuracy* (A) and *error* (E). Accuracy is defined as number of examples correctly classified divided by the total number of examples. Error is defined as number of examples incorrectly classified divided by the total number of examples (Gonçalves, 2011). Formally, they can be defined as

$$A_i = \frac{\text{TP}_i + \text{TN}_i}{\text{TP}_i + \text{TN}_i + \text{FP}_i + \text{FN}_i}$$

$$E_i = 1 - A_i = \frac{\text{FP}_i + \text{FN}_i}{\text{TP}_i + \text{TN}_i + \text{FP}_i + \text{FN}_i}$$

(Sebastiani, 2002).

Handling unbalanced data

Despite their simplicity, the use of the metrics accuracy and error has a major disadvantage. For classes that are very unbalanced, meaning that the number of positive examples strongly differs from the number of negative examples, a trivial classifier rejecting or accepting all examples performs very well (Cohen, 1996). For instance, suppose a binary classification problem that divides a corpus of 1,000 documents into the classes c_i and c_j. Suppose 980 documents belong to c_i (majority class) and 20 documents belong to c_j (minority class). A trivial classifier assigning all documents in the corpus the class c_i would get the high accuracy $A = 980/1000 = 98\%$. A more sophisticated classifier predicting 50% of all documents would do almost equally well ($A = (10 + 980)/1000 = 99\%$), suggesting that the classifiers are nearly equivalent, which is clearly wrong (Gonçalves, 2011).

To address this problem, there are two different main approaches. First, classification errors can be assigned different cost factors according to the importance of the class. This can be done using a *cost matrix* C, which contains the same rows and columns as the confusion matrix in Table 2.3. In the cost matrix each cell $C(i, j)$ represents the cost for classifying an example to class i when it in reality belongs to class j (Domingos, 1999). Notice that most learning schemes are able to predict not only the actual value of an instance, but also the probability of that prediction to be correct. The cost values can be combined with these probabilities and for each instance, the classifier makes a prediction by minimizing the costs (Witten et al., 2011). Second, *oversampling* and *undersampling* can be used to artificially change the distribution of the examples before training. Oversampling creates synthetic examples (often duplicates) belonging to the minority class, whereas undersampling discards examples belonging to the majority class. Undersampling has the disadvantage of deleting potentially useful examples in the training set. On the other hand, oversampling has the disadvantage of generating copies of examples and thus, increasing the risk of overfitting. In addition, the data set becomes larger, which increases the training time (Weiss et al., 2007). SMOTE, an implementation of an oversampling approach is publicly available and free of charge (Chawla et al., 2002). It addresses the problem of oversampling by creating random examples rather than copying existing ones. Despite their disadvantages, oversampling and undersampling obtained good results in the past (e.g. Chen et al., 2004; Kubat & Matwin, 1997). Weiss et al. (2007) report that cost-based approaches consistently outperform over- and undersampling when dealing with large data sets (more than 10,000 examples). For smaller data sets, neither of the approaches can be identified as clearly superior.

k-fold cross validation

k-fold cross validation (Kohavi, 1995) is a way to evaluate data statistically valid on a single data set even when only a limited amount of data is available. It is done by dividing the data set randomly into k disjoint subsets of approximately equal size. This random sampling should be done in a way that each class has about the same proportion in the training set as in the test set. Otherwise, certain classes would be overrepresented in the test set and bias the results. This process is called *stratification*. Successively, k rounds of training are performed. In each round, a different subset is used as test data and the remaining data is used for training, until in the end each subset has been used for testing exactly once. The most commonly used value for k is 10, a variant which is also called *tenfold cross validation*. Research has shown that $k = 10$ usually performs best (Witten et al., 2011, Chapter 5).

Chapter 3

Review of relevant systems

Forecasting price movements using text mining techniques has been investigated by many researchers in the past two decades. In this chapter, we discuss the systems we consider to be the most important ones for the research in this thesis. Most systems use different types of input data, support different domains and serve different purposes (some predicting price trends, others volatility). However, the systems are typically developed following the schema in Figure 3.1.

News articles and stock (or stock index) price data are first gathered and stored locally. The news articles are then preprocessed, which often includes word stemming, stop word removal and feature extraction using approaches such as bag-of-words or word phrases. In a next step, the price data are used to assess the influence of the news article on the stock price. For instance, if a price jump occurs a few minutes after a news release, the news could be assumed to influence the price positively. The news articles are given labels like UP, DOWN and STABLE accordingly. Next, the labeled news articles are used as training data and a classifier is trained. In the forecast (application) phase, the system is fed with news and automatically labels them, typically in order to support investors' trading decisions.

3.1 Description of relevant systems

Wüthrich et al. (1998)

Wüthrich et al. pioneered by developing a prototype that aims to predict movements of five major stock indices in the U.S., Asia and Europe, amongst them the Dow Jones Industrial Average, daily at 7:45 am Hong Kong time (Cho et al., 1999; Wüthrich et al., 1998). News articles published online by the Wall Street

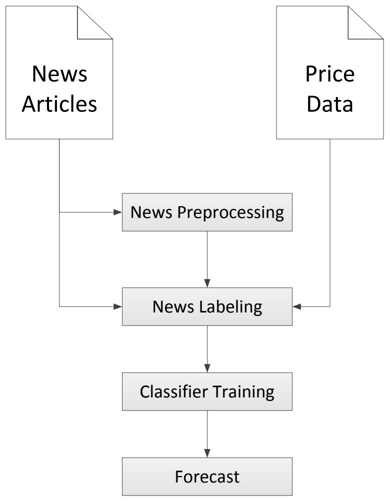


Figure 3.1: General schema of a typical trading system

Journal[1] containing stock, currency and bond market news articles are downloaded and locally stored. A thesaurus containing more than 400 individual word phrases such as ”bond strong” or ”property weak” is developed by a domain expert. In a next step, a naïve Bayes classifier is trained: The thesaurus words occurring in all news articles published before a particular day are counted and the occurrences are transformed into weights. These weights are together with the closing prices used to create probabilistic rules that are described in detail in Cho & Wüthrich (1998). For instance, the likelihood of the Nikkei 225 index going up on 6th March depends on the weights of the terms ”stock rose” on 5th March and ”property surge” on 4th March. Based on these rules, the index is forecasted to either move up ($> 0.5\%$), down ($< -0.5\%$) or remains steadily (in any other case) at the 6th March.

To measure the performance, the system was tested in 60 stock trading days in the period 6th December 1997 to 6th March 1998. The accuracy, i.e. the percentage of system predictions that are correct, is measured to be 43.6% for all indices on average. The system is also measured in terms of financial performance by buying (short-selling) an index in the morning when the respective market is predicted to move up (down). If the system predicts steady, no trading is done. All positions are liquidated when the markets close in the evening. Assuming to make 0.5% profit for each correct up or down prediction and 0.5% loss for each incorrect prediction, the average profit for each index is calculated to be 7.5% over three months. Since

[1]`http://www.wsj.com`

the system traded at only 40 days, this equals a round trip profit (profit for buying and selling a security) of $7.5\%/40 = 0.13\%$ (Mittermayer & Knolmayer, 2006b). No transaction costs are taken into account.

However, the good financial performance reported cannot be achieved in reality. The authors make the invalid assumption that closing prices are on average identical to the next day's opening prices (Mittermayer & Knolmayer, 2006b). Moreover, the authors themselves acknowledge that the system could benefit from taking numeric time series data into account (Wüthrich et al., 1998).

Fawcett & Provost (1999)

Fawcett & Provost do not try to exploit security price movements, but rather aim to issue alarms before a stock price "spike" happens. A spike is defined as a 10% price change in either direction. News stories and stock prices for approximately 6,000 companies in a three months time frame are gathered. Details such as the news source, the stock price frequency or the company choice are not specified. A story is labeled with a price spike if it appears from midnight the day before the spike until 10:30 am at the day of the spike. An adjusted version of the fraud detection system DC-1 (Fawcett & Provost, 1997) is used for extracting words and bigrams from the news and does the classifier training.

The implementation and analysis details of this system are not published.

Lavrenko et al. (1999)

The system Ænalyst developed by Lavrenko et al. tries to predict forthcoming trends in prices of single stocks (Lavrenko et al., 1999, 2000a,b). In a first step, stock price trends are identified using the piecewise linear regression technique (Pavlidis & Horowitz, 1974). Time series are transformed into segments of approximately two hours and associated with one of the labels SURGE (segment slope $\geq 75\%$), SLIGHT+ (segment slope $\geq 50\%$), plunge (segment slope $\leq -75\%$), SLIGHT- (segment slope $\leq -50\%$) and NO RECOMMENDATION (any other case). Next, financial news articles are gathered from Yahoo! Finance[2], which maintains a list of stories considered to be relevant to a particular stock symbol. Each news article is then associated with a price trend and its according label if the news article's time stamp is h hours or less before the start of the trend. Using a window of 5 to 10 hours tends to work best. The training is performed using the bag-of-words

[2]Formerly `http://biz.yahoo.com`, now `http://finance.yahoo.com/`

representation and a naïve Bayes classifier and is evaluated using the 10-fold cross validation approach.

The system's financial performance is evaluated through a market simulation from 15th October 1999 to 10th February 2000 using 38,469 news articles and prices of 127 U.S. stocks that are sampled every 10 minutes during the market hours. The training is performed between October and December 1999, whereas in the 40 days starting on 3rd January the system monitors the news and executes the following trading strategy. Whenever a news article occurs that is rated positively (negatively), the system buys (short-sells) 10,000 $ worth of stock of the according company. The stocks are liquidated after the arbitrarily chosen time of 1 hour, unless the stock can be liquidated earlier for a profit of 100 $ or more. The system is claimed to earn 280,000 $ in these 40 days, which equals 0.23% profit per round trip due to a staggering number of trades the system needs to execute. Like Wüthrich et al. the researchers do not consider transaction costs.

The impressive performance of 0.23% per round trip is however not realistic. First, compared to other systems, the amount of trades necessary is extremely high, which causes exorbitant transaction costs. Second, with 10,000 $ investment capital, the system could only perform a maximum of 325 round trips instead of the approximately 12,000 round trips needed in 40 days (Mittermayer, 2006, p. 112). The authors therefore assume unlimited funds to trade, which strongly reduces profits due to costs of borrowing. Third, the 127 U.S. stocks are picked according to criteria such as high past profits, which introduces a significant bias towards highly volatile stocks, which reduces the risk of noise trades. (Mittermayer & Knolmayer, 2006b).

The system is also criticized for choosing the time window ($h = 5$ to $h = 10$) that denotes the time for the market to absorb the news stories too large. This assumption is considered to contradict most economic theories (e.g. Adler & Adler, 1984; Blumer, 1975). Moreover, during the training phase news stories might be associated with two or more (possibly contradictory) trends, which is a dilemma (Fung et al., 2005).

Peramunetilleke & Wong (2002)

Peramunetilleke & Wong intent to forecast future foreign exchange (FX) rates with their system (Peramunetilleke & Wong, 2002). Differently from the earlier systems, only news headlines are taken into account. The researches use a rule-based approach similar to the one used by Wüthrich et al.: A thesaurus containing over 400 word sequences (e.g. "Germany, lower, interest, rate") was in advance developed by a domain expert. The number of occurrences of the sequences is counted and weighted. Probabilistic classification rules are generated from the weights and daily exchange rate closing prices. For training the data a self-made algorithm is proposed, which labels the news as DOLLAR_UP (FX $+ \geq 0.023\%$), DOLLAR_DOWN (FX $- \leq 0.023\%$) or DOLLAR_STEADY (any other case). The change of 0.023% is chosen to distribute all news about equally to all three labels.

The system is evaluated in a market simulation in the relatively short time interval 22-30 September 1993. No financial performance evaluation is done, but in the best case a 53% accuracy in predicting the FX rate is achieved. The researchers claim this to be similarly accurate as human traders.

Gidófalvi & Elkan (2003)

Gidófalvi & Elkan present a system that tries to predict future stock prices on an intraday level (Gidófalvi & Elkan, 2003). A similar earlier version of the system is discussed in Gidófalvi (2001). As opposed to the systems described above, stock prices are used on a minute-by-minute basis. Unfortunately, the source of the news stories has not been published. The researchers tackle the problem of reappearance of similar or identical news articles by eliminating news with a high similarity measured using the first 256 characters of the article. The prices are aligned to the news stories using so called *windows of influence*, meaning time intervals throughout which the news story might have an effect on the stock price. For instance, a window of influence of [-20, +30] means the time interval 20 minutes before until 30 minutes after the news occurs. In a next step the news stories are labeled UP, DOWN or NORMAL based on the price movement of the according stock. To determine the price movement, the researchers take the stock's β-value, the stock's volatility compared to the volatility of the market index, into account. Lastly, the system is trained using a 3-class naïve Bayes classifier.

A market simulation evaluates the financial performance from 26th July 2001 to 16th March 2002. Data before 1st November 2001 belong to the training set, data thereafter to the test set. The data include stock prices of the 30 DJIA companies and in total around 6,000 news stories occurring during market hours. Transaction costs are not considered. Interestingly, the interval [-20, 0] generates the highest performance, meaning it creates most profits to trade the stock 20 minutes in advance and liquidate it at the moment the news occurs. This suggests evidence for insider information. Profits are moderate (0.1% profit per round trip). It is also important to notice that these profits could not be obtained in reality, since information about future news occurrences is exploited in the simulation.

Thomas (2003)

The system developed by Thomas combines a numerical trading rule learner with an ontology based news rule learning approach (Thomas, 2003). For earlier versions of the system that use a genetic algorithm approach to forecast financial markets, see Thomas & Sycara (2000, 1999). Financial news articles are gathered from Yahoo! Finance and only the news headlines are taken into account. An ontology consisting of more than 50 categories (e.g. MERGER, LAWSUIT or PRODUCT ANNOUNCEMENT) is derived by hand. Next, Thomas manually builds classifiers that are supposed to identify the categories and take the form of logical combinations of regular expressions.[3] The classifiers are built using a news headline corpus regarding every company in the Russell 3000 index in the week 5th March to 11th March 2001. The classifiers' accuracy is then evaluated using the news headlines in the week 12th March to 18th March 2001. Precision is roughly 90% and recall roughly 70%. The classifiers are then combined with a technical rules trader developed earlier: If a news story of a certain category (e.g. earnings) occurs, no position in the according stock is taken for 15 days.

There is no clear performance simulation of the system (Mittermayer & Knolmayer, 2006b), but Thomas shows that the Sharpe Ratio (excess return per unit of deviation) can be significantly increased for a given trading strategy when leaving out massages by certain categories. These categories are ANALYST DOWNGRADE, CONFERENCE CALL, EARNINGS REPORT and EARNINGS OUTLOOK.

Schulz et al. (2003)

Schulz et al. do not try to exploit stock price movements with their system, but rather aim to identify which news articles are relevant to the stock price at all (Schulz et al., 2003; Spiliopoulou et al., 2003). This is considered to be an important issue, since market participants are proven to suffer from an information overload (Farhoomand & Drury, 2002). Schulz et al. focus on the German stock market and gather only news articles that companies have to publish by the German Securities Trading Law (WpHG). Each news article is labeled PRICE_RELEVANT or PRICE_IRRELEVANT based on the excess profit of the stock at the day the news article is published. The excess profit is estimated based on the market model (Sharpe, 1963). Training is done by the commercial Software SAS Enterprise Miner using a regression classifier.

[3] All classifiers developed are published in Thomas (2003, pp. 174-185).

Since stock prices are not forecasted, the financial performance cannot be evaluated. Instead, the average classification error is measured to be 39%. However, the more important classification error for news labeled as PRICE_RELEVANT is significantly higher (57%). One reason for this high classification error rate is considered to be the following: The type of news used by the system tends to be used by companies for advertisement purposes and is therefore biased (Kaserer & Nowak, 2001).

Fung et al. (2005)

Fung et al. developed a system to forecast intraday stock price trends. The newest version of the system (Fung et al., 2005) is summarized here; earlier versions are described in Fung et al. (2003, 2002). Fung et al. use a similar time series segmentation technique like Lavrenko et al. in their system described earlier. However, by adjusting the algorithm used, they try to avoid the problems associated with large prediction time windows and the possible alignment of one news article to more than one price trend. The news articles are gathered using the commercial trading platform Reuters 3000 Xtra[4] and labeled according to the price trends automatically as POSITIVE, NEGATIVE or NEUTRAL. The training is done using a SVM classifier.

A financial performance evaluation is done from 20th January 2003 to 20th June 2003. More than 350,000 real-time news stories and all intraday stock transactions of all Hong Kong stocks (unless stocks with "too few transaction records") are gathered. Stocks are purchased (sold short) when a news story labeled as POSITIVE (NEGATIVE) occurs and liquidated after three days. In a 5-months period, the researchers claim a accumulated profit of 18.06% and a rate of correct predictions of 61.6%. Unfortunately, the profit per round trip cannot be calculated, since the number of trades realized is not documented. The researchers consider news stories that are similar in content but have very different implications to be the main reason for prediction errors.

Phung (2005)

Phung aims to automatically extract appropriate key phrases out of financial news in order to support stock price predicting systems (Phung, 2005). This is considered to be helpful, since the earlier systems Peramunetilleke & Wong; Wüthrich

[4]http://thomsonreuters.com/products_services/financial/financial_products/a-z/3000_xtra/

et al. frequently use keywords provided by domain experts, which are not necessarily applicable to different business sectors. The news articles are gathered from the Malaysian newspaper "The Star Online"[5] and are filtered using hand-picked word queries relevant to company and sector. Subsequently, an adapted version of the automatic keyphrase extraction algorithm (KEA) proposed by Witten et al. (1999b), which uses a naïve Bayes classifier, is used to extract key word phrases from the news articles.

Since Phung only extracts financial keywords based on word queries, no financial performance evaluation is done. However, the prediction accuracy is evaluated using a test period from 1st February to 30th April 2004 and a training period from 1st May to 31st July. 90 news articles are gathered in total. In the test period, precision is 21.1%, recall is 21.4%. Phung explains this relatively low accuracy partly with the unreliability of word queries chosen initially.

Mittermayer & Knolmayer (2006a)

Mittermayer & Knolmayer developed another system trying to predict intraday stock prices using text mining approaches (Mittermayer, 2004, 2006; Mittermayer & Knolmayer, 2006a). Only U.S. press releases are taken into account, which are acquired from the newswire service PR Newswire. The following news articles are filtered out in advance: News articles associated with more than one company, news articles that occur outside the trading hours and news articles belonging to a category that is considered to be non-relevant. The news articles are automatically labeled as follows: The 15 minutes after a news release are divided into 49 moving average time windows of 90 seconds each. Using real time stock prices, the profits of each time window are calculated compared to the average price between 1 minute before and 1 minute after the news occurrence. For a maximum profit of $> 3\%$ and a maximum loss of $< 3\%$, news articles are labeled as GOOD. News articles are labeled as BAD analogously. If maximum and minimum loss both exceed a certain threshold, news articles are labeled as UNCLEAR and are excluded from the training process. In other case, the news articles are labeled as NEUTRAL. The data are in a next step trained using a bag-of-words approach and different classifiers. The SVM classifier with polynomial kernel performs best. The researchers also use a handcrafted thesaurus containing words and word phrases assumed to drive stock prices. Features in this thesaurus are forced into the final set of features, which partially overrides the results of the bag-of-words approach. The thesaurus is made publicly available in (Mittermayer, 2006, pp. 240-242).

[5] `http://biz.thestar.com.my/`

Training this done from 1st April to 31st December 2002 with all news published by PRNewswire and intraday prices of all S&P 500 stocks in 15-second intervals. An accuracy and financial performance evaluation is done via a market simulation form 1st January to 31st December 2003. The researchers use a similar trading strategy like Lavrenko et al.. Stocks are purchased (short-sold) when news articles are labeled as GOOD (BAD). All positions are liquidated after 15 minutes, with the exception of stocks gaining $> 0.5\%$ or losing $< 2\%$. Those are immediately liquidated when hitting the threshold. The accuracy rate is 82%, which is exceptionally good, taking into account that no earlier system obtained more than 50%. The average profits per round trip are in the best case 0.29%, outperforming the results reported by Lavrenko et al.. The researchers explain the good performance with the careful selection of news articles, the application of noise-reducing heuristics and their novel labeling approach.

Robertson et al. (2006)

Robertson et al. developed a system focusing on stocks at the U.S., UK and Australian market (Robertson et al., 2006, 2007a,b,c; Robertson, 2008). Similarly to Schulz et al. their goal is to separate news articles relevant to the market behavior from irrelevant ones. News articles are gathered using the commercial software Bloomberg Professional[6] and include Press Announcements, Annual Reports, Analyst Recommendations and general news from more than 200 different news providers. An adapted version of the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model proposed by Bollerslev (1986) is used to calculate the difference between forecasted and realized stock volatility in a defined time window Δt after a news arrival. News articles with a high forecast error are labeled as INTERESTING, all other news articles are labeled as UNINTERESTING. The data are trained using an SVM classifier and a C4.5 decision tree classifier. Choosing the SVM classifier and $\Delta t = 5$ minutes performs best in most test settings.

Forecast accuracy is evaluated through a market simulation using intraday stock prices of stocks in the S&P 100, FTSE 100 and ASX 100 indices in a time period from 1st May 2005 to 31st August 2006. The accuracy measured for the U.S. market is 80%, which is similar to the results Mittermayer & Knolmayer achieved.[7] However, the researchers acknowledge that the prediction of stock

[6]Bloomberg Professional® is a trademark of Bloomberg Finance L.P., a Delaware limited partnership, or its subsidiaries.

[7]The researchers claim in Robertson et al. (2007a) to achieve a higher accuracy than Mittermayer (2004). However, they compare the accuracy (percentage of vectors correctly classified) to the average weighted recall, which is invalid.

prices done by Mittermayer & Knolmayer is harder than the classification in relevant and irrelevant news (Robertson et al., 2007a).

Schumaker & Chen (2006)

Schumaker & Chen aim to forecast intraday stock prices with their system (Schumaker & Chen, 2006, 2010, 2008, 2009).[8] News articles are gathered from Yahoo! Finance and features are extracted using three different approaches, namely bag-of-words, noun phrases and named entities. Unlike in most of the other systems, news articles are not labeled with a single term such as UP or DOWN but are given discrete numeric price predictions. This is achieved by performing linear regression on the minute-by-minute price data 60 minutes prior to the news occurrence and extrapolating what the stock price should be 20 minutes later. For training, the support vector regression (SVR) method described in Vapnik (1995) is used, an adapted version of SVM that is able to handle discrete number analysis.

Performance is evaluated by a market simulation in 26th October to 28th November 2005, which is shorter than the time periods used by most other systems. However, since all stock quotes are gathered on an intraday level, the time period seems to be appropriate. Stocks are traded based on the approach used by Mittermayer & Knolmayer (2006a). Once a news article is published, the system buys (short-sells) the according stock if the price is predicted to be $\geq 1\%$ higher (lower) 20 minutes later. The directional accuracy, meaning the percentage of times the predicted price value was in the correct direction, is 50.7% using the noun phrases approach, outperforming bag-of-words (49.3%) and named entities (49.2%). The researchers claim to achieve a profit of 3.60% using the named entities approach, outperforming bag-of-words (2.24%) and noun phrases (2.15%).

However, looking more closely at the results reveals that 108 trades are necessary to achieve the best performance of 3.60%, investing a total of 108,000\$ and gaining a total of 3,893\$. This makes an average profit of $3,893\$/108 = 36.05\$$ for each trade, boiling it down to a profit per round trip of 0.36% (named entities), 0.22% (bag-of-words) and 0.22% (noun phrases). A 0.36% round trip profit is still better than the ones achieved by most of the other systems. This might be surprising since the directional accuracy is not particularly high (49.2% chance of predicting the correct direction). One explanation could be the conservative trading strategy mentioned in Schumaker & Chen (2006): Taking only named entities into account leads to less trades than using bag-of-words or noun phrases,

[8]A different version of the system taking Sentiment Analysis into account (Schumaker et al., 2012a,b), is not described here in detail.

which in turn decreases the probability of purchases that cause losses.

Groth & Muntermann (2008)

Groth & Muntermann developed a system forecasting intraday price movements of German stocks using ad hoc disclosures enforced by German law (Groth & Muntermann, 2008, 2009). Announcements are obtained from the Deutsche Gesellschaft für Ad-hoc-Publizität (DGAP) and are automatically labeled as POSITIVE or NEGATIVE, depending on whether the according stock price 15 minutes after the announcement time is higher or lower than the price at the announcement time. The authors assume this time frame to be too short for significant influences by simultaneous market fluctuations. Therefore, they do not take the market index into account. Features are extracted using the bag-of-words approach. The training is performed using an SVM classifier with a linear kernel.

Performance is evaluated by means of a 10-fold cross validation. The SVM approach is compared to a *DefaultLearner* (Mierswa et al., 2006) that creates a model based on a default value for all examples. Since there are more instances labeled as POSITIVE, the *DefaultLearner* simply classifies all instances POSITIVE. The overall accuracy of the SVM approach is 56.50% and worse than the *DefaultLearner* accuracy (60.76%). Financial performance is evaluated using ad hoc announcements published between 1st August 2003 and 29th July 2005. On occurrence of an announcement labeled as POSITIVE (NEGATIVE), stocks are purchased (short-sold) and liquidated 15 minutes later. Despite the low accuracy, SVM earns an average 1.05% profit per round trip and outperforms the *DefaultLearner* from the financial point of view. A reason for this discrepancy might be that the SVM approach outperforms the *DefaultLearner* in recall and precision for the NEGATIVE class, which might offset expected losses from the poor classification quality of the POSITIVE class (Groth & Muntermann, 2009). Finally, the authors apply a *t*-test to prove that the results are statistically significant.

Unfortunately, no details are given about the size and amount of trades necessary to achieve the good financial performance. An assumption about expected borrowing and transaction costs is therefore not possible.

The authors developed an adapted version of the system focusing on forecasting stock price volatility rather than prices (Groth, 2010; Groth & Muntermann, 2011). The system uses a similar training and evaluation setup as before, but labels the announcements according to an intraday market risk model proposed by Ahn et al. (2001). Similarly to Schulz et al. the goal of this system is merely to support traders confronted with information overflow issues.

Lin et al. (2011)

Lin et al. try to predict daily stock price movements using financial reports (Lee et al., 2010; Lin et al., 2011). The financial reports are downloaded from EDGAR, the Electronic Data-Gathering, Analysis, and Retrieval system of the U.S. Securities and Exchange Commission.[9] The financial reports are labeled based on the labeling approach proposed by Mittermayer & Knolmayer (2006a). A time window of $t = 2$ days is defined. A financial report is labeled as RISE if stock price increases $\geq 3\%$ at least once and triggers a shift of the average price of $\geq 2\%$ above the opening price of the release day during the time window t. A financial report is labeled as DROP analogously. All other financial reports are labeled as NO MOVEMENT. In a novel approach, the researchers differ between qualitative and quantitative features. Qualitative features are defined as tokens extracted using the conventional bag-of-words approach. Quantitative features are financial ratios regarding company performance, namely operating margin, return on equity (ROE), return on total assets (ROTA), equity to capital, and receivables turnover. In a last step, training is performed using a new approach named HRK (hierarchical agglomerative and recursive K-means clustering).

In a performance evaluation, the researchers use all financial reports and daily opening and closing stock prices of all companies listed in the S&P 500 index from 1st January 1995 to 31st December 2008. All data before 1st January 2006 is used as training data. If a financial report labeled as RISE (DROP) is published, the according stock is purchased (short-sold) at the open of the next trading day and liquidated at the close of two trading days after the release. Performance numbers are reported based on industry sectors and an average performance is not explicitly given. However, considering the number of financial reports used, a weighted average can be calculated. The proposed HRK approach significantly outperforms both SVM and naïve Bayes in terms of accuracy and financial profits. Average accuracy is 65.3% (SVM 62.5%), profits per round trip are 0.67% (SVM 0.34%).

Li et al. (2011)

Li et al. aim to predict movements of stock prices using financial news published in traditional Chinese (Li et al., 2011). News articles are purchased from the commercial platform Caihua and are associated with 23 stocks contained in the Hang Seng index (HSI). The news articles are labeled as POSITIVE (NEGATIVE) if the associated stock price goes 0.3% up (down) t minutes after the news release. The authors choose the threshold of 0.3% as they estimate 0.3% to be the average trad-

[9] `http://www.sec.gov/edgar.shtml`

ing costs on the market. t is chosen to be 5, 10, 15, 20, 25 and 30 minutes to compare the prediction quality of different prediction intervals. Similarly to Lin et al., the authors take quantitative features besides the bag-of-words approach into account. However, they use technical rather than fundamental indicators, namely five market indicators and the relative difference in percentage of price (RPD) extracted following a method proposed by Tay & Cao (2001). Training is performed using a multi-kernel learning approach (MKL) that trains an SVM classifier with bag-of-words features as one sub-kernel and the quantitative features as a second sub-kernel. As a benchmark they use the *News* approach, which ignores the quantitative features and only takes bag-of-words into account.

Performance evaluation is done by means of a 5-fold cross validation from 1st January to 31st October 2001 and an independent test phase from 1st November to 31st December 2001. In all test runs, the MKL approach outperforms the *News* approach. Choosing $t = 20$, which leads to the best prediction quality, MKL has an accuracy of 64.23% (cross validation) or 53.87% (independent testing), bag-of-words 63.06% or 52.38%. A more recent version of the system (Wang et al., 2012) can improve the accuracy of MKL to 65.29% or 56.69% by introducing the stock trading volume as an additional feature. Unfortunately, the authors do not evaluate their system financially, which results in a limited comparability to the other systems.

Hagenau et al. (2012)

Hagenau et al. developed a system forecasting daily stock prices based on corporate announcements enforced by law and published in Germany and the UK (Hagenau et al., 2012). The researchers focus in their work on improving prediction accuracy achieved by earlier systems using different feature types and feature selection methods. Announcements are obtained from DGAP (Deutsche Gesellschaft für Adhoc-Publizität) and EuroAdhoc. Announcements are automatically labeled as POSITIVE or NEGATIVE by comparing daily opening and closing stock prices. Similarly to Schulz et al., the researchers take the market model into account for price calculation. Next, features are extracted using the approaches dictionary (i.e. features are taken from the positive and negative word list in the Harvard-IV-4 dictionary used earlier by Tetlock et al. (2008)) bag-of-words, 2-gram, noun phrases and 2-word combinations (i.e. an extension of 2-gram without the restriction of zero distance between two words). Then a chi-squared based method proposed by Forman (2003) is used to reduce features with low explanatory power. Lastly, data are trained using an SVM classifier.

Performance evaluation is done in a market simulation between 1998 and

2010. Generally, classification accuracy is improved using the chi-squared selection method. Accuracy is best using the 2-word combinations approach combined with chi-squared (65.1%), achieving slightly better results than noun phrases (63.5%). bag-of-words and the Dictionary approach perform worse in most cases. Financial performance is claimed to be on average 1.1% per round trip in the best case (2-word combinations). Unfortunately, crucial details such as the number of trades performed and the holding period of each stock traded are missing. Therefore, the validity of the financial results has to be doubted. However, an interesting finding is that slight percentage changes in accuracy cause high profit increases.

3.2 Key findings

The aspects of the described systems that we consider to be most important for this thesis are summarized in Table 3.1.

The following key findings can be observed:

- Most of the earlier systems (until 2003) use Naïve Bayes or hand-crafted decision rules for training. The more recent systems tend to prefer the SVM classifier or its variations, since it usually leads to higher accuracy rates and profits.

- The bag-of-words approach or hand-picked word phrases are popular methods for feature selection. However, more recently researchers claim the superiority of more complex approaches such as named entities or 2-word combinations (Hagenau et al., 2012; Schumaker & Chen, 2006). Two systems use quantitative features in addition to the bag-of-words approach (Li et al., 2011; Lin et al., 2011).

- The news articles are in most cases assigned three different labels (UP, DOWN, and STABLE) when predicting future prices. An exception is however the system by Schumaker & Chen (2006), which labels the news with discrete price predictions. More recently, prediction systems that use only two labels (UP and DOWN) have become increasingly popular.

- Many systems that use general market news rely on Yahoo! Finance as their data source. Most important reasons for this choice are the diversity of sources and the ease of accessing the data (Schumaker & Chen, 2009). However, the data might contain a lot of noise and the probability of news just summarizing earlier news is high. Therefore, a restriction of the news

System	Prediction type	News type	Feature type	Primary classifier	No. Labels	Price frequency	Profit per round trip
Wüthrich et al. (1998)	Index prices	Market news	Word phrases	Naïve Bayes	3	daily	0.13%
Fawcett & Provost (1999)	Stock prices	N/A	Bigrams	N/A	N/A	daily	N/A
Lavrenko et al. (1999)	Stock prices	Market news	Word phrases	Naïve Bayes	5	10 min	0.23%
Peramunetilleke & Wong (2002)	Exchange rates	Market news	Word phrases	Decision rules	3	60 min	N/A
Gidófalvi & Elkan (2003)	Stock prices	N/A	Bag-of-words	Naïve Bayes	3	10 min	0.10%
Thomas (2003)	Stock volatility	Market news	Word phrases	Decision rules	39	daily	0.10%
Schulz et al. (2003)	Stock volatility	Ad hoc news	Bag-of-words	Regression	2	daily	N/A
Fung et al. (2005)	Stock prices	Market news	Bag-of-words	SVM	3	intraday	N/A
Phung (2005)	Stock prices	Market news	Word phrases	Naïve Bayes	N/A	N/A	N/A
Mittermayer & Knolmayer (2006a)	Stock prices	Press releases	Bag-of-words	SVM	4	1 min	0.29%
Robertson et al. (2006)	Stock volatility	Market news	Bag-of-words	SVM	2	intraday	N/A
Schumaker & Chen (2006)	Stock prices	Market news	Named entities	SVR	numbers	1 min	0.36%
Groth & Muntermann (2008)	Stock prices	Ad hoc news	Bag-of-words	SVM	2	1 min	1.05%
Lin et al. (2011)	Stock prices	Financial reports	qual./quant.	HRK	3	daily	0.67%
Li et al. (2011)	Stock prices	Market news	qual./quant.	SVM multikernel	2	1 min	N/A
Hagenau et al. (2012)	Stock prices	Ad hoc news	2-word combinations	SVM	2	daily	1.10%

Table 3.1: Comparison of all relevant systems

corpus done by a few systems (e.g. Hagenau et al., 2012; Mittermayer & Knolmayer, 2006a) can be considered to be beneficial.

- The performance simulations of all systems are either implicitly or explicitly based on the assumption of zero transaction costs. This leads to a bias in favor of systems needing a high number of trades to achieve the financial performance reported.

These findings summarize ideas and methods that were successfully applied in the past. However, they also discover existing limitations of related works. In the following chapter, we aim to utilize these findings when developing a novel stock price forecasting system.

Chapter 4

Trading system

The system presented in this chapter is based on the idea of predicting intraday stock price movements by analyzing news articles using text mining techniques. The system partly builds on ideas reviewed in the last chapter. When comparing our system to the systems reviewed, the following outstanding characteristics can be highlighted:

- Many of the systems reviewed rely on news sources which can be assumed to be inherently noisy (see Section 3.2). To address this problem, our system uses news releases regulated by the U.S. government as described in Section 2.1.2.
- We use a novel set of features as classification input, which includes in particular named entities, POS tags and document sentiment (see Section 2.2.1).
- We propose different methods to improve the performance when training data that are highly unbalanced.
- We train and evaluate two different data sets resulting from different labeling approaches. One data set contains two classes, the other three classes.
- We implement a binary metalearning algorithm in order to capture the semantics of the underlying training data more precisely.
- Rather than only considering one classifier, we implement and evaluate the performance of all important classifiers described in Section 2.2.5.
- As opposed to earlier systems, we perform a market simulation taking transaction costs into account in order to test the system in a more realistic setting.

In the rest of this chapter, we describe the basic design of the system (Section 4.1) and explain the system implementation in detail (Section 4.2).

4.1 Design

A basic schema of the system is illustrated in Figure 4.1. Its underlying process can be divided into a *training phase*, a *classification phase* and an *evaluation phase*.

In the beginning of the training phase, we acquire news articles (press releases) published by major newswire services and dealing with a set of U.S. companies. Similarly, we download intraday stock prices of the same set of companies. We store both data sets locally. In the news preprocessing step, we filter out certain news articles such as news articles published not in the trading time or news articles associated with many different companies. Since the news articles are provided in a proprietary format, this step also includes extracting the important information from the news files: the release date and time, news headline, news body, and the companies involved in the news. We also convert the news documents into a proper format that will later allow us to feed the data to a classifier. In the next step, we try to reduce the number of irrelevant news by applying a rule-based thesaurus and dismissing all news not matching any of its rules. In the news labeling step, we align the news articles with the stock prices. Since every news article is associated with a company name and the release time, we can automatically assign each news article labels such as BUY, SELL, or HOLD. The choice of each label depends on the according stock's price movement shortly after the news release. Once all news articles in the training data set are labeled, we use a machine learning framework combined with a classification software to train a model based on the data. In the performance tuning phase, we perform an n-fold cross validation with this model and use common evaluation metrics to assess the performance. Based on this evaluation, we change key parameters that are likely to influence the prediction performance. For instance, we extract a different set of features, use different resampling methods, or change classifier parameters. We repeat the parameter tuning, training, and n-fold cross validation until we are satisfied with the prediction performance.

In the classification phase, we acquire news articles published in a time frame different from the one in the training phase in order to ensure that training and test set are independent. Similarly, we acquire the according stock prices. We perform the news preprocessing and the thesaurus filtering step in the same fashion as during the training phase. We then assign the news articles in the original data set labels predicted by the tuned model created during the training phase. As explained in Section 2.2.5, this model is the function Φ. In addition, we create a copy of the test data set. We assign the news articles in the copy of the data set labels created analogously to the training phase and thus obtain the function $\breve{\Phi}$. By comparing both functions, we can evaluate the predicting performance by means of the

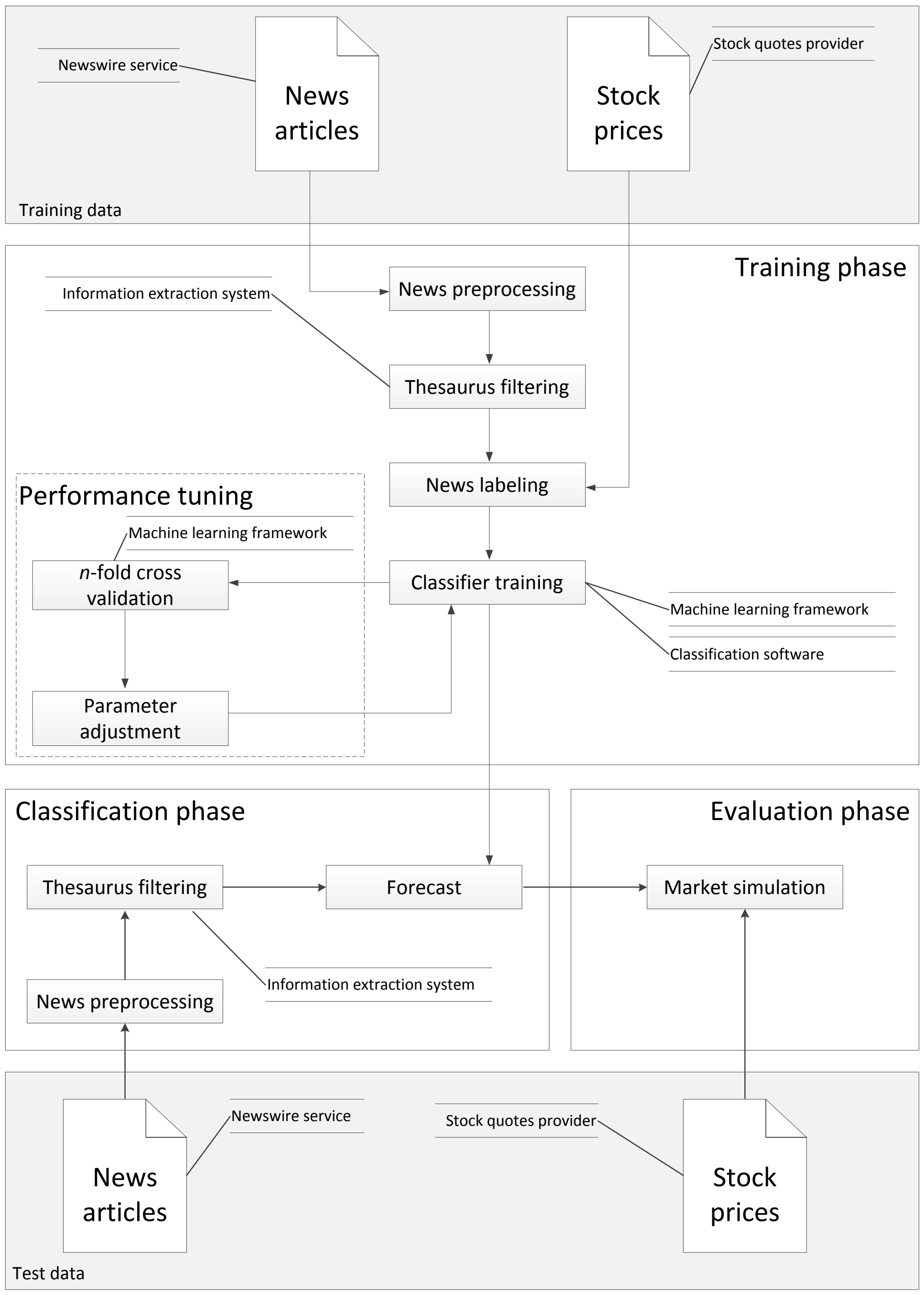


Figure 4.1: Trading system schema

independent test set.[1]

In the evaluation phase, we finally run a market simulation based on the labels predicted by the classifier to evaluate the financial performance of the system. For every news article labeled BUY, we assume to place a buy market order for the according stock. For every news article labeled SELL, we assume to place a short sale order. We liquidate each position after a fixed holding period. This allows us to calculate the profits the system is able to achieve in the independent test period.

4.2 Implementation

In this section, we explain the implementation of the trading system in detail. Most parts of the system are implemented using the programming language Java.

In Section 4.2.1, we describe the data preparation. This includes acquiring news and stock prices, extracting important information from the news articles, filtering out irrelevant news and converting the news into a proper format to prepare them to be fed to a classifier. In Section 4.2.2, we describe the actual classifier training. This includes labeling the news automatically, extracting features that represent the semantics of the news, dealing with the issue of unbalanced data and training a classifier.

4.2.1 Data preparation

Data acquisition

The data used as input for the trading system is comprised of stocks of the U.S. market. This selection is done for the following reasons:

- The U.S. market is one of the most liquid markets worldwide, which decreases the likelihood of non-trading periods and thus makes price predictions more reliable. Moreover, we expect a high news coverage for companies that are very frequently traded.
- Most of the related systems described in Chapter 3 focus on the U.S. market as well. This improves the comparability with our system.
- Most language processing tools freely available focus on the English language.
- The stock and news data is easy to acquire.

[1]For the sake of simplicity, we do not illustrate this independent prediction evaluation in Figure 4.1, since this step is optional and not an essential part of the system schema.

We look only at a subset of publicly traded U.S. companies, namely the ones that are part of the S&P 500 (Standard & Poors 500) index. S&P 500 is a representative subset of the most liquid U.S. stocks. Two types of data need to be acquired for both the training and the testing period: stock quotes and documents containing news articles. Both kinds of data are acquired for the time 6th February to 23rd April 2012 (training period) and 7th May to 15th June 2012 (test period). We intentionally leave a blank time slot of around two weeks between training and test phase in order to ensure that both data sets are as independent as possible.

The stock quotes are obtained from the commercial service Bloomberg Professional[2] that provides real time access to a variety of business data such as analytics, charts and price statistics (Bloomberg, 2012). All stocks are traded on one of the two major U.S. stock exchanges, the NYSE[3] and the NASDAQ[4]. Using the Bloomberg Professional API, we download a list of all companies in the S&P 500 index as of 23rd April 2012 (the last day of the training period). The list contains each company name along with a Bloomberg specific company ticker. For instance, the company Hewlett-Packard belongs to the ticker HPQ UN. The token UN means the stock is traded at NYSE, the token UW means it is traded at NASDAQ. The complete list can be found in Appendix A.A. The tickers are used to obtain all trades executed in the training and test period. The amount of trades obtained for each stock depends on the stock liquidity. In total, we obtain 134,425,927 trades for the training period and 98,553,650 trades for the test period.

An excerpt of an example file is shown in Listing 4.1. The Hewlett-Packard

Listing 4.1: Trade data file example of the Hewlett-Packard stock (GMT time)

```
2012-03-19T16:19:33.000;TRADE;24.54;1900;
2012-03-19T16:19:37.000;TRADE;24.53;800;
2012-03-19T16:20:06.000;TRADE;24.52;4500;
2012-03-19T16:21:53.000;TRADE;24.51;100;
```

stock was traded four times within two minutes and its price dropped from 24.54$ to 24.51$. The last line determines the volume of each trade, e.g. 800 shares were traded at 24.53$. Subsequently, we transform the data into price snapshots taken every 15 seconds. If there is no trade at the snapshot time, we use the *realization method* to calculate missing stock prices, as described in Section 2.1.2. This means, if there is no trade at the snapshot time, the last price available is taken. For

[2]Bloomberg Professional® is a trademark of Bloomberg Finance L.P., a Delaware limited partnership, or its subsidiaries.

[3]`https://nyse.nyx.com/`

[4]`http://www.nasdaq.com/`

instance, the Hewlett-Packard stock trades are transformed into price snapshots of 24.53$ at 16:19:45, 24.53$ at 16:20:00 and 24.52$ at 16:20:15. The next price change is at 16:22:00 to 24.51$. This transformation step is consistent with Mittermayer & Knolmayer (2006a) and will be used for the subsequent labeling step described in Section 4.2.2. The transformation reduces the number of total trades to 58,441,282 for the training set and 21,946,509 for the test set, which means that stocks in the S&P 500 trade on average more frequently than every 15 seconds.

As discussed in Section 2.1.2, newswire services are a reasonable choice for a data source, since they publish press releases enforced by the Regulation Fair Disclosure. The market leaders PR Newswire and Business Wire offer their services through the information provider LexisNexis[5] that aggregates news and other documents from a variety of sources in its database (LexisNexis, 2012a). The data are easily accessible through their LexisNexis Web Services Kit, a SOAP (simple object access protocol) based web service that allows transferring the data in the XML format (LexisNexis, 2012b). Although it is intended for commercial use, LexisNexis kindly provided us with a temporary account for the project. The query shown in Listing 4.2 used to request news for a specific company. The company

Listing 4.2: LexisNexis query

```
COMPANY(Hewlett-Packard Co 9*%)
AND LANGUAGE(English)
AND (PUB(Business Wire) OR PUB(PR Newswire))
AND NOT PUB(PR Newswire Europe)
AND NOT PUB(PR Newswire UK Disclose)
AND NOT PUB(PR Newswire Asia)
AND NOT PUB(Business Wire Video Feed)
AND NOT PUBLICATION-TYPE(Web Blog)
```

name, in this case Hewlett-Packard is taken from the ticker list obtained previously from Bloomberg. The token 9*% is used to retrieve only news articles with at least a 90% relevancy for Hewlett-Packard. The relevancy value is an indicator determined by LexisNexis and the rationale behind it has not been made publicly available. However, a review of news samples led us to the conclusion that the picked news articles are indeed highly relevant to the particular company. Non-English news articles are filtered out as well as news articles from irrelevant news sources that happen to contain the keywords Business Wire or PR Newswire. The query is given to a Java object that represents the LexisNexis request along with additional information specifying the requested return format and time period.

[5]LexisNexis® is a trademark of the LexisNexis family of companies.

News Formatting

The news documents provided by LexisNexis contain the news article itself formatted in HTML as well as a variety of meta information. An example of a news document can be found in Appendix A.B. The information we consider to be useful for the later steps are the news text, the news headline, the publish date and time, and a list of the companies involved in the news article. These pieces of information are enclosed in XML tags. An example is illustrated in Table 4.1. We extract the content within the tags using XPath, a query language developed by the international standards organization World Wide Web Consortium (W3C). For instance, the query in Listing 4.3 is used to extract the news headline. XML tags

Listing 4.3: Extracting a news headline via XPath

```
//html/body/div[@class='HEADLINE']//text()
```

such as `<div>` are referred to as *nodes* in the XPath terminology. The query presented searches for the node `<html>`, traverses into the child node `<body>` and its child node `<div class="HEADLINE">` and extracts the bare text enclosed by the according tags. This simple method has the advantage of filtering out all HTML tags (in this example, the tag `<h1>` and `</h1>`), which are not needed for training a classifier.

News Filtering

This step has the goal of sorting out irrelevant news and eliminating as much data noise as possible. First, we filter out all news articles that are not in the trading hours of NYSE (NYSE, 2012) and NASDAQ (NASDAQ, 2012). This includes weekends, national holidays and news outside the market hours 9:30 am to 4:00 pm. Additionally, we filter out news articles published in the time from 3:40 pm to 4:00 pm to ensure that a 20 minutes price reaction can be observed. Schumaker & Chen (2010) recommend to also cut off news articles in the time from 9:30 am to 10:30 am to reduce noise caused by price reactions on the news articles published over night. However, since this eliminates a large part of news gathered, which means less data that can be trained, we do not perform this cut-off. Another source of noise is the “company in passing” problem mentioned by Schumaker & Chen (2006), meaning that a news article deals with many different companies. For instance, a news article might cause the stock price of Microsoft to rise, but may at the same time mention Google and Yahoo! in a negative context, which may prevent a classifier from interpreting this article correctly. To deal with this

Piece of information	Enclosing tag	Content example
Publishing date and time	`<div class="PUB-DATE"> </div>`	`<span class="hit"> <b>April</b> </span> 10, 2012 Tuesday 12:46 PM EST`
News headline	`<div class="HEADLINE"> </div>`	`<h1>Inventor, Richard P. Mettke of Columbus, Ohio Wins &#34;Round One&#34; Against Hewlett-Packard in Federal Court in Their Attempt to Dismiss His Lawsuit for 275,000,000</h1>`
News body	`<div class="BODY"> </div>`	`<div class="REAL-LEAD"> <p>Inventor, ... breach.</p> </div> <div class="BODY-1"> <p>HP walked away ... to trial.</p> </div>`
Companies involved	`<div class="LN-CO"> </div>`	`<span class="term"> <span class="hit"> <b>HEWLETT</b> </span>-<span class="hit"> <b>PACKARD</b> </span> <span class="hit"> <b>CO</b> </span> <span class="score"> (<span class="hit"> <b>90%)</b> </span> </span> </span> <span class="term">; COMPUSA INC <span class="score"> (53%)</span> </span>`

Table 4.1: Relevant information in a news document

problem, we eliminate news articles that contain more than two company tickers provided in the news meta data. The more restrictive variant would be to also eliminate news with exactly two tickers. However, this would lead to a substantial reduction of the news corpus. Since we have partly addressed this problem by only retrieving news with 90% company relevancy in the first place, we decide not to eliminate news with two tickers.

There may be still some news articles left that are according to human judgment clearly irrelevant to the stock price. It is highly probable that such news articles will confuse a classifier. To address this issue, we use an adapted version of a rule-based thesaurus developed and published by Mittermayer (2006).[6] The thesaurus is hand-crafted and based on a review of financial news with the aim of ensuring their price relevancy. We eliminate all news articles that do not match at least one rule contained in the thesaurus. All rules are written in the JAPE language and executed using the information extraction system ANNIE described in Section 2.2.1. An example rule is shown in Listing 4.4. Before the actual rule is

Listing 4.4: Thesaurus rule example written in JAPE

```
Macro: SEQ
(
({Token.kind != word})*
({Token.kind == word})
({Token.kind != word})*
)

Rule: News06
(
({Token.string ==~ "announc(e|es|ed|ing)"})
(SEQ)[0,10]
({Token.string == "workforce"}
({Token.string == "and"}{Token.string == "facilities"})?
{Token.string == "reduction"})
):m
--> :m.Match = {rule = "News06"}
```

declared, the macro `SEQ` is defined: a `word` token occurs between two sequences of arbitrarily many tokens that are not words (e.g. symbols or punctuation marks). The macro defines the connection between the two parts of the rule `News06`: The line `(SEQ)[0,10]` means that the sequence defined in the macro can be repeated

[6]We use the identical thesaurus, but omit the rules “up” and “down” to improve the effect of eliminating irrelevant news.

zero to ten times in a row. Before this sequence, an inflected form of the word "announce" has to occur. After the sequence, either the phrase "workforce reduction" or the phrase "workforce and facilities reduction" has to occur. If this is the case, the whole sequence is given the name `m`. On the right side of the sign `-->` the annotation `Match` is created for the sequence `m`. In other words, if a form of the word announce is followed by the phrase "workforce reduction" or "workforce and facilities reduction" with a maximum of ten words between them, the rule is fired and a match is declared. As soon as one match is found, the searching is stopped for a particular news article. If no rule finds a match, that news article is eliminated.

We acknowledge that there might be faster methods to extract rule-based patterns. However, the method has the advantage of being easily adaptable. For instance, one could eliminate a news article only if two rule matches occur. Moreover, the runtime does not play a crucial role, since this step only needs to be performed once for the training and once for the test data.

News conversion

For the subsequent training, we will use the freely available machine learning framework Weka developed and maintained at the University of Waikoto (Hall et al., 2009). For this we convert the data in the proprietary file format ARFF (Attribute-Relation File Format) used as input for Weka (Witten et al., 2011, p. 52 ff.). An example for a typical training data set in the ARFF format is shown in Listing 4.5. The first line depicts the name of the relation and is used to uniquely identify the data set. The following lines declare the attributes (features), which can be one of the types `numeric`, `string`, `date`, and `nominal`. The last attribute `signal` is the class attribute, which is nominal and can take on the values `b` (buy), `h` (hold) or `s` (sell). The `@data` token declares the beginning of the list of all instances (news articles). Each of the following lines represents one instance. The instance values are separated by commas. Notice that the last value of both example instances is replaced by a `?` token. This means the value is unknown yet, since we did not perform the labeling yet. All these `?` tokens will be replaced by `b`, `s` or `h` during the labeling step described in the next section.

4.2.2 Training

News Labeling

Having the data prepared as described in Section 4.2.1, the news articles are now ready to be assigned labels. The labeling is based on the assumption that the information in a news article takes 20 minutes to be fully reflected in the stock price.

Listing 4.5: Example for the training data set in the ARFF format

```
@relation LexisNexisNews

@attribute compTicker string
@attribute newsDate date 'yyyy-MM-dd\'T\'HH:mm:ss'
@attribute headLine string
@attribute newsBody string
@attribute @signal@ {b,h,s}

@data
'A UN',2012-04-18T15:00:00,'TRADE NEWS: Agilent
   Technologies Launches Most Versatile LC System Available
    ;  New Infinity 1290 Quaternary LC System Provides
   Powerful Capabilities for Separations Science ','Agilent
    Technologies Inc. (NYSE: A) today introduced the...',?
'A UN','',2012-04-17T19:40:00,'Fitch Affirms Agilent
   Technologies\' IDR at \'BBB+\'; Outlook Stable ','Fitch
   Ratings has affirmed the following ratings ...',?
\%more instances
```

As discussed in Chapter 3, this time period was used by previous systems (Mittermayer, 2006; Schumaker & Chen, 2010). Li et al. (2011) report that the 20 minutes period yields generally good results with their system.

We use an adapted version of the labeling proposed by Mittermayer & Knolmayer (2006a). A snapshot of the stock price is taken at the beginning of the minute the news is published (*start price* P_t at time t). Another snapshot is taken exactly 20 minutes later (*end price* P_{t+1} at time $t+1$). We now calculate the return R over this 20 minutes period. As discussed in Section 2.1.2, a method to calculate returns eliminating the bias introduced by the bid/ask spread is the *logarithmic return*:

$$R = \ln\left(P_{t+1}/P_t\right)$$

If $R > 0.3\%$, the news is labeled as BUY. If $R < -0.3\%$, the news is labeled as SELL. In any other case the news is labeled as HOLD. The threshold of 0.3% is consistent with the work of Li et al. (2011), who argue that profits below 0.3% boil down the profits after transaction costs to zero. Other systems take the price movement of the market into account when calculating the profits (e.g. Hagenau et al., 2012; Schulz et al., 2003). However, as concluded in Section 2.1.2 this is not necessary when dealing with short time periods and we therefore implicitly use the raw returns model.

Our labeling approach results in a 3-class problem with the class values BUY (81 news articles), HOLD (778 news articles) and SELL (80 news articles). This class distribution is heavily unbalanced (around 83% news articles labeled HOLD). As described in Section 2.2.6, classifiers tend to assign all documents to the majority class in such settings. In this case, this is the HOLD class, which will boil down the financial profits to zero. To address this problem, we use another labeling approach producing a 2-class problem. Instead of using the threshold 0.3%, we label news articles as BUY if $R \geq 0$ (581 news articles) and as SELL if $R < 0$ (358 news articles). A similar labeling approach was proposed by Hagenau et al. (2012). We acknowledge that it is likely to result in a trading strategy that buys and sells stocks yielding low returns and not being profitable after transaction costs. However, it might still perform financially better than the 3-class problem in case the prediction performance is much higher.

In order to validate whether the automatic labeling approach generally estimates the price influence of news correctly, we perform a manual labeling on a random sample of the news corpus extracted using the 3-class problem. Although the biggest class is the HOLD class, we draw 80 sample instances of each class as we consider the BUY and the SELL class to be more important. Our primary goal is to determine whether the automatic labeling approach misclassifies many news articles into BUY instead of SELL and similarly, SELL instead of BUY. These types of misclassification errors cause the greatest financial damage. Thus, we manually label only news articles as BUY or SELL. We label news articles as SELL if they contain information such as dividend decreases, credit rating downgrades and patent infringements. We label news articles as BUY analogously. If the decision does not seem obvious or we consider the news to be irrelevant to the stock price, we leave the label blank. In total, we label 40 news articles as BUY, 34 news articles as SELL and leave the remaining 166 news articles blank. The results are the following: 56% of the news articles labeled automatically as BUY are correct compared to the manual labels, the remaining 44% are incorrectly classified as SELL. Analogously, 56% of the news articles labeled automatically as SELL are correct. This means that our automatic labeling approach is able to slightly beat a random labeling (50%).

The results suggest that a trained classifier is not likely to achieve a very high prediction accuracy. However, we decide to use the automatically labeled news data for training the system for the following reasons. First, a manual labeling approach involves the resource-consuming labor of at least one (preferably more than one) domain expert. These resources can be viewed as additional fix costs of the system, which reduce the potential profits. Since the system potentially benefits from more input data, the system would not scale well. Second, a well-documented automatic labeling is transparent and facilitates future work in this area. Third, classifiers might be able to discover price relevant text characteristics that are not obvious to humans.

Feature Extraction

As discussed in Section 2.2.1, there are different approaches to extract features from the news headline and body that can be used to subsequently train a classifier. We use the following approaches:

- Unigrams (bag-of-words)
- Bigrams
- Part of speech (POS) bigrams
- Named entities
- POS categories
- Sentiment

Unigrams can easily be extracted using the Weka *StringToWordVector* filter (Witten et al., 2011, p. 439 f.). It performs tokenization, stop words removal, stemming and TF-IDF value transformation. All of these steps can be tailored to the problem at hand. For tokenization, we use the standard delimiters *space*, *tab* and the symbols `.,;:'"()?!`. Whenever one of these delimiters occurs, the character sequence is split into a new token. For stop words removal, we used a commonly used English stop word list proposed by Lewis et al. (2004). The list is shown in Appendix B.A. For stemming, we use the widely accepted Porter stemmer described in Section 2.2.2. The TF-IDF transformation optionally converts the feature values from word frequencies into $w_{i,j} = TF_{i,j} \times \log(N/DF_i)$, where $TF_{i,j}$ is the term frequency and $\log(N/DF_i)$ is the inverted document frequency (see Section 2.2.3).

To extract bigrams, we use the freely available TagHelper tools (Rosé et al., 2008) that support text analysis in different languages and build on Weka. After the elimination of stop words and stemming, word pairs appearing next to each other are extracted. As described in Section 2.2.1, POS bigrams are similar to bigrams, but they are pairs of 36 grammatical categories referred to as the Penn Treebank tag set.

We extract named entities using the information extraction system ANNIE described in Section 2.2.1. The Semantic Tagger integrated in ANNIE is able to identify and annotate words belonging to predefined categories using a built-in set of JAPE rules. In a first step, we eliminate e-mails and websites since parts of these tokens are often mistakenly annotated as named entities, which introduces a potential noise source. We filter out all words except for the ones annotated with the following categories recommended in the MUC-7 framework (Chinchor, 1998): *Time*, *Location*, *Organization*, *Person*, *Money*, *Percent* and *Date*. These categories have been successfully used by Schumaker & Chen (2009). Additionally, we add all words belonging to the new category *Jobtitle* as we expect those words to contain useful information.

We use ANNIE's POS tagger to extract words belonging to certain POS categories defined in the Penn Treebank tag set. Specifically, we use the POS categories belonging to one of the groups *nouns*, *adjectives* and *adverbs*. Zak & Ciura (2005) report a satisfying performance when taking only nouns, adjectives, adverbs and verbs into account.

For each news article, we calculate a number representing the article sentiment (see Section 2.2.1). For this task, we use the dictionary described by Taboada et al. (2011). It contains words and word phrases (terms) along with an integer value s. The terms are divided into the categories *adjective*, *adverb*, *interjection*, *noun* and *verb*. If $s > 0$, the according term is positive, if $s < 0$, it is negative. The higher the absolute value, the clearer is the positive or negative meaning of the term. For instance, the word "disaster" is associated with -4 (clearly negative), the word "vital" is associated with $+1$ (slightly positive). The sentiment values of all terms contained in a news article are summed up. To avoid a bias towards longer news articles, we divide the result by the total number of words. Although companies issuing press releases generally tend to prefer a positive tone in order to avoid loss of reputation (Mercer, 2004), we expect the sentiment value to be related to the stock price reaction on news.

Handling unbalanced data

As discussed in Section 2.2.6, it is necessary to address the issue of unbalanced data before training a classifier. Since our 3-class problem is clearly unbalanced with 83% of the instances belonging to the majority class HOLD, we use the following approaches:

- Undersampling via *SpreadSubsample*
- Oversampling via *SMOTE*
- Assigning costs via *MetaCost*

Undersampling is done using the Weka filter *SpreadSubsample* (Witten et al., 2011, Chapter 11.3). A random subsample of the majority class is drawn and discarded before the training. It is possible to specify the ratio between the largest and smallest class. For instance, our 3-class dataset contains 778 HOLD instances (largest class) and 80 SELL instances (smallest class). A ratio of 2:1 means that 160 randomly chosen hold instances will be left for training.

SMOTE was developed by Chawla et al. (2002). It performs oversampling by adding synthetic examples to the minority class in the following way. For any examples belonging to the minority class, other examples in the minority class are chosen using the k nearest neighbors technique. The distance between the feature vector of the example under consideration and the feature vector of its nearest neighbor is then multiplied by a random number between 0 and 1. The result is added to the feature vector under consideration. The new synthetic example is therefore placed at a random point between two nearest neighbors. SMOTE has shown better performance than undersampling on unbalanced data sets (Chawla et al., 2002). SMOTE is implemented by means of a Weka filter (Witten et al., 2011, Chapter 11.3). It provides the opportunity to autodetect the minority class, and to specify the oversampling percentage and the number k of nearest neighbors. We choose the oversampling percentage depending on how unbalanced the specific data is. For instance, we calculate the percentage p for our 2-class problem with the majority class c_{buy} and the minority class c_{sell} such that $p = (c_{\text{sell}} - c_{\text{buy}})/c_{\text{buy}}$. This results in a set of examples evenly distributed among both classes, which is recommended by Weiss & Provost (2003), who report a reasonable performance with this distribution for different data sets. We choose the default setting to be $k = 5$ as recommended by (Chawla et al., 2002). After applying SMOTE, we use the Weka filter *Randomize* to randomly reorder all instances in order to avoid any bias caused by synthetic examples created next to each other.

MetaCost was first described by Domingos (1999) and is based on the idea of the *cost matrix C* described in Section 2.2.6. It is implemented by means of a Weka meta classifier and works as follows (Witten et al., 2011, Chapter 8). The training set is randomly divided into several training subsets. Each of these subsets may produce different prediction probabilities. All prediction probabilities are averaged and used to produce a single prediction probability. This process is usually referred to as *bagging* (Breiman, 1996). The *MetaCost* meta classifier minimizes the costs calculated from the probability estimates obtained from bagging and the values of the cost matrix, and relabels each instance in the training set accordingly. Subsequently, it learns one single model based on the relabeled training data. Thus, the costs are implicitly taken into account in the new model.

It is important to understand that these approaches should be applied only on the training data and not on the test data. The filters implementing these steps are called *supervised filters*. For instance, undersampling the majority class via the *SpreadSubsample* in the test data is invalid, since the test data are supposed to be unknown. The result would be a reported performance that cannot be achieved in reality. Weka provides a metalearning scheme named *FilteredClassifier* that wraps the learning algorithm into the filtering process to avoid this behavior. This is also useful for the *StringToWordVector* used for feature extraction. *FilteredClassifier* causes only words in the training set to be extracted. New words occurring in the test set are eliminated (Witten et al., 2011, Chapter 11.3). In order to use different filters and process them successively, we use the Weka filter *MultiFilter*.

Notice that some of the filters used depend on random numbers (e.g. SMOTE for creating the synthetic examples). To make sure that the performance results are stable and do not randomly change each time we build a new model, we specify a constant random seed for each of these filters.

Classifier

Once the news articles are labeled and the features are extracted, a classifier can be trained in order to create a model that is able to label unseen test data. As described in Section 2.2.5, the most common classifiers are the k-nearest neighbor classifier, decision trees, naïve Bayes and support vector machines (SVM). We train each classifier using the Weka framework in order to compare their performance. Weka provides a straight-forward implementation of the naïve Bayes classifier that can be reasonably used without any parameter tuning. The k-nearest neighbor algorithm is implemented by Weka's IBk classifier (Aha et al., 1991). In its standard version, it uses the Euclidean distance as base for the distance function. It is possible to specify the number k of nearest neighbors. The decision tree algorithm C4.5 (see Section 2.2.5) is implemented by the Weka classifier J48.

SVM is implemented using the LIBSVM software package (Chang & Lin, 2011) that can be used by Weka using the wrapper WLSVM developed by EL-Manzalawy & Honavar (2005). LIBSVM provides the commonly used kernels linear, polynomial, radial basis function (RBF), and sigmoid. The linear kernel is a good choice for data with many more features than instances (Hsu et al., 2010), as it is the case in text mining. Groth & Muntermann (2008) also use the linear kernel in their application. However, Mittermayer & Knolmayer (2006a) report that some non-linear kernels outperform the linear kernel. Thus we evaluate the performance of the linear kernel and the RBF kernel. While it is only necessary to tune the cost parameter C for the linear kernel, Hsu et al. (2010) stress the importance of tuning the parameter C and γ (see Section 2.2.5) for non-linear kernels. The tuning is implemented by LIBSVM as follows (Hsu et al., 2010): Exponentially growing pairs of C and γ are successively trained and evaluated using k-fold cross validation. The (C, γ) with the highest overall accuracy is taken and trained again to build the final model. An example is illustrated in Figure 4.2. The colored lines indicate ar-

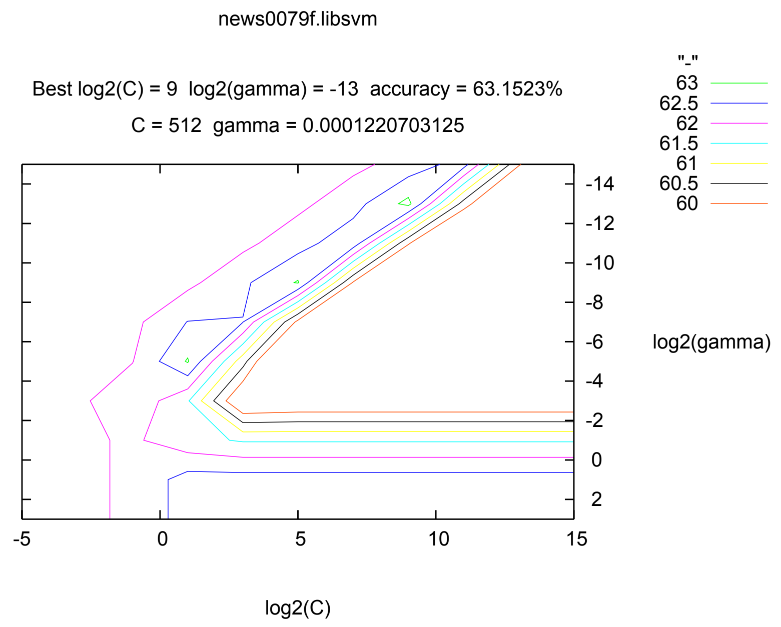


Figure 4.2: LIBSVM parameter tuning

eas in which a certain level of accuracy is reached. In this example, the parameter pair $C = 2^9$ and $\gamma = 2^{-13}$ yields the highest accuracy (63.1523%) and therefore is used to create the final model.

In Section 4.2.2, we introduced the 3-class problem that deals with the labels BUY, HOLD and SELL. In the following, we transform this problem into two binary class problems hoping to improve the classifier performance by better capturing the semantics of the problem. The main idea behind the transformation is that the BUY class and the SELL class can be considered more important than the HOLD class. The reason is that news articles labeled as BUY or SELL will trigger a trade and cause transaction costs and (negative) profits, whereas news articles labeled as HOLD do not trigger any financial transaction. We build the new problem as follows. All instances of the original 3-class problem are copied twice. In the first copy, the label values are changed to SELL or NO SELL. In the second copy, the label values are changed to BUY or NO BUY. We perform these relabeling steps conveniently using the Weka filter *MakeIndicator* that is able to replace any nominal attribute by a boolean one. Subsequently, each copy is used to train a separate classifier. If one classifier predicts the label BUY for an instance and the other classifier predicts NO SELL, this instance is labeled BUY. Similarly, if one classifier predicts SELL and the other one predicts NO BUY, the final prediction is SELL. In all other cases, the final prediction is HOLD. We implement this novel classifier by extending the abstract Java class RANDOMIZABLESINGLECLASSIFIERENHANCER provided by the Weka API. Our classifier is designed as *metalearning algorithm*, meaning that it modifies the functionality of an arbitrary classifier (Witten et al., 2011, Chapter 11).

Chapter 5

Evaluation

The main goal of the system described in the last chapter is to forecast short term stock price movements in order to realize highest possible profits. In this chapter, we perform an evaluation of the system with two main objectives: First, we aim to evaluate the prediction performance of our system and research how we can increase the performance by varying the labeling approach and the training characteristics. By optimizing the prediction performance, we hope to increase the potential financial performance of the system. Second, we aim to evaluate how our system performs financially. We do so by designing and running a market simulation that allows the system to operate under conditions close to reality.

5.1 Evaluation settings

As described in the last chapter, we made a number of design decisions that may influence the performance of the system. The most important system characteristics can be divided into the *input data*, *labeling approach* and *classifier training*. In the following, we describe the system characteristics that are fixed. We leave those characteristics unchanged throughout the evaluation process.

All choices made regarding the *input data* naturally remain fixed. The news sources of the system are press releases published by the newswire services PR Newswire and Business Wire. The news articles are published from 6th February to 23rd April 2012 (training period) and 7th May to 15th June 2012 (test period). We only take news articles into account dealing with companies that are part of the S&P 500 index. We only consider news articles published between 9:30 am and 3:40 pm within trading days. We retrieve only news articles with 90% relevancy for the respective company (according to LexisNexis). Subsequently, we filter out news articles containing more than two company tags. Similarly, we discard news articles rejected by the thesaurus.

We fix a few characteristics regarding the *labeling approach*. The most important one is to use the time frame of 20 minutes in order to label the news articles depending on the stock price movement. As stated in Section 4.2.2, we base our decision on previous research, yielding good results with the 20 minutes time frame. This also increases the performance comparability to similar systems. Furthermore, as concluded in Section 4.2.2, it is beneficial to use the logarithmic return to calculate the stock returns for labeling.

Regarding the *classifier training*, we fix the text preprocessing steps of removing stop words and stemming all words using the Porter stemmer. These choices are consistent to most systems described in Chapter 3 and as stated in Section 2.2.2, they have been successfully used in practice.

5.2 Evaluation methodology

Based on the fixed settings described in the last section, we vary the set of system characteristics that we consider to be crucial for the prediction performance. In order to conveniently manipulate all characteristics, we store all relevant settings in a public Java class. The class contains public constants, each representing a setting variable such as the label threshold, the classifier used, or one of the training parameters. In the following, we describe the characteristics we vary.

As for the *input data*, we leave all settings described in the last section fixed. Varying them would unnecessarily increase the complexity of the evaluation.

We vary the *labeling approach* in one aspect: By choosing different thresholds for determining the labels of the news, we produce a 3-class problem (threshold 0.3%) and a 2-class problem (threshold 0%). As described in Section 4.2.2, the 2-class problem can help to address the problem of unbalanced data. In addition, in Section 3.2 we concluded that related systems have been successfully developed based on both types of problems. We take the novel approach to evaluate the performance of both a 2-class and a 3-class problem using the same input data.

Regarding the *classifier training*, there is a number of characteristics that are of interest in the area of text mining. First, the feature representation can be changed between unigrams (bag-of-words), bigrams, POS bigrams, named entities, POS categories, sentiment, and combinations of these features. While applying complex features rather than the simple bag-of-words representation has generally shown limited success in the past (Moschitti & Basili, 2004), stock forecasting systems were able to improve their performance by using named entities (Schumaker & Chen, 2006) and bigrams (Hagenau et al., 2012). Another parameter that might influence the system performance, is the choice of a dimensionality reduction method described in Section 2.2.4. Hagenau et al. (2012) report a performance improvement by using the chi-squared method, Mittermayer & Knolmayer (2006a) increase their system accuracy by using term frequency to reduce the number of features. Moreover, we vary the popular weighting schemes TF-IDF and boolean. For training, we compare four of the most popular classifiers (see Section 2.2.5), namely the k-nearest neighbor classifier, the decision tree algorithm C4.5 using its Weka implementation J48, naïve Bayes and support vector machines (SVM) with the linear and the radial basis function kernel. In addition, we vary the SVM parameters C and γ as described in Section 4.2.2. In order to achieve a good performance despite the unbalanced nature of the data, we compare the approaches undersampling, oversampling, and the assignment of costs. For the 3-class problem, we finally evaluate how the application of the binary metalearning algorithm introduced in Section 4.2.2 influences the classifier performance.

5.3 Results

In this section, we evaluate the classification performance of the *3-class* problem and the *2-class* problem separately. We use the evaluation metrics described in Section 2.2.6. Since both problems contain comparatively few instances, we choose to use the 10-fold cross validation approach implemented by Weka on the training set. Subsequently, we evaluate the performance on an independent test set.

5.3.1 2-class problem

In the initial setting, we use bag-of-words as features and perform no dimensionality reduction. We use the SVM classifier, which is used by the most systems described in Chapter 3. Joachims (1998) reports that the SVM applied to text categorization tasks outperform other classifiers. As concluded in Section 4.2.2, the linear SVM kernel is a reasonable first choice. As opposed to other SVM kernels, the linear kernel does not involve γ (see Section 2.2.5), and the only parameter that needs to optimized is C. In order to ensure a stable performance, we optimize C in this and each further setting separately by performing the LIBSVM optimization approach described in Section 4.2.2 and fixing γ to an arbitrary value. The classification performance is based on ten models that are created and evaluated by Weka independently. The results are shown in Table 5.1. Weka performs important metrics calculations based on the confusion matrix. They are described in Section 2.2.6 and include the accuracy (A), precision (π), recall (ρ) and balanced F-measure (F_1). The last three metrics are given for both the BUY and SELL class. Consistently with Mittermayer & Knolmayer (2006a), we use macro-averaging to calculate the combined results for both classes.

Training	π^{buy}	ρ^{buy}	π^{sell}	ρ^{sell}	F_1^{buy}	F_1^{sell}	F_1	A
SVM	62.0%	97.4%	42.3%	3.1%	75.8%	5.8%	40.8%	61.4%
Default	61.9%	100.0%	0.0%	0.0%	76.5%	0.0%	38.2%	61.9%
Random	61.9%	50.0%	38.1%	50.0%	55.3%	43.3%	49.3%	50.0%
Test								
SVM	57.1%	98.4%	42.9%	1.6%	72.3%	3.1%	37.7%	56.9%
Default	57.1%	100.0%	0.0%	0.0%	72.7%	0.0%	36.3%	57.1%
Random	57.1%	50.0%	42.9%	50.0%	53.3%	46.2%	49.7%	50.0%

Table 5.1: Evaluation with the initial setting on the 2-class problem. The training set is evaluated by means of a ten-fold cross validation.

When we evaluate the performance on the independent test set, we observe a performance drop (61.4% in the training set versus 56.9% in the test set). This behavior is consistent with the findings of Hagenau et al. (2012) and shows that it is essential to evaluate the classifier performance on an independent test set.[1] We compare the results with two different benchmark approaches. First, we use the *RandomLearner* approach proposed by Mittermayer & Knolmayer (2006a). It labels all instances randomly assuming that the instances are distributed uniformly. The recall is for both classes 50% since the *RandomLearner* on average labels half of all examples correctly. The precision depends on the class size since the number of examples labeled correctly at random increases with the class size. Second, we adapt the *DefaultLearner* benchmark method used by Groth & Muntermann (2008). As the BUY class is the majority class in our 2-class problem, the *DefaultLearner* labels all instances BUY. Since no instances are labeled SELL, the *DefaultLearner* naturally has a π, ρ and F_1 of zero for the SELL class. The overall accuracy of the *DefaultLearner* is lower in the independent test set than in the training set. The reason is that the test set happens to be less unbalanced (57.1% BUY instances). Notice that the SVM as well labels a large percentage of examples BUY, which results in a low ρ value (1.6%) for SELL and a high ρ value for BUY (98.4%). The reason might be that the data set is unbalanced and the SVM therefore tends to classify most examples into the majority class, as suggested by Ben-Hur & Weston (2010). We will address the unbalanced data problem later in this section. The overall accuracy of SVM is slightly lower than the accuracy of

[1]The performance drop can also partly be explained with different class distributions of training and test set.

	F_1^{buy}	F_1^{sell}	F_1	A
Bag-of-words	75.8%	5.7%	40.8%	61.4%
POS categories	75.9%	2.7%	39.3%	61.3%
POS bigrams	76.2%	0%	38.1%	61.6%
Bigrams	75.2%	17.9%	46.6%	61.9%
Named entities	75.2%	5.7%	40.5%	60.7%

Table 5.2: Performance comparison with different feature sets

the *DefaultLearner*. However, in terms of overall balanced F-measure, the SVM outperforms the *DefaultLearner* on both training and test set.

For the subsequent tuning steps, we only illustrate the performance metrics that we consider to be most important, which are F_1^{buy}, F_1^{sell}, F_1, and A. We take the balanced F-measure into account rather than weighting recall and precision differently for the following reason. A false positive in the BUY class leads to an unintended stock purchase, a false negative leads to an unintended short sale. Both events can be expected to be equally harmful for the financial performance of the system. Analogously, this is true for the SELL class. All results including the ρ and π values are shown in Table B.3 in Appendix B.B.

In the next step, we evaluate the influence of different feature subsets on the classification performance. The results are depicted in Table 5.2. In contrast to the findings of Schumaker & Chen (2006), we find the named entities feature representation to perform worst in terms of accuracy. This might be due to the following reasons. First, the named entity feature representation ignores big parts of the texts that might contain valuable information. Second, subsuming word information by the information in named entities is not always possible. For instance, the feature "George_Bush" learned by the classifier during the training phase will not trigger the word "Bush" in a test document (Moschitti & Basili, 2004). Bag-of-words (BoW) and POS categories perform similarly well. We conclude that taking out all words except nouns, adjectives and adverbs does not strongly influence the classifier performance. The best accuracy (61.9%) and overall balanced F-measure (46.6%) is obtained by the bigrams representation. Bigrams also outperform the other feature sets in terms of the SELL class F-measure, while obtaining an only slightly lower BUY class F-measure. This confirms the findings of Wang & Manning (2012), who report a performance improvement of bigrams compared to BoW in a sentiment classification task. Thus, bigrams might capture semantics relevant

	F_1^{buy}	F_1^{sell}	F_1	A
Bigrams	75.2%	17.9%	46.6%	61.9%
Bigrams, BoW	75.1%	5.1%	40.1%	60.5%
Bigrams, BoW, POS bigrams	75.8%	10.6%	43.2%	61.8%
Bigrams, POS bigrams	75.8%	5.2%	40.5%	61.4%

Table 5.3: Bigrams combined with different feature sets

to investors better than the other representation approaches.

We subsequently combine the best performing feature representation bigrams with other representations as shown in Table 5.3. Except for the F-measure of the class BUY, all performance indicators drop when adding different feature sets. The additional features might introduce more noise without carrying more essential information. However, using bigrams, POS bigrams and BoW as features outperforms the two other combinations (bigrams, POS bigrams and bigrams, BoW) in both F-measure and accuracy. In addition, we enhance the bigram feature set with the sentiment feature. The performance of this setting is $F_1^{\text{buy}} = 76.1\%$, $F_1^{\text{sell}} = 3.2\%$, $F_1 = 48.3\%$ and $A = 61.7\%$, which is a slight accuracy decrease. A possible explanation for this behavior is that the sentiment values are not a good indicator of how the news articles are supposed to be labeled and therefore confuse the classifier. Although there is evidence that the sentiment in earnings press releases influences investor's trading decisions (Henry, 2008), managers have more incentives to provide investors with positive disclosures than with negative ones (Mercer, 2004), which may cause them to publish press releases with a misleadingly positive tone. We fix the bigram feature set without sentiment, since this setting obtains the best performance so far.

In the next step, we change the boolean feature representation to the TF-IDF weighting approach (Section 4.2.2). The results are slightly worse than using the boolean representation: $F_1^{\text{buy}} = 74.5\%$, $F_1^{\text{sell}} = 17.6\%$, $F_1 = 46.1\%$ and $A = 61.1\%$. Thus, we do not change the boolean representation.

Using bigrams as features could improve the SELL class F-measure to 17.9%. Since this result is still comparatively low and the sell class is the minority class, it might be possible to increase the performance by using the methods *SMOTE* and *SpreadSubsample* to handle unbalanced data proposed in Section 4.2.2. For both methods, we choose parameters that result in an even distribution of the SELL and the BUY class, as recommended by Weiss & Provost (2003). That means, we

	F_1^{buy}	F_1^{sell}	F_1	A
Standard	75.2%	17.9%	46.6%	61.9%
SpreadSubsample	58.6%	49.0%	53.8%	54.3%
SMOTE	64.9%	44.5%	54.7%	57.0%

Table 5.4: Different approaches to deal with the unbalanced data set

specify the ratio between the smallest and the largest class for SpreadSubsample as 1, resulting in both classes containing 358 examples. For SMOTE, we apply the approach explained in Section 4.2.2 to obtain a data set with 581 examples in each class.

The results are depicted in Table 5.4. We compare the results of SpreadSubsample to the results of the setting without resampling (standard). The F-measure of the SELL class highly increases (49.0% versus 17.9%), but the BUY class F-measure drops (58.6% versus 75.2%). This is not surprising, since the prediction model has less tendency to classify examples into the majority class. However, despite a higher overall F-measure, the accuracy drops to 54.3%. This can be explained by the fact that the BUY class performance decrease outweighs the SELL class performance increase due to their difference in size. As discussed in Section 2.2.6, undersampling might delete potentially useful examples. The oversampling done by SMOTE also leads to a worse accuracy compared to the standard approach, but the best overall F-measure. Both overall F-measure and accuracy are better than using the SpreadSubsample approach. This confirms the findings of Weiss et al. (2007), who recommend using oversampling rather than undersampling when dealing with small data sets. The reason of outperforming the standard approach is the better F-measure of the SELL class (44.5% versus 17.9%), and a comparatively smaller difference in the BUY class F-measure (64.9% versus 75.2%). Since there is no clear winner regarding accuracy and F-measure, we continue with two further evaluation sets, one using SMOTE, the other one using standard (no resampling).

We compare the linear SVM classifier with all other classifiers discussed in Section 2.2.5. As recommended by Hsu et al. (2010), for the SVM with the radial basis function (RBF) kernel we optimize γ and C using the parameter optimization implemented by LIBSVM (see Section 4.2.2). For this setting, we obtain the optimal value pair $\gamma = 2^{-1} = 0.5$ and $C = 2^3 = 8$. We will perform the γ and C optimization for each further setting separately. For k-NN, we choose $k = 10$, which is consistent with Mittermayer & Knolmayer (2006a). The results are shown in Table 5.5. k-NN achieves the highest F-measure for the SELL class, but the

	F_1^{buy}	F_1^{sell}	F_1	A
SVM linear	64.9%	44.5%	54.7%	57.0%
SVM RBF	71.8%	30.9%	51.4%	60.0%
J48	72.9%	22.6%	47.8%	59.9%
Bayes	74.8%	16.2%	45.5%	61.2%
k-NN	27.4%	57.5%	42.5%	46.3%

Table 5.5: Different classifiers with the SMOTE approach

	F_1^{buy}	F_1^{sell}	F_1	A
SVM linear	75.2%	17.9%	46.6%	61.9%
SVM RBF	75.8%	13.1%	44.5%	62.1%
J48	73.7%	16.4%	45.1%	60.1%
Bayes	71.3%	30.3%	50.8%	59.3%
k-NN	76.5%	0.6%	38.6%	62.0%

Table 5.6: Different classifiers without resampling

BUY class F-measure is the lowest. This results in the worst overall F-measure and accuracy values, leading to an accuracy even worse than the accuracy of the *RandomLearner*. SVM RBF outperforms both J48 and k-NN in terms of overall F-measure and accuracy, confirming the results of Joachims (1998), who argues that SVMs are most suitable for text classification due to their superior ability to handle high dimensional feature spaces, sparse feature vectors, and few irrelevant features. However, Bayes achieves the highest overall accuracy, although it has the lowest F-measure for the SELL class. Between SVM linear and SVM RBF there is no clear winner. SVM linear performs better in terms of overall F-measure, SVM RBF in terms of overall accuracy.

When we compare the different classifiers without any resampling (Table 5.6), all F-measure values for the SELL class are lower than with SMOTE (except for Bayes). This indicates that the predictions tend to be more biased towards the majority class. k-NN reaches an overall accuracy similar to SVM, but labels only

	F_1^{buy}	F_1^{sell}	F_1	A
No reduction	75.8%	13.1%	44.5%	62.1%
IG	75.5%	11.3%	43.4%	61.7%
CHI	75.5%	11.3%	43.4%	61.7%

Table 5.7: Different methods of dimensionality reduction

a few examples SELL, leading to the worst SELL class F-measure. Bayes seems to more reluctant to the majority bias and achieves the best SELL class F-measure and the best overall F-measure. SVM RBF performs with respect all metrics similarly to SVM linear but has the best overall accuracy of all settings with or without SMOTE. Therefore, we use SVM RBF without SMOTE for the next step.

As discussed in Section 2.2.4, dimensionality reduction can help to reduce training time and address the problem of overfitting. Mittermayer & Knolmayer (2006a) achieve the best performance when they determine the numbers of features that are kept to 15% of the documents available. For our data set, that means keeping only 140 features for training. This number may seem surpisingly low, but is in line with Jain & Chandrasekaran (1982), who recommend using instances five to ten times the number of features. We only take the metrics information gain (IG) and chi squared (CHI) into account, since they perform superior compared to the other metrics for the task of text classification (Yang & Pedersen, 1997). The results are shown in Table 5.7. IG and CHI perform identically as both metrics tend to choose a nearly identical feature set for training. This confirms the findings of Yang & Pedersen (1997), who report a high correlation between both metrics. The performance slightly drops both in terms of F-measure and accuracy compared to the full feature set. Reducing the features does not seem to have a strong negative influence on the overall performance. However, in contrast to Hagenau et al. (2012) we do not observe that the CHI feature selection reduces overfitting, which would result in a better performance for the independent test set. Accuracy in the independent test set is also slightly lower with IG and CHI than with all features (56.0% versus 56.3%).

We compare the final setting, which is the SVM RBF with no dimensionality reduction (tuned SVM), to the performance of the *DefaultLearner* and the linear SVM used in the first setting (basic SVM, see Table 5.1) on both the training set and the independent test set. The results are shown in Table 5.8. On the training set, the tuned SVM performs similarly or better than the basic SVM with respect to all performance metrics and outperforms both basic SVM and the *DefaultLearner*

Training	π^{buy}	ρ^{buy}	π^{sell}	ρ^{sell}	F_1^{buy}	F_1^{sell}	F_1	A
Basic SVM	62.0%	97.4%	42.3%	3.1%	75.8%	5.8%	40.8%	61.4%
Tuned SVM	62.7%	95.7%	51.9%	7.5%	75.8%	13.1%	44.5%	62.1%
Default	61.9%	100.0%	0.0%	0.0%	76.5%	0.0%	38.2%	61.9%
Test								
Basic SVM	57.1%	98.4%	42.9%	1.6%	72.3%	3.1%	37.7%	56.9%
Tuned SVM	57.9%	85.5%	47.1%	17.2%	69.0%	25.2%	47.1%	56.3%
Default	57.1%	100.0%	0.0%	0.0%	72.7%	0.0%	36.3%	57.1%

Table 5.8: Evaluation with the final setting on the 2-class problem

with respect to overall F-measure and accuracy. On the independent test set, the tuned SVM performs better in terms of the SELL class F-measure but slightly worse in terms of BUY class F-measure than basic SVM and *DefaultLearner*. Since the buy class is larger than the sell class, this leads to a higher overall F-measure but a lower overall accuracy compared to the basic SVM and the *DefaultLearner*. These findings are similar to the ones obtained by Groth & Muntermann (2009). Although their SVM classifier obtains worse accuracy than the *DefaultLearner*, they achieve significantly better financial performance than the *DefaultLearner*.

5.3.2 3-class problem

In the following, we present the evaluation results on the 3-class problem. Since this data set is highly unbalanced (82.9% HOLD, 8.6% BUY, 8.5% SELL), our main focus in this section is to address this issue. We evaluate the F-measure for each class separately, the macro-averaged F-measure, and the overall accuracy. In contrast to the 2-class problem, the importance of the different classes differs with respect to the expected financial performance of the model. Classification errors in the BUY or SELL class can lead to high losses if the classifier predicts BUY instead of SELL or the other way round. Classification errors in the HOLD class are likely to cause either no trade at all or a wrong trade with small positive or negative profits. Thus, we focus on the single class F-measure values rather than only considering overall F-measure and accuracy when performing the parameter tuning. For similar reasons it could be argued that precision and recall are not equally important: For the HOLD class, false negatives for are more important than false positives. For the BUY and the SELL class, false positives are more important than false negatives. However, we choose the balanced F-measure (F_1) rather than a weighted F-measure as it increases the consistency with related systems that use

	F_1^{buy}	F_1^{hold}	F_1^{sell}	F_1	A
SVM linear	0%	90.5%	0%	30.2%	82.6%
SVM RBF	0%	90.4%	0%	30.1%	82.5%
J48	0%	89.4%	4.1%	31.2%	80.5%
Bayes	17.5%	50.1%	17.0%	28.2%	36.2%
k-NN	0%	90.6%	0%	30.2%	82.9%
Random	13.7%	47.5%	13.6%	24.9%	33.3%

Table 5.9: Different classifiers on the 3-class problem

F_1 as performance indicator (e.g. Mittermayer & Knolmayer, 2006a). All results including recall and precision separately for all classes are presented in Table B.4 in Appendix B.B. Furthermore, the *DefaultLearner* is not satisfying as a benchmark for the 3-class problem since it would label all instances as HOLD, which is what we aim to avoid. Therefore, we consistently with Mittermayer & Knolmayer (2006a) only use the *RandomLearner* as a benchmark in the following.

In the first setting, we use the parameter combination that performed best in the last section, namely bigrams as feature set, no use of the sentiment feature, the boolean feature representation, and no dimensionality reduction. We compare the performance of the different classifiers with the *RandomLearner* (Table 5.9). For SVM linear, SVM RBF and k-NN, we observe an F_1 of zero for both the BUY and the SELL class. Similarly, J48 performs poorly for the BUY and SELL class. As concluded in the last section (see Table 5.6), the classifiers tend to predict in favor of the majority class. Similarly with the results on the 2-class problem, Bayes is the classifier least affected by this issue. It outperforms the *RandomLearner* with respect to all performance metrics. The performance drop in the overall accuracy compared to the other classifiers does not necessarily lead to a worse financial performance, since it is caused by misclassification errors in the class HOLD, which are expected to be comparatively harmless.

Next, we compare the performance of all classifiers when applying no further approach (standard) with SMOTE, SpreadSubsample, and the *MetaCost* approach described in Section 4.2.2. When applying SMOTE, we oversample both the BUY and the SELL class until all classes have equally many examples. Similarly, when applying SpreadSubsample, we undersample the HOLD class until all classes have equally many examples, as recommended by Weiss & Provost (2003).

We apply MetaCost in a fashion similar to Domingos (1999). Let $Pr(i)$ be the probability that an example belongs to class i in the training set. For $i = j$, we choose $C(i,j) = 0$. If $i \neq j$ and if i represents the HOLD class, we choose $C(i,j) = Pr(i)/Pr(j)$. In any other case, we choose $C(i,j) = 1$. Thus, the costs of false positives of the majority class HOLD are assigned dependent on its size relative to the minority class sizes.

The results are shown in Table 5.10. Due to the resampling or cost assignment, we observe the performance of all classifiers to drop for the HOLD class and to increase for the BUY and SELL class. An exception for this behavior is the Bayes classifier, which was not strongly affected by the unbalanced data in the initial setting and therefore does not benefit as much from the resampling or cost assignment. Its overall accuracy is worse than the *RandomLearner* for all three methods. The k-NN classifier performs in terms of overall F-measure and accuracy better using MetaCost than using SMOTE or SpreadSubsample. When training the data with SVM linear, SVM RBF and J48, SMOTE does not strongly influence the performance. The overall F-measure increases but the overall accuracy decreases. The F-measure values for BUY and SELL class are lower than the ones of the *RandomLearner*. The reason for this might be an overgeneralization problem: The artificial examples in the minority class are created without considering the majority class. This problem increases in case of highly unbalanced data (Bunkhumpornpat et al., 2009). Undersampling via SpreadSubsample leads to a relatively high performance increase of the BUY and the SELL class. J48 outperforms the SVM classifiers in both overall F-measure and accuracy and outperforms the *RandomLearner* in all performance metrics. The effect of MetaCost on the performance strongly varies when changing the classifier. For SVM RBF, we observe a comparatively high increase of the BUY and SELL class F-measure but a strong decrease of the HOLD class F-measure. Overall F_1 and A are below the *RandomLearner*. In contrast, for SVM linear and J48 the overall accuracy remains similar but the BUY and SELL class performance suffers and is not higher than the performance of the *RandomLearner*. The varying performance of MetaCost may have the reason that its performance heavily relies on the instability of the base classifier (Chawla et al., 2008) as bagging is used to estimate the underlying prediction probabilities (see Section 4.2.2).

The two settings that outperform the *RandomLearner* in all metrics are highlighted in gray color. In the next step, we train the data with both settings using the binary metalearning algorithm introduced in Section 4.2.2. The results are shown in Table 5.11. For both J48 and Bayes, we observe a strong increase of the HOLD class F-measure and a slight decrease of the BUY class F-measure (J48) or the SELL class F-measure (Bayes). The reason for this might be that the metalearning classifier transforms the 3-class problem into two binary problems and therefore

		F_1^{buy}	F_1^{hold}	F_1^{sell}	F_1	A
SVM linear	Standard	0%	90.5%	0%	30.2%	82.6%
	SpreadSubsample	13.4%	55.7%	15.8%	28.3%	40.1%
	SMOTE	2.1%	89.4%	11.5%	34.3%	80.7%
	MetaCost	0%	90.2%	4.6%	31.6%	82.2%
SVM RBF	Standard	0%	90.4%	0%	30.1%	82.5%
	SpreadSubsample	16.5%	48.7%	6.8%	24%	34.7%
	SMOTE	4.5%	90.0%	12.1%	35.5%	81.9%
	MetaCost	13.0%	41.3%	14.1%	22.8%	28.5%
J48	Standard	0%	89.4%	4.1%	31.2%	80.5%
	SpreadSubsample	16.7%	56.1%	14.6%	29.1%	40.5%
	SMOTE	6.1%	85.8%	4.5%	32.1%	74.4%
	MetaCost	4.5%	86.3%	15.3%	35.4%	75.1%
Bayes	Standard	17.5%	50.1%	17.0%	28.2%	36.2%
	SpreadSubsample	15.5%	36.5%	15.9%	22.6%	26.6%
	SMOTE	14.6%	46.6%	10.9%	24.0%	32.5%
	MetaCost	15.4%	23.7%	15.3%	18.1%	19.3%
k-NN	Standard	0%	90.6%	0%	30.2%	82.9%
	SpreadSubsample	26.5%	38.9%	9.9%	25.1%	30.8%
	SMOTE	17.7%	43.0%	14.7%	25.1%	30.9%
	MetaCost	10.8%	52.5%	17.0%	26.8%	37.5%
	Random	13.7%	47.5%	13.6%	24.9%	33.3%

Table 5.10: Different approaches to deal with the unbalanced data set

		F_1^{buy}	F_1^{hold}	F_1^{sell}	F_1	A
J48 SpreadSubsample	Standard	16.7%	56.1%	14.6%	29.1%	40.5%
	Binary	11.8%	69.3%	17.3%	32.8%	53.2%
Bayes	Standard	17.5%	50.1%	17.0%	28.2%	36.2%
	Binary	18.4%	72.1%	13.4%	34.6%	55.8%

Table 5.11: Performance evaluation of the binary metalearning algorithm

creates two more unbalanced data sets. For instance, a data set with $1/3$ instances in each class is transformed into two problems with a $1/3$ to $2/3$ distribution. The metalearning classifier outperforms the standard version in both overall F-measure and accuracy. However, due to the performance differences in the single classes, it is not clear whether this will lead to an improvement in terms of the financial performance.

We compare the performance of Bayes with metalearning classifier (Bayes binary), as one of the best performing settings shown in Table 5.11, with the performance of the *RandomLearner* on the training set (with 10-fold cross validation) and the independent test set. The results are depicted in Table 5.12. Although its overall accuracy drops from 55.8% on the training set to 48.2% on the test set, Bayes binary still outperforms the *RandomLearner* on both training and test set with respect to both overall F-measure and accuracy. On both training and test set, Bayes performs slightly worse with respect to the SELL class F-measure but considerably better than the *RandomLearner* with respect to the other classes.

Training	F_1^{buy}	F_1^{hold}	F_1^{sell}	F_1	A
Bayes binary	18.4%	72.1%	13.4%	34.6%	55.8%
Random	13.7%	47.5%	13.6%	24.9%	33.3%
Test					
Bayes binary	27.0%	63.4%	17.4%	35.9%	48.2%
Random	22.1%	45.2%	18.9%	28.7%	33.3%

Table 5.12: Evaluation with a final setting on the 3-class problem

5.4 Financial evaluation

In order to evaluate the trading system financially, we perform a market simulation by following the trading recommendations the classifier predicts. Based on our theoretical review (Section 2.1), we aim to make assumptions that are close to reality.

In the following, we describe the settings of the market simulation (Section 5.4.1). Then we present and discuss the results (Section 5.4.2).

5.4.1 Market simulation settings

We perform the market simulation in the time frame 7th May to 15th June 2012 (test period). We run the classifier model obtained after the parameter tuning in the last section on the unlabeled test set. As a result, we obtain a set of news labeled by the classifier. The labels are either BUY or SELL for the 2-class problem, and either BUY, HOLD or SELL for the 3-class problem.

At the start of the simulation, we assume an available budget of 500,000$. This amount is consistent with the one chosen by Mittermayer (2006). At the time a news article is published that is labeled BUY, we place a market order (see Section 2.1.1) to buy stocks of the associated company worth 500,000$. Analogously, we place a short sale order worth 500,000$ in case of a news article labeled SELL. In both cases, we liquidate the position 20 minutes after the news arrival. News articles labeled HOLD (in the 3-class problem) do not cause a trade. In accordance with Mittermayer (2006), we assume that the process of downloading the news message and the classifier prediction takes a maximum of 30 seconds. Therefore, we place an order with a delay of 30 seconds. We assume that this delay does not have a strong influence on the profits, since the delay is very short compared to the total holding period. For the sake of simplicity, we implicitly assume infinite divisibility of a share, i.e. we are able to buy or sell exact 500,000$ of a stock. As most shares trade for less than 500$, the trade prices in our model differ only in small fractions from the reality.

More importantly, we assume that we are able to buy 500,000$ worth of stock without moving the market. However, as described in Section 2.1.1, we would have to pay price impact costs. A possible way to calculate these costs is to consider the whole limit order book and reduce the profits by the price impact premium for each trade. Unfortunately, we do not have the complete limit order books available. Therefore, we assume fixed average price impact costs as discussed in Section 2.1.1. The trading costs are given in costs per round trip and are comprised of direct costs, costs caused by the bid-ask-spread and price impact costs. The total cost can be assumed to sum up to $c \approx 69$ bps $= 0.69\%$ for each round trip.[2] Li et al. (2011) propose the lower estimate of 30 bps.

For the stock price data we utilize the same data set we used during the training phase and transform it into price snapshots as described in Section 4.2.1. For each round trip, we calculate the profit R_b for a buy market order in the following fashion:

$$R_b = (\frac{P_{20}}{P_{0.5}} - 1 - c) \cdot 500,000.$$

[2]Since we look at S&P500 stocks, we adapt the trading costs for large caps estimated by ITG (2012): one-way costs of 34.4 bps, which is equivalent to round trip costs of $\approx$ 69 bps

$P_{0.5}$ is the stock price 30 seconds after news release (buy price) and P_{20} is the stock price 20 minutes after news release (sell price). For a short sale transaction the profit R_s is the following:

$$R_s = (-\frac{P_{20}}{P_{0.5}} + 1 - c) \cdot 500,000.$$

The *RandomLearner* and the *DefaultLearner* introduced in Section 5.3.1 can be used as a benchmark to evaluate the financial performance. In addition, we use a buy-and-hold strategy that buys the S&P 500 index at the opening price of the first day of the test period (7th May) and sells the index at the closing price of the last day of the test period (15th June). We do not take dividend yield into account, which is consistent to Allen & Karjalainen (1999); Becker & Seshadri (2003). Fama & Blume (1966) estimate dividend yield for the DJIA (Dow Jones Industrial Average) to be 0.016% per day, and Becker & Seshadri (2003) argue that it is less for the broader index S&P 500.

5.4.2 Results

We present the financial results of the classifier predictions of both the 2-class and the 3-class problem. For each problem, we evaluate settings that performed well during the parameter tuning (see Table 5.7 for the 2-class problem and Table 5.11 for the 3-class problem).

The results for the 2-class problem are shown in Figure 5.1. We compare the classifier performance to both *RandomLearner* and *DefaultLearner*. Since the *RandomLearner* places buy and short sale orders with the same likelihood, the profits for each independent trade are on average zero, leading to a total average of zero. We confirm this behavior by running the market simulation 500,000 times for the *RandomLearner* with the result of approximately zero profits. The *DefaultLearner* achieves a profit of 2.292 bps for each round trip or a total profit of 13.1%. This reflects the fact that in the test set the stock price on average tends to rise after a news article occurs. The profit R_{bh} achieved by the buy-and-hold strategy can be calculated as follows: $P_0 = 1,368.79$ is the S&P 500 opening level on 7th May and $P_1 = 1,342.84$ is the closing level on 15th June, then is $R_{bh} = P_1/P_0 - 1 = -1.9\%$.

Since the test set consists of 448 news articles and there is no HOLD label, the number of performed trades is 448 for all classifiers as well as for the *RandomLearner* and the *DefaultLearner*. All classifiers achieve a positive financial performance and thus outperform the *RandomLearner*. Despite the high performance of Bayes in terms of overall F-measure and the high overall accuracy of SVM RBF, these classifiers underperform the *DefaultLearner* financially. k-NN achieves

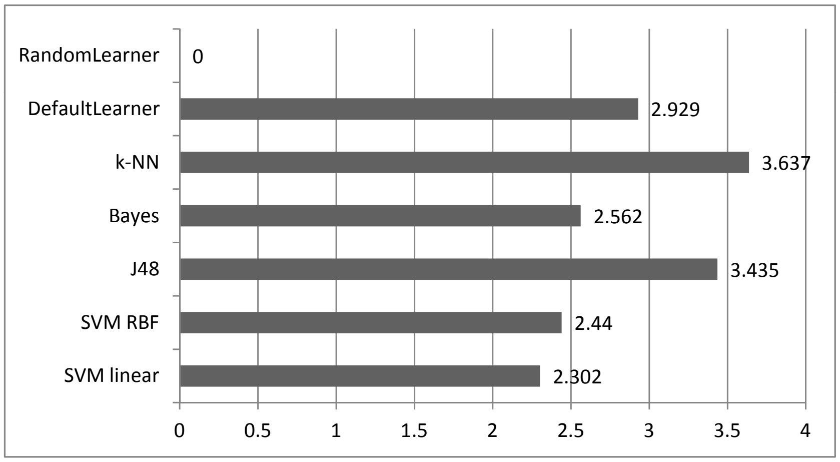


Figure 5.1: Financial performance of the classifiers in *bps per round trip* (2-class problem)

3.637 bps (16.3% total profit) per round trip and J48 achieves 3.435 bps per round trip (15.4% total profit), both classifiers outperforming the *DefaultLearner*.

In the following, we evaluate whether the results are statistically significant. First, we compare k-NN to the *DefaultLearner* approach using the paired t test described in Hsu & Lachenbruch (2008). Suppose that x_i is the profit for each trade achieved by k-NN, y_i is the profit for each trade achieved by the *DefaultLearner*, and their difference is $d_i = x_i - y_i$ for $i = 1, \ldots, n$, where $n = 448$ is the total number of trades performed. We assume these differences to be normally distributed, i.e. the random variable $D = X - Y$ is normally distributed. We further assume $D_i = X_i - Y_i$ to be independent. The mean μ_d and the variance σ_d^2 can be calculated as follows:

$$\mu_d = \mu_x - \mu_y = \frac{1}{n}\sum_{i=1}^{n} x_i - \frac{1}{n}\sum_{i=1}^{n} y_i$$

$$\sigma_d^2 = \frac{1}{n-1}\sum_{i=1}^{n}(d_i - \mu_d)^2.$$

Suppose the null hypothesis is $H_0 : \mu_x - \mu_y \leq 0$, i.e. the average profits of k-NN are not higher than the average profits of the *DefaultLearner*, and the alternative hypothesis is $H_1 : \mu_x - \mu_y > 0$. A Student's t statistic can then be calculated as

$$t = \sqrt{n} \cdot \frac{\mu_d}{\sigma_d}.$$

	μ_d (bps)	σ_d (bps)	df	t	P value	$P < 0.05$?
k-NN	0.707	15.550	447	0.963	0.168	no
J48	0.506	14.172	447	0.755	0.225	no

Table 5.13: Significance of profit differences to the *DefaultLearner*

We compare this statistic with the Student's t distribution with $n-1$ degrees of freedom (df), obtaining a P value that reflects how plausible the actual observations are under the assumption that the null hypothesis is true (Hubbard & Lindsay, 2008). The null hypothesis is typically rejected when the P value is below 0.05 or 0.01 (Blackwelder, 1982). The results for k-NN and J48 are shown in Table 5.13. We observe that the higher average profits compared to the *DefaultLearner* are not significant for both k-NN and J48.

Next, we test whether the profit differences of the classifiers compared to the *RandomLearner* are significant. We choose an approach similar to Groth & Muntermann (2009), who apply an unpaired *t* test for this task. We run the market simulation for the *RandomLearner* 10,000 times and thus obtain a sample of $m = 4,480,000$ trades each yielding a profit of y_i for $i = 1,\dots,m$. For the classifier (k-NN or J48), we obtain a sample of $n = 448$ trades, each yielding a profit of x_i for $i = 1,\dots,n$. We assume both samples to be independent from each other, i.e. the sample variables $X_i,\dots,X_n$ and $Y_i,\dots,Y_m$ are independent from each other. In addition, we assume X and Y to be normally distributed. We calculate the pooled variance

$$\sigma^2 = \frac{\text{df}_x \cdot \sigma_x^2 + \text{df}_y \cdot \sigma_y^2}{\text{df}_x + \text{df}_y},$$

with $\text{df}_x = m-1$ and $\text{df}_y = n-1$. Since we observe both samples to have similar variances σ_x^2 and σ_y^2, the Student's t statistic can be calculated as follows:

$$t = \sqrt{\frac{n \cdot m}{n+m}} \cdot \frac{\mu_x - \mu_y}{\sigma}.$$

We obtain a P value by comparing this statistic with a Student's t distribution with $\text{df} = \text{df}_x + \text{df}_y = 4,480,446$. The results are depicted in Table 5.14. Notice that in each test, the standard deviation of the *RandomLearner* σ_y remains 47.752, which results in $\sigma = 47.752$.[3] We conclude that none of the classifiers achieve significantly higher mean profits than the *RandomLearner* at the 0.05 level. However, the results of k-NN and J48 are significant at the 0.1 level.

[3]Since df_y is much larger than df_x, $\sigma_y \approx \sigma$ holds.

	σ_x (bps)	σ (bps)	t	P value	$P < 0.05$?
k-NN	47.666	47.752	1.617	0.053	no
Bayes	47.736	47.752	1.128	0.130	no
J48	47.681	47.752	1.523	0.064	no
SVM RBF	47.743	47.752	1.073	0.142	no
SVM linear	47.747	47.752	1.017	0.155	no

Table 5.14: Significance of profit differences to the *RandomLearner*

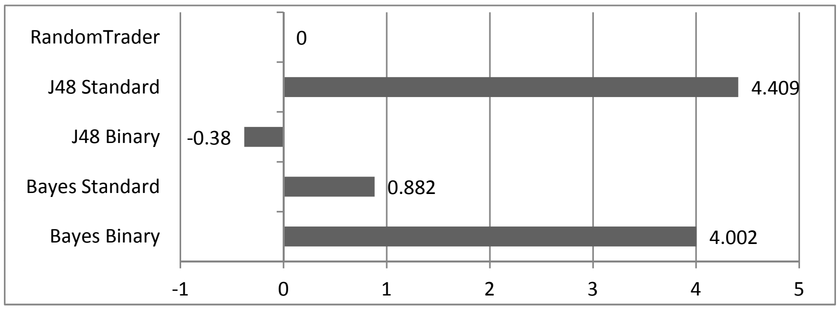


Figure 5.2: Financial performance of the classifiers in *bps per round trip* (3-class problem)

The results for the 3-class problem are depicted in Figure 5.2. We compare the settings J48 with SpreadSubsample and Bayes as they were able to outperform the *RandomLearner* with respect to all performance metrics (Section 5.3.2). We use the *RandomLearner* as a benchmark, again achieving a profit of zero, performing 299 trades. Notice that the *DefaultLearner* would also achieve a profit of zero in the 3-class problem since it would label all examples as HOLD, the majority class. J48 without the binary metalearning algorithm performs best with 4.409 bps each round trip or 15.2% total profit, performing 344 trades. However, when applying the binary metalearning algorithm, the profit drops slightly below zero (-0.38 bps with 240 trades, -0.9% total profit). The reason might be the comparatively low BUY class F-measure. Bayes achieves only 0.882 bps with 362 trades (3.2% total profit), but its performance increases when applying the binary metalearning algorithm to 4.002 bps with 208 trades (8.3% total profit). The binary metalearning algorithm seems to lead to a more conservative trading strategy, which causes a single trade

	σ_x (bps)	σ (bps)	df	t	P value	$P < 0.05$?
J48 SpreadS.	45.514	47.711	2,987,742	1.694	0.045	yes
Bayes	47.749	47.749	2,988,347	0.368	0.356	no
Bayes binary	46.367	47.764	2,985,797	1.214	0.112	no

Table 5.15: Significance of profit differences to the *RandomLearner* (3-class problem)

to be more profitable when training with Bayes.

In order to test whether these results are statistically significant, we use the unpaired t test described above to compare the classifiers with the *RandomLearner*. Since J48 binary performs worse than the *RandomLearner*, we do not take this setting into account. Since the 3-class problem includes HOLD examples, the degrees of freedom vary with the amount of trades performed. The results are shown in Table 5.15. The results of Bayes and Bayes binary are not significant. The result of J48 with SpreadSubsample is significant at the 0.05 level.

For both the 2-class and the 3-class problem, the classifiers are able to achieve a positive profit before trading costs (except for J48 with the binary metalearning algorithm). The classifiers outperform the buy-and-hold strategy in all settings. However, when taking into account trading costs estimated to either 69 bps or 30 bps per round trip, the profit drops below zero and all classifiers underperform the buy-and-hold strategy. These findings highlight the importance of testing the trading system under realistic assumptions. In Section 6.2, we will propose possibilities to further improve the system performance.

In Appendix B.B, we present the results of a set of additional experiments.

Chapter 6

Conclusion

Financial news articles are proven to influence stock prices and thus, many professional traders rely on newswire services as their major information source. However, the increasing amount of up-to-date textual information leads to a substantial information overload, making it harder for market participants to select the information relevant to them. Text mining as a means to analyze large amounts of textual data is a promising approach to address this problem. The goal of this thesis is to forecast short term stock price movements using text mining techniques.

In the following, we provide a summary of this thesis (Section 6.1). We then acknowledge the limitations of this thesis and propose ideas for further improvement (Section 6.2).

6.1 Summary

In Chapter 1, we provided an introduction to the thesis. In Chapter 2, we laid out the theoretical foundations important for the subsequent chapters. In Section 2.1, we focused on basic financial concepts. We discussed the different ways to place an order at the market, and mentioned trading costs as an issue that needs to be considered by investors aiming for profits at the stock market. We then looked at the different types and sources of financial news and identified newswire services as an important channel used to distribute regulated news. We discussed the efficient market hypothesis that questions the investors' ability of earning excess profits based on new information. We presented the event study methodology as a means to test the efficient market hypothesis. In Section 2.2, we introduced the relevant text mining techniques. We explained the concept of transforming a text into a feature vector representation and presented the features most important in this thesis. We then went through the process of text preprocessing, assigning different feature weights according to their importance, and eliminating less important features. We compared the most common classifiers, provided means to handle unbalanced data sets, and presented metrics to evaluate the prediction performance of a classifier.

In Chapter 3, we surveyed related systems that use text mining techniques with the aim of either forecasting security prices or price volatility. We extracted ideas and approaches that tended to be successful in the past, including using a support vector machine as a classifier, or restricting the diversity of news sources. We found potential areas for improvement, such as applying more complex sets of features or evaluating the system financially based on more realistic assumptions.

In Chapter 4, we presented our own stock price forecasting system. After illustrating the general design of the system, we described our approach for acquiring the news and stock price data. We extracted the needed information such as the publishing time, the headline, and the text body from the news articles. We then filtered out less price relevant news articles and converted the news articles into a suitable format for the subsequent steps. Next, we labeled the news automatically, extracted different feature sets from the news, addressed the issue of unbalanced data sets, and used the feature sets to train a classifier that is able to predict labels for an independent test set of news articles.

In Chapter 5, we evaluated the system performance for two different data sets (2-class problem and 3-class problem), both resulting from different labeling approaches. We compared different system characteristics such as the feature set used, the method for handling unbalanced data, and the classifier trained with the aim of improving the prediction performance. Complex features such as part of speech categories and named entities do not improve the prediction accuracy compared to the bag-of-words approach. However, the bigram representation leads to an improved prediction performance and thus might be able to capture the relevant text semantics more precisely. Given the data set is not highly unbalanced, we found that the SMOTE oversampling approach outperforms the SpreadSubsample undersampling approach. The classifier SVM is frequently described as superior in the text classification task. However, our experiments suggested that for the 3-class problem, the Naïve Bayes and the decision tree were the only classifiers able to outperform a random baseline with respect to all performance metrics.

Finally, we presented a market simulation based on assumptions close to reality in order to evaluate the system performance financially. We concluded that positive profits are achievable by all classifiers in the 2-class problem. In the 3-class problem, we observed positive profits for three of the four evaluated settings. However, when taking realistic trading costs into account, the profits are likely to drop below zero. In the next section, we will discuss possibilities for future work that may have the potential to further improve the performance.

6.2 Future work

The right choice of input data is one of the most important parts of a trading system such as the one developed in this thesis. More specifically, the likelihood of the stock prices being influenced by the news articles used for training, should be as high as possible. It can be assumed that this leads to less noise in the data and therefore to a better prediction performance. Apart from the approaches presented in this thesis (see Section 4.2.1) there are further ways to achieve this. First, the trading volume can be taken into account. It has been shown that the stock's trading volume is related to the announcement of relevant news (e.g. Kim & Verrecchia, 1991; Ryan & Taffler, 2004). Therefore, only news articles followed by a significant increase in trading volume (compared to the stock's average trading volume) can be used for training. Second, only news articles from a certain domain can be taken into account since the text characteristics responsible for changes in the stock price might vary from domain to domain. For instance, one can consider only news articles dealing with the patent business of companies in the pharmaceutical sector. Although this limits the scope of the results, it potentially results in a more precise prediction model. However, both of these approaches require a much larger data corpus (e.g. 2 years intraday price data and news data) than the one used in this thesis (17 weeks).

Some of the systems reviewed in this thesis predict stock price volatility. A recent example is the system of Robertson et al. (2006). This system is based on a model specifically suitable for the task of volatility prediction, i.e. the expected price fluctuation of a stock. A possible method that might increase the financial performance is to train two separate classifiers. Consider a two layer approach in the classification phase. The first classifier predicts the expected volatility and thus the expected stock price relevance of a published news article. Only if the news article passes this relevance check will it be given to a classifier that predicts the actual stock price movement (UP or DOWN).

The automatic labeling approach has improvement potential. In general, the higher the absolute value of the threshold used for labeling the news BUY or SELL, the more likely it is that the news is responsible for the price trend. Increasing the threshold is therefore beneficial. However, it in turn increases the size of the HOLD class, making the data set more unbalanced. With more input data available this issue can be simply addressed using undersampling, which led in this thesis in most settings to poor prediction quality partly due to the small size of data left. Due to the limited resources available in this thesis, we assessed the effectiveness of the automatic labeling approach by manually labeling a random sample. Taking this approach one step further, a team of domain experts can independently label all training instances. Whenever all domain experts agree on either a label BUY or SELL, the training set is labeled accordingly. In case of disagreement, the label UNCERTAIN is assigned. The resulting training set can serve as a gold standard for the training set labeled automatically. Surprisingly, none of the systems reviewed in Chapter 3 attempt to establish such a gold standard.

When evaluating the classifier prediction performance, we observed the issue of the labeled data being unbalanced. SMOTE is one of the methods we used to address it. However, in case of strongly unbalanced data, SMOTE encounters an overgeneralization problem since it does not consider the majority class when creating synthetic examples (Bunkhumpornpat et al., 2009). Different attempts have been made to improve SMOTE with respect to this problem (e.g. Bunkhumpornpat et al., 2009; Ramentol et al., 2012; Stefanowski & Wilk, 2008) which may increase the prediction performance.

In the market simulation, we calculated the profits and considered the estimated trading costs independently including the price impact costs (see Section 2.1.1). Since a limited number of shares is available for the best price and a premium must be paid for each additional share an investor wants to buy, a more realistic way to calculate the price impact costs is to take the whole limit order book into account. This approach would require the order book data for all trades in the simulation time.

We believe that by tapping the full potential of text mining in the financial domain there are many golden nuggets yet to be discovered. We hope that the contributions presented in this thesis will help future research towards improving the quality of trading decisions powered by text mining techniques.

Chapter 7

Bibliography

Adler, P. A. & Adler, P. (1984). *The Social Dynamics of Financial Markets*, chapter The market as collective behavior, (pp. 85–105). Jai Press.

Agrawal, R., Bayardo, R., & Srikant, R. (2000). Athena: Mining-based interactive management of text databases. In C. Zaniolo, P. Lockemann, M. Scholl, & T. Grust (Eds.), *Advances in Database Technology*, volume 1777 of *Lecture Notes in Computer Science* (pp. 365–379). Berlin, Heidelberg: Springer.

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6, 37–66.

Ahn, H.-J., Bae, K.-H., & Chan, K. (2001). Limit orders, depth, and volatility: Evidence from the stock exchange of Hong Kong. *The Journal of Finance*, 56(2), 767–788.

Allen, F. & Karjalainen, R. (1999). Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, 51(2), 245–271.

Asahara, M. & Matsumoto, Y. (2003). Japanese named entity extraction with redundant morphological analysis. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1* (pp. 8–15). Stroudsburg: Association for Computational Linguistics.

Baeza-Yates, R. A. & Ribeiro-Neto, B. (2011). *Modern Information Retrieval*, chapter Modeling, (pp. 57–130). Pearson Education: Boston.

Becker, L. & Seshadri, M. (2003). GP-evolved technical trading rules can outperform buy and hold. In *Procceedings of the Sixth International Conference on Computational Intelligence and Natural Computing* Cary, North Carolina.

Ben-Hur, A. & Weston, J. (2010). A user's guide to support vector machines. *Data Mining Techniques for the Life Sciences*, 609, 223–239.

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

Bessembinder, H. & Venkataraman, K. (2010). *Encyclopedia of Quantitative Finance*, chapter Bid–ask spreads. John Wiley & Sons: Hoboken, New Jersey.

Blackwelder, W. C. (1982). "Proving the null hypothesis" in clinical trials. *Controlled Clinical Trials*, 3(4), 345–353.

Bloomberg (2012). Bloomberg professional. `http://www.bloomberg.com/professional/`. Accessed on 12/07/30.

Blumer, H. (1975). *Readings in Collective Behavior*, chapter Outline of collective behavior, (pp. 22–45). Rand McNally College Publishing.

Bodie, Z., Kane, A., & Marcus, A. J. (1989). *Investments*. New York: Irwin Professional Publishing.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3), 307–327.

Borthwick, A., Sterling, J., Agichtein, E., & Grishman, R. (1998). NYU: Description of the MENE named entity system as used in MUC-7. In *Proceedings of the Seventh Message Understanding Conference* San Francisco: Morgan Kaufmann.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (pp. 144–152). New York: ACM.

Brücher, H., Knolmayer, G., & Mittermayer, M.-A. (2002). Document classification methods for organizing explicit knowledge. In *Proceedings of the Third European Conference on Organizational Knowledge, Learning, and Capabilities* Athens.

Brealey, R. A., Myers, S. C., & Allen, F. (2011). *Principles of corporate finance*, chapter Payout policy, (pp. 419–445). McGraw-Hill/Irwin: New York, 10. edition.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2), 123–140.

Bunkhumpornpat, C., Sinapiromsaran, K., & Lursinsap, C. (2009). Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In T. Theeramunkong, B. Kijsirikul, N. Cercone, & T.-B. Ho (Eds.), *Advances in Knowledge Discovery and Data Mining*, volume 5476 of *Lecture Notes in Computer Science* (pp. 475–482). Berlin, Heidelberg: Springer.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2, 121–167.

Chan, W. S. (2003). Stock price reaction to news and no-news: Drift and reversal after headlines. *Journal of Financial Economics*, 70(2), 223–260.

Chang, C.-C. & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2, 1–27.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357.

Chawla, N. V., Cieslak, D. A., Hall, L. O., & Joshi, A. (2008). Automatically countering imbalance and its empirical relationship to cost. *Data Mining and Knowledge Discovery*, 17(2), 225–252.

Chen, C., Liaw, A., & Breiman, L. (2004). *Using random forest to learn unbalanced data*. Technical Report 666, University of California, Department of Statistics, Berkeley.

Chinchor, N. A. (1998). Named entity task definition. In *Proceedings of the Seventh Message Understanding Conference* San Francisco: Morgan Kaufmann.

Cho, V. & Wüthrich, B. (1998). Towards real time discovery from distributed information sources. In *Proceedings of the Second Pacific-Asia Conference on Research and Development in Knowledge Discovery and Data Mining* (pp. 376–377). London: Springer.

Cho, V., Wüthrich, B., & Zhang, J. (1999). Text processing for classification. *Journal of Computational Intelligence in Finance*, 7(2), 6–22.

Cohen, W. W. (1996). Learning to classify english text with ILP methods. In L. De Raedt (Ed.), *Advances in Inductive Logic Programming* (pp. 124–143). IOS Press.

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297.

Cover, T. & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1), 21–27.

Cunningham, H., Maynard, D., Bontcheva, K., & Tablan, V. (2002). GATE: An architecture for development of robust HLT applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics* (pp. 168–175). Stroudsburg: Association for Computational Linguistics.

Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Aswani, N., Roberts, I., Gorrell, G., Funk, A., Roberts, A., Damljanovic, D., Heitz, T., Greenwood, M. A., Saggion, H., Petrak, J., Li, Y., & Peters, W. (2011). Text processing with GATE (version 6). University of Sheffield, Department of Computer Science. `http://gate.ac.uk/releases/gate-6.1-build3913-ALL/tao.pdf`. Accessed on 12/08/23.

Cunningham, H., Maynard, D., & Tablan, V. (2000). *JAPE: A Java Annotation Patterns Engine (Second Edition)*. Technical report CS–00–10, University of Sheffield, Department of Computer Science.

Davis, A. K., Piger, J. M., & Sedor, L. M. (2012). Beyond the numbers: Measuring the information content of earnings press release language. *Contemporary Accounting Research*, 29(3), 845–868.

Demsetz, H. (1968). The cost of transacting. *The Quarterly Journal of Economics*, 82(1), 33–53.

Deutsche Börse (2012). The open Xetra order book. 10 best bid and ask quotes with volume. `http://www.boerse-frankfurt.de/en/aktien/xetra-orderbuch#tab_id=dax`. Accessed on 12/07/23.

Domingos, P. (1999). MetaCost: A general method for making classifiers cost-sensitive. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 155–164). New York: ACM.

Dörre, J., Gerstl, P., & Seiffert, R. (1999). Text mining: Finding nuggets in mountains of textual data. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 398–401). New York: ACM.

Duda, R. O. & Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. New York: John Wiley & Sons.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern Classification*, chapter Nonparametric techniques, (pp. 161–214). John Wiley & Sons: New York.

Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics - Special issue on using large corpora: I*, 19(1), 61–74.

Edelen, R. M., Evans, R. B., & Kadlec, G. B. (2007). Scale effects in mutual fund performance: The role of trading costs. Working Paper. SSRN.

EL-Manzalawy, Y. & Honavar, V. (2005). *WLSVM: Integrating LibSVM into Weka Environment*. Software available at `http://www.cs.iastate.edu/~yasser/wlsvm`. Accessed on 12/10/20.

Elton, E. J., Gruber, M. J., Brown, S. J., & Groetzmann, W. N. (2011a). *Modern Portfolio Theory and Investment Analysis*, chapter Financial markets, (pp. 11–27). John Wiley & Sons: New York, 5. edition.

Elton, E. J., Gruber, M. J., Brown, S. J., & Groetzmann, W. N. (2011b). *Modern Portfolio Theory and Investment Analysis*, chapter Efficient markets, (pp. 396–437). John Wiley & Sons: New York, 5. edition.

Fama, E. F. (1965). The behavior of stock-market prices. *The Journal of Business*, 38(1), 34–105.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2), 383–417.

Fama, E. F. (1991). Efficient capital markets: II. *The Journal of Finance*, 46(5), 1575–1617.

Fama, E. F. & Blume, M. E. (1966). Filter rules and stock-market trading. *The Journal of Business*, 39(1), 226–241.

Fan, W., Wallace, L., Rich, S., & Zhang, Z. (2006). Tapping the power of text mining. *Communications of the ACM*, 49(9), 76–82.

Farhoomand, A. F. & Drury, D. H. (2002). Managerial information overload. *Communications of the ACM*, 45(10), 127–131.

Fawcett, T. & Provost, F. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1, 291–316.

Fawcett, T. & Provost, F. (1999). Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 53–62). New York: ACM.

Fayyad, U., Piatetsky-shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, 17, 37–54.

Feldman, R. & Sanger, J. (2006). *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. New York: Cambridge University Press.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3, 1289–1305.

Frawley, W. J., Piatetsky-shapiro, G., & Matheus, C. J. (1992). Knowledge discovery in databases: An overview. *AI Magazine*, 13(3), 57–70.

French, K. R. (2008). Presidential address: The cost of active investing. *The Journal of Finance*, 63(4), 1537–1573.

Fung, G., Xu Yu, J., & Lam, W. (2003). Stock prediction: Integrating text mining approach using real-time news. In *IEEE International Conference on Computational Intelligence for Financial Engineering* (pp. 395–402). Hong Kong: IEEE.

Fung, G., Yu, J., & Lam, W. (2002). News sensitive stock trend prediction. In M.-S. Chen, P. Yu, & B. Liu (Eds.), *Advances in Knowledge Discovery and Data Mining*, volume 2336 of *Lecture Notes in Computer Science* (pp. 481–493). Berlin, Heidelberg: Springer.

Fung, G., Yu, J. X., & Lu, H. (2005). The predicting power of textual information on financial markets. *IEEE Intelligent Informatics Bulletin*, 5(1), 1–10.

Gerstl, P., Hertweck, M., & Kuhn, B. (2002). Text Mining: Grundlagen, Verfahren und Anwendungen. *HMD – Praxis der Wirtschaftsinformatik*, 222, 38–48.

Gidófalvi, G. (2001). *Using News Articles to Predict Stock Price Movements*. Technical report, University of California, Department of Computer Science and Engineering, San Diego.

Gidófalvi, G. & Elkan, C. (2003). *Using News Articles to Predict Stock Price Movements*. Technical report, University of California, Department of Computer Science and Engineering, San Diego.

Gonçalves, M. (2011). *Modern Information Retrieval*, chapter Text classification. Pearson Education: Boston.

Gonçalves, T. & Quaresma, P. (2005). Enhancing a Portuguese text classifier using part-of-speech tags. In M. Kłopotek, S. Wierzchoń, & K. Trojanowski (Eds.), *Intelligent Information Processing and Web Mining*, volume 31 of *Advances in Soft Computing* (pp. 189–198). Berlin, Heidelberg: Springer.

Grossman, S. J. & Stiglitz, J. E. (1980). On the impossibility of informationally efficient markets. *The American Economic Review*, 70(3), 393–408.

Groth, S. & Muntermann, J. (2008). A text mining approach to support intraday financial decision-making. In *Proceedings of the Fourteenth Americas Conference on Information Systems* Toronto.

Groth, S. S. (2010). Enhancing automated trading engines to cope with news-related liquidity shocks. In *Eighteenth European Conference on Information Systems* Pretoria.

Groth, S. S. & Muntermann, J. (2009). Supporting investment management processes with machine learning techniques. In H. R. Hansen, D. Karagiannis, & H.-G. Fill (Eds.), *Business Services: Konzepte, Technologien, Anwendungen. Neunte Internationale Tagung Wirtschaftsinformatik*, volume 247 (pp. 275–286). Wien: Österreichische Computer Gesellschaft.

Groth, S. S. & Muntermann, J. (2011). An intraday market risk management approach based on textual analysis. *Decision Support Systems*, 50(4), 680–691.

Hagenau, M., Liebmann, M., Hedwig, M., & Neumann, D. (2012). Automated news reading: Stock price prediction based on financial news using context-specific features. In *45th Hawaii International Conference on System Science* (pp. 1040–1049). Maui: IEEE.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1), 10–18.

Harris, L. (2003). *Trading and exchanges. Market Microstructure for Practitioners*, chapter Orders and order properties. Financial Management Association. Oxford University Press: New York.

Hatzivassiloglou, V. & McKeown, K. R. (1997). Predicting the semantic orientation of adjectives. In *Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics* (pp. 174–181). Stroudsburg: Association for Computational Linguistics.

Henry, E. (2008). Are investors influenced by how earnings press releases are written? *Journal of Business Communication*, 45(4), 363–407.

Hepple, M. (2000). Independence and commitment: Assumptions for rapid training and execution of rule-based POS taggers. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics* (pp. 278–277). Stroudsburg: Association for Computational Linguistics.

Herrmann, K. (2002). Rakesh Agrawal: Athena: mining-based interactive management of text databases. Hauptseminar Informatik "Database Hall of Fame", Technische Universität München. `http://wwwbayer.in.tum.de/lehre/WS2001/HSEM-bayer/textmining.pdf`. Accessed on 12/06/04.

Hobbs, J. R. & Riloff, E. (2010). Information extraction. In N. Indurkhya & F. J. Damerau (Eds.), *Handbook of Natural Language Processing*. Boca Raton: CRC Press, Taylor and Francis Group, 2. edition.

Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (2010). *A Practical Guide to Support Vector Classification*. Technical Report 1, National Taiwan University, Department of Computer Science, Taipei.

Hsu, H. & Lachenbruch, P. A. (2008). *Wiley Encyclopedia of Clinical Trials*, chapter Paired t test. John Wiley & Sons: Hoboken, New Jersey.

Hubbard, R. & Lindsay, R. (2008). Why p values are not a useful measure of evidence in statistical significance testing. *Theory & Psychology*, 18(1), 69–88.

Hull, D. A. (1998). Stemming algorithms: A case study for detailed evaluation. *Journal of the American Society for Information Science*, 47(1), 70–84.

ITG (2012). ITG's global cost review 2012/Q1. Final results as of 7/11/2012. `http://itg.com/news_events/papers/ITGGlobalCostReview_2012Q1.pdf`. Accessed on 12/07/28.

Jain, A. & Chandrasekaran, B. (1982). 39 dimensionality and sample size considerations in pattern recognition practice. In P. Krishnaiah & L. Kanal (Eds.), *Classification Pattern Recognition and Reduction of Dimensionality*, volume 2 of *Handbook of Statistics* (pp. 835–855). New York: Elsevier Science.

Jensen, M. C. (1978). Some anomalous evidence regarding market efficiency. *Journal of Financial Economics*, 6(2-3), 95–101.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In C. Nédellec & C. Rouveirol (Eds.), *Proceedings of the Tenth European Conference on Machine Learning*, volume 1398 of *Lecture Notes in Computer Science* (pp. 137–142). Berlin, Heidelberg: Springer.

Joachims, T. (1999). *Making Large-Scale Support Vector Machine Learning Practical*, (pp. 169–184). MIT Press: Cambridge.

Kaserer, C. & Nowak, E. (2001). Die Anwendung von Ereignisstudien bei Ad-hoc-Mitteilungen. Zugleich Stellungnahme zu dem Beitrag "Die Informationswirkung von Ad hoc-Meldungen" von Klaus Röder. *Zeitschrift für Betriebswirtschaft*, 71(11), 1353–1356.

Kim, O. & Verrecchia, R. E. (1991). Trading volume and price reactions to public announcements. *Journal of Accounting Research*, 29(2), 302–321.

Kissell, R. (2006). The expanded implementation shortfall: "Understanding transaction cost components". *The Journal of Trading*, 1(3), 6–16.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial intelligence*, volume 2 (pp. 1137–1143). San Francisco: Morgan Kaufmann.

Krovetz, R. (1993). Viewing morphology as an inference process. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 191–202). New York: ACM.

Kubat, M. & Matwin, S. (1997). Addressing the curse of imbalanced training sets: One-sided selection. In D. H. Fisher (Ed.), *Proceedings of the Fourteenth International Conference on Machine Learning* (pp. 179–186). Burlington: Morgan Kaufmann.

Lam, W., Low, K. F., & Ho, C. Y. (1997). Using a Bayesian network induction approach for text categorization. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence* (pp. 745–750). San Francisco: Morgan Kaufmann.

Lavrenko, V., Lawrie, D., Ogilvie, P., & Schmill, M. (1999). Ænalyst - electronic analyst of stock behavior. CmpSci 791 m Project Draft. University of Massachusetts, Department of Computer Science, Amherst.

Lavrenko, V., Schmill, M., Lawrie, D., Ogilvie, P., Jensen, D., & Allan, J. (2000a). Language models for financial news recommendation. In *Proceedings of the Ninth International Conference on Information and Knowledge Management* (pp. 389–396). New York: ACM.

Lavrenko, V., Schmill, M., Lawrie, D., Ogilvie, P., Jensen, D., & Allan, J. (2000b). Mining of concurrent text and time series. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 37–44). New York: ACM.

Lee, A. J. T., Lin, M.-C., Kao, R.-T., & Chen, K.-T. (2010). An effective clustering approach to stock market prediction. In *Proceedings of the Fourteenth Pacific Asia Conference on Information Systems* Taipei: Association for Information Systems. Paper 54.

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). RCV1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5, 361–397.

LexisNexis (2012a). About LexisNexis. `http://www.lexisnexis.com/en-us/about-us/about-us.page`. Accessed on 12/07/29.

LexisNexis (2012b). Welcome to the LexisNexis® Web Services Kit. `http://www.lexisnexis.com/webserviceskit/`. Accessed on 12/07/30.

Li, X., Wang, C., Dong, J., Wang, F., Deng, X., & Zhu, S. (2011). Improving stock market prediction by integrating both market news and stock prices. In A. Hameurlain, S. Liddle, K.-D. Schewe, & X. Zhou (Eds.), *Database and Expert Systems Applications*, volume 6861 of *Lecture Notes in Computer Science* (pp. 279–293). Berlin, Heidelberg: Springer.

Lin, M.-C., Lee, A. J. T., Kao, R.-T., & Chen, K.-T. (2011). Stock price movement prediction using representative prototypes of financial reports. *ACM Transactions on Management Information Systems*, 2(3), 1–18.

Lintner, J. (1965). The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets. *The Review of Economics and Statistics*, 47(1), 13–37.

Luhn, H. P. (1957). A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4), 309–317.

MacKinlay, A. C. (1997). Event studies in economics and finance. *Journal of Economic Literature*, 35(1), 13–39.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. New York: Cambridge University Press.

Manning, C. D. & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. Cambridge: MIT Press.

Marcus, M. P., Marcinkiewicz, M. A., & Santorini, B. (1993). Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics - Special Issue on Using Large Corpora: II*, 19(2), 313–330.

Marrero, M., Sánchez-Cuadrado, S., Lara, J. M., & Andreadakis, G. (2009). Evaluation of named entity extraction systems. *Advances in Computational Linguistics. Research in Computing Science*, 41, 47–58.

Massy, W. F. (1965). Discriminant analysis of audience characteristics. *Journal of Advertising Research*, 5, 39–48.

McCallum, A. & Nigam, K. (1998). A comparison of event models for Naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization* (pp. 41–48).: AAAI Press.

Mercer, M. (2004). How do investors assess the credibility of management disclosures? *Accounting Horizons*, 18(3), 185–196.

Michie, D., Spiegelhalter, D. J., & Taylor, C. (1994). *Machine Learning, Neural and Statistical Classification*. Upper Saddle River, New Jersey: Ellis Horwood.

Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., & Euler, T. (2006). YALE: rapid prototyping for complex data mining tasks. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 935–940). New York: ACM.

Mikheev, A., Moens, M., & Grover, C. (1999). Named entity recognition without gazetteers. In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics* (pp. 1–8). Stroudsburg: Association for Computational Linguistics.

Mitchell, T. (1997). *Machine Learning*. New York: McGraw-Hill Education (ISE Editions).

Mitra, L. & Mitra, G. (2011). *The Handbook of News Analytics in Finance*, chapter Applications of news analytics in finance: A review, (pp. 1–39). John Wiley & Sons: Hoboken, New Jersey.

Mittermayer, M.-A. (2004). Forecasting intraday stock price trends with text mining techniques. In R. H. Sprague (Ed.), *Proceedings of the 37th Annual Hawaii International Conference on System Sciences* Big Island: IEEE Computer Society.

Mittermayer, M.-A. (2006). *Einsatz von Text Mining zur Prognose kurzfristiger Trends von Aktienkursen nach der Publikation von Unternehmensnachrichten*. Berlin: dissertation.de.

Mittermayer, M.-A. & Knolmayer, G. (2006a). NewsCATS: A news categorization and trading system. In *Sixth International Conference on Data Mining* (pp. 1002–1007). Washington, D.C.: IEEE Computer Society.

Mittermayer, M.-A. & Knolmayer, G. (2006b). Text mining systems for market response to news: A survey. Working Paper. Universität Bern, Institut für Wirtschaftsinformatik, Bern.

Morningstar (2011). The Morningstar category classifications (for portfolios available for sale in the United States). Morningstar methodology paper. `http://corporate.morningstar.com/fr/documents/MethodologyDocuments/MethodologyPapers/MorningstarCategory_Classifications.pdf`. Accessed on 12/07/25.

Moschitti, R. & Basili, R. (2004). Complex linguistic features for text classification: A comprehensive study. In *Proceedings of the 26th European Conference on Information Retrieval* (pp. 181–196). Berlin, Heidelberg: Springer.

Mucklow, B. (1991). Logarithmic versus proportional returns: A note. Working Paper. University of Wisconsin.

Mucklow, B. (1994). Market microstructure: An examination of the effects on intraday event studies. *Contemporary Accounting Research*, 10(2), 355–382.

Munz, M. (2011). *The Handbook of News Analytics in Finance*, chapter Measuring the value of media sentiment: A pragmatic view, (pp. 109–128). John Wiley & Sons: Hoboken, New Jersey.

Nadeau, D. & Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1), 3–26.

Nadeau, D., Turney, P., & Matwin, S. (2006). Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In L. Lamontagne & M.

Marchand (Eds.), *Advances in Artificial Intelligence*, volume 4013 of *Lecture Notes in Computer Science* (pp. 266–277). Berlin, Heidelberg: Springer.

NASDAQ (2012). Nasdaq trading schedule. `http://www.nasdaq.com/about/trading-schedule.aspx`. Accessed on 12/07/31.

Navarro, G. & Ziviani, N. (2011). *Modern Information Retrieval*, chapter Documents: Languages & properties, (pp. 203–254). Pearson Education: Boston.

NYSE (2012). New York stock exchange trading hours and holidays. `http://www.nyx.com//en/holidays-and-hours/nyse`. Accessed on 12/07/31.

Odean, T. (1999). Do investors trade too much? *The American Economic Review*, 89(5), 1279–1298.

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, volume 10 (pp. 79–86). Stroudsburg: Association for Computational Linguistics.

Pavlidis, T. & Horowitz, S. (1974). Segmentation of plane curves. *IEEE Transactions on Computers*, C-23(8), 860–870.

Peramunetilleke, D. & Wong, R. K. (2002). Currency exchange rate forecasting from news headlines. In *Proceedings of the Thirteenth Australasian Database Conference*, volume 5 (pp. 131–139). Darlinghurst: Australian Computer Society.

Phung, Y.-C. (2005). Text mining for stock movement predictions: A Malaysian perspective. In A. Zanasi, C. Brebbia, & N. Ebecken (Eds.), *Data Mining VI. Data Mining, Text Mining And Their Business Applications* (pp. 103–112). Southampton: WIT Press.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, 14(3), 130–137.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1), 81–106.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Francisco: Morgan Kaufmann Publishers.

Ramentol, E., Verbiest, N., Bello, R., Caballero, Y., Cornelis, C., & Herrera, F. (2012). SMOTE-FRST – a new resampling method using fuzzy rough set theory. In C. Kahraman, F. T. Bozbura, & E. E. Kerre (Eds.), *Uncertainty Modeling in Knowledge Engineering and Decision Making. Proceedings of the Tenth International Flins Conference* Istanbul: World Scientific.

Renehan, E. J. (2004). Robber baron. *American Heritage*, 55(5). `http://www.americanheritage.com/content/robber-baron-0`. Accessed on 12/10/20.

Robertson, C., Geva, S., & Wolff, R. C. (2006). What types of events provide the strongest evidence that the stock market is affected by company specific news? In *Proceedings of the Fifth Australasian Conference on Data Mining and Analystics*, volume 61 (pp. 145–153). Darlinghurst: Australian Computer Society.

Robertson, C., Geva, S., & Wolff, R. C. (2007a). Can the content of public news be used to forecast abnormal stock market behaviour? In *Proceedings of the Seventh IEEE International Conference on Data Mining* (pp. 637–642). Washington, D.C.: IEEE Computer Society.

Robertson, C., Geva, S., & Wolff, R. C. (2007b). The intraday effect of public information: Empirical evidence of market reaction to asset specific news from the US, UK, and Australia. SSRN Working Paper Series. `http://ssrn.com/abstract=970884`. Accessed on 12/07/22.

Robertson, C., Geva, S., & Wolff, R. C. (2007c). News aware volatility forecasting: is the content of news important? In *Proceedings of the Sixth Australasian Conference on Data Mining and Analytics*, volume 70 (pp. 161–170). Darlinghurst: Australian Computer Society.

Robertson, C. S. (2008). *Real time financial information analysis*. PhD thesis, Queensland University of Technology.

Rosé, C., Wang, Y.-C., Cui, Y., Arguello, J., Stegmann, K., Weinberger, A., & Fischer, F. (2008). Analyzing collaborative learning processes automatically: Exploiting the advances of computational linguistics in computer-supported collaborative learning. *International Journal of Computer-Supported Collaborative Learning*, 3, 237–271.

Russell, S. J. & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*, chapter Learning from examples, (pp. 693–767). Prentice-Hall series in artificial intelligence. Prentice-Hall: Upper Saddle River, New Jersey, 3. edition.

Ryan, P. & Taffler, R. J. (2004). Are economically significant stock returns and trading volumes driven by firm-specific news releases? *Journal of Business Finance & Accounting*, 31(1-2), 49–82.

Salton, G. & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5), 513–523.

Salton, G. & McGill, M. J. (1986). *Introduction to Modern Information Retrieval*. New York: McGraw-Hill.

Schulz, A., Spiliopoulou, M., & Winkler, K. (2003). Kursrelevanzprognose von Ad-hoc-Meldungen: Text Mining wider die Informationsüberlastung im Mobile Banking. In *Wirtschaftinformatik Proceedings 2003*. Paper 63.

Schumaker, R. & Chen, H. (2006). Textual analysis of stock market prediction using financial news articles. In *Proceedings of Twelfth Americas Conference on Information Systems*: AIS Electronic Library.

Schumaker, R. & Chen, H. (2010). A discrete stock price prediction engine based on financial news. *Computer*, 43(1), 51–56.

Schumaker, R., Zhang, Y., & Huang, C. (2012a). Evaluating sentiment in financial news articles. *Communications of International Information Management Association*, 53, 458–464.

Schumaker, R. P. & Chen, H. (2008). Evaluating a news-aware quantitative trader: The effect of momentum and contrarian stock selection strategies. *Journal of the American Society for Information Science and Technology*, 59(2), 247–255.

Schumaker, R. P. & Chen, H. (2009). Textual analysis of stock market prediction using breaking financial news: The AZFin text system. *ACM Transactions on Information Systems*, 27(2), 1–19.

Schumaker, R. P., Zhang, Y., Huang, C.-N., & Chen, H. (2012b). Evaluating sentiment in financial news articles. *Decision Support Systems*, 53(3), 458–464.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1), 1–47.

SEC (2000). Final rule: Selective disclosure and insider trading. `http://www.sec.gov/rules/final/33-7881.htm`. Accessed on 12/06/14.

SEC (2007). "SEC Fee" — Section 31 Transaction Fees. `http://www.sec.gov/answers/sec31.htm`. Accessed on 12/08/17.

SEC (2008). Commission guidance on the use of company web sites. `http://www.sec.gov/rules/interp/2008/34-58288.pdf`. Accessed on 12/06/14.

SEC (2009). Self-regulatory organizations; notice of filing and immediate effectiveness of proposed rule change by New York stock exchange LLC amending the exchange's timely alert policy. `http://www.sec.gov/rules/sro/nyse/2009/34-59823.pdf`. Accessed on 12/06/11.

Sekine, S. & Nobata, C. (2003). Definition, dictionaries and tagger for extended named entity hierarchy. In *Proceedings of the International Conference on Language Resources and Evaluation*.

Sharpe, W. F. (1963). A simplified model for portfolio analysis. *Management Science*, 9(2), 277–293.

Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance*, 19(3), 425–442.

Snell, A. & Tonks, I. (2003). A theoretical analysis of institutional investors' trading costs in auction and dealer markets. *The Economic Journal*, 113(489), 576–597.

Spiliopoulou, M., Schulz, A., & Winkler, K. (2003). Text Mining an der Börse: Einfluss von Ad-hoc-Mitteilungen auf die Kursentwicklung. In C. Becker & H. Redlich (Eds.), *Data Mining und Statistik in Hochschule und Wirtschaft. Proceedings der 7. Konferenz der SAS-Anwender in Forschung und Entwicklung (KSFE)* (pp. 215–228). Aachen: Shaker.

Stefanowski, J. & Wilk, S. (2008). Selective pre-processing of imbalanced data for improving classification performance. In *Proceedings of the Tenth International Conference on Data Warehousing and Knowledge Discovery* (pp. 283–292). Berlin, Heidelberg: Springer.

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2), 267–307.

Tay, F. E. H. & Cao, L. (2001). Application of support vector machines in financial time series forecasting. *Omega*, 29(4), 309–317.

Terada, A. & Tokunaga, T. (2003). Corpus based method of transforming nominalized phrases into clauses for text mining application. *IEICE Transactions on Information and Systems*, 86(9), 1736–1744.

Tetlock, P. C. (2007). Giving content to investor sentiment: The role of media in the stock market. *The Journal of Finance*, 62(3), 1139–1168.

Tetlock, P. C., Saar-Tsechansky, M., & Macskassy, S. (2008). More than words: Quantifying language to measure firms' fundamentals. *The Journal of Finance*, 63(3), 1437–1467.

Thomas, J. & Sycara, K. (2000). Integrating genetic algorithms and text learning for financial prediction. In A. Wu (Ed.), *Proceedings of the GECCO-2000 Workshop on Data Mining with Evolutionary Algorithms* (pp. 72–75). Las Vegas.

Thomas, J. D. (2003). *News and Trading Rules*. PhD thesis, Carnegie Mellon University, Pittsburgh.

Thomas, J. D. & Sycara, K. (1999). The importance of simplicity and validation in genetic programming for data mining in financial data. In *Proceedings of the joint GECCO-99 and AAAI-99 Workshop on Data Mining with Evolutionary Algorithms: Research Directions*.

Tong, R. (2001). An operational system for detecting and tracking opinions in online discussions. In *Working Notes of the SIGIR Workshop on Operational Text Classification* (pp. 1–6). New York. ACM.

Turney, P. D. (2002). Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (pp. 417–424). Stroudsburg: Association for Computational Linguistics.

UBM (2011). Annual reports & accounts 2011. `http://asp-gb.secure-zone.net/v2/indexPop.jsp?id=1134/2178/4760&lng=en`. Accessed on 12/07/02.

van Rijsbergen, C. J. (1979). *Information Retrieval*. Newton, MA: Butterworth-Heinemann, 2. edition.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*, chapter Constructing learning algorithms, (pp. 119–166). Springer: New York.

Wang, F., Liu, L., & Dou, C. (2012). Stock market volatility prediction: A service-oriented multi-kernel learning approach. In *Proceedings of the Ninth IEEE International Conference on Services Computing* (pp. 49–56). Washington, D.C.: IEEE Computer Society.

Wang, S. & Manning, C. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (pp. 90–94). Jeju Island: Association for Computational Linguistics.

Weiss, G. M., McCarthy, K., & Zabar, B. (2007). Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs? In R. Stahlbock, S. F. Crone, & S. Lessmann (Eds.), *Proceedings of the 2007 International Conference on Data Mining* (pp. 35–41). Las Vegas: CSREA Press.

Weiss, G. M. & Provost, F. J. (2003). Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19, 315–354.

Weiss, S. M., Indurkhya, N., Zhang, T., & Damerau, F. J. (2005). *Text Mining. Predictive Methods for Analyzing Unstructured Information*, chapter From textual information to numerical vectors, (pp. 15–46). Springer: New York.

Weissmann, D. (2004). Stemming - Methoden und Ergebnisse. Working Paper. Universität Heidelberg, Institut für Computerlinguistik.

Wilbur, J. & Sirotkin, K. (1992). The automatic identification of stop words. *Journal of Information Science*, 18, 45–55.

Witten, I. H., Don, K. J., Dewsnip, M., & Tablan, V. (2004). Text mining in a digital library. *International Journal on Digital Libraries*, 4, 56–59.

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems. San Francisco: Morgan Kaufmann Publishers, 3. edition.

Witten, I. H., Moffat, A., & Bell, T. C. (1999a). *Managing Gigabytes: Compressing and Indexing Documents and Images*. San Francisco: Morgan Kaufmann Publishers.

Witten, I. H., Paynter, G. W., Frank, E., Gutwin, C., & Nevill-Manning, C. G. (1999b). KEA: Practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM Conference on Digital Libraries* (pp. 254–255). New York: ACM.

Wüthrich, B., Permunetilleke, D., Leung, S., Cho, V., Zhang, J., & Lam, W. (1998). Daily prediction of major stock indices from textual www data. In *Proceedings of the Fourth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* New York: ACM.

Xu, J. & Croft, W. B. (1998). Corpus-based stemming using co-occurrence of word variants. *ACM Transactions on Information Systems*, 16, 61–81.

Yang, Y. (1994). Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 13–22). New York: Springer.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1, 69–90.

Yang, Y. & Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 42–49). New York: ACM.

Yang, Y. & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In D. H. Fisher (Ed.), *Proceedings of ICML-97, 14th International Conference on Machine Learning* (pp. 412–420). Nashville: Morgan Kaufmann Publishers.

Yang, Y. & Wilbur, J. (1996). Using corpus statistics to remove redundant words in text categorization. *Journal of the American Society for Information Science*, 47(5), 357–369.

Zak, I. & Ciura, M. (2005). Automatic text categorization. In *33rd International Conference on Information Systems Architecture and Technology*.

Zanni, L., Serafini, T., & Zanghirati, G. (2006). Parallel software for training large scale support vector machines on multiprocessor systems. *Journal of Machine Learning Research*, 7, 1467–1492.

Appendix A

Data preparation

A.A S&P 500 companies used as data source

Table A.1: All S&P 500 companies and their Bloomberg tickers

Ticker	Company name	Ticker	Company name
A UN	Agilent Technologies Inc	JNPR UN	Juniper Networks Inc
AA UN	Alcoa Inc	JOY UN	Joy Global Inc
AAPL UW	Apple Inc	JPM UN	JPMorgan Chase & Co
ABC UN	AmerisourceBergen Corp	JWN UN	Nordstrom Inc
ABT UN	Abbott Laboratories	K UN	Kellogg Co
ACE UN	ACE Ltd	KEY UN	KeyCorp
ACN UN	Accenture PLC	KFT UN	Kraft Foods Inc
ADBE UW	Adobe Systems Inc	KIM UN	Kimco Realty Corp
ADI UW	Analog Devices Inc	KLAC UW	KLA-Tencor Corp
ADM UN	Archer-Daniels-Midland Co	KMB UN	Kimberly-Clark Corp
ADP UW	Automatic Data Processing Inc	KMX UN	CarMax Inc
ADSK UW	Autodesk Inc	KO UN	Coca-Cola Co/The
AEE UN	Ameren Corp	KR UN	Kroger Co/The
AEP UN	American Electric Power Co Inc	KSS UN	Kohl's Corp
AES UN	AES Corp/The	L UN	Loews Corp
AET UN	Aetna Inc	LEG UN	Leggett & Platt Inc
AFL UN	Aflac Inc	LEN UN	Lennar Corp
AGN UN	Allergan Inc	LH UN	LABORATORY CORP OF AMERICA HOLDINGS
AIG UN	American International Group Inc	LIFE UW	Life Technologies Corp
AIV UN	Apartment Investment & Management Co	LLL UN	L-3 COMMUNICATIONS HOLDINGS INC
AIZ UN	Assurant Inc	LLTC UW	Linear Technology Corp
AKAM UW	Akamai Technologies Inc	LLY UN	Eli Lilly & Co
ALL UN	Allstate Corp/The	LM UN	Legg Mason Inc
ALTR UW	Altera Corp	LMT UN	Lockheed Martin Corp
AMAT UW	Applied Materials Inc	LNC UN	Lincoln National Corp
AMD UN	Advanced Micro Devices Inc	LO UN	Lorillard Inc
AMGN UW	Amgen Inc	LOW UN	Lowe's Cos Inc
AMP UN	Ameriprise Financial Inc	LSI UN	LSI Corp
AMT UN	American Tower Corp	LTD UN	Ltd Brands Inc
AMZN UW	Amazon.com Inc	LUK UN	Leucadia National Corp
AN UN	AutoNation Inc	LUV UN	Southwest Airlines Co
ANF UN	Abercrombie & Fitch Co	LXK UN	Lexmark International Inc
ANR UN	Alpha Natural Resources Inc	M UN	Macy's Inc
AON UN	Aon PLC	MA UN	Mastercard Inc
APA UN	Apache Corp	MAR UN	MARRIOTT INTERNATIONAL INC
APC UN	Anadarko Petroleum Corp	MAS UN	Masco Corp
APD UN	Air Products & Chemicals Inc	MAT UW	Mattel Inc
APH UN	Amphenol Corp	MCD UN	McDonald's Corp
APOL UW	Apollo Group Inc	MCHP UW	Microchip Technology Inc
ARG UN	Airgas Inc	MCK UN	McKesson Corp
ATI UN	Allegheny Technologies Inc	MCO UN	Moody's Corp

Table A.1 – continued from previous page

Ticker	Company name	Ticker	Company name
AVB UN	AvalonBay Communities Inc	MDT UN	Medtronic Inc
AVP UN	Avon Products Inc	MET UN	MetLife Inc
AVY UN	Avery Dennison Corp	MHP UN	McGraw-Hill Cos Inc/The
AXP UN	American Express Co	MJN UN	Mead Johnson Nutrition Co
AZO UN	AutoZone Inc	MKC UN	MCCORMICK AND CO INC
BA UN	Boeing Co/The	MMC UN	Marsh & McLennan Cos Inc
BAC UN	Bank of America Corp	MMI UN	Motorola Mobility Holdings Inc
BAX UN	Baxter International Inc	MMM UN	3M Co
BBBY UW	Bed Bath & Beyond Inc	MO UN	Altria Group Inc
BBT UN	BB&T Corp	MOLX UW	Molex Inc
BBY UN	Best Buy Co Inc	MON UN	Monsanto Co
BCR UN	C R BARD INC	MOS UN	Mosaic Co/The
BDX UN	Becton Dickinson and Co	MPC UN	MARATHON PETROLEUM CO LP
BEAM UN	Beam Inc	MRK UN	Merck & Co Inc
BEN UN	Franklin Resources Inc	MRO UN	Marathon Oil Corp
BF_B UN	Brown-Forman Corp	MS UN	Morgan Stanley
BHI UN	Baker Hughes Inc	MSFT UW	Microsoft Corp
BIG UN	Big Lots Inc	MSI UN	Motorola Solutions Inc
BIIB UW	Biogen Idec Inc	MTB UN	M&T Bank Corp
BK UN	BANK OF NEW YORK MELLON CORP	MU UW	Micron Technology Inc
BLK UN	BlackRock Inc	MUR UN	Murphy Oil Corp
BLL UN	Ball Corp	MWV UN	MeadWestvaco Corp
BMC UW	BMC Software Inc	MYL UW	MYLAN INC
BMS UN	Bemis Co Inc	NBL UN	Noble Energy Inc
BMY UN	Bristol-Myers Squibb Co	NBR UN	Nabors Industries Ltd
BRCM UW	Broadcom Corp	NDAQ UW	NASDAQ OMX Group Inc/The
BRK_B UN	Berkshire Hathaway Inc	NE UN	Noble Corp
BSX UN	Boston Scientific Corp	NEE UN	NextEra Energy Inc
BTU UN	Peabody Energy Corp	NEM UN	Newmont Mining Corp
BWA UN	BorgWarner Inc	NFLX UW	Netflix Inc
BXP UN	Boston Properties Inc	NFX UN	Newfield Exploration Co
C UN	Citigroup Inc	NI UN	NiSource Inc
CA UW	CA Inc	NKE UN	NIKE Inc
CAG UN	ConAgra Foods Inc	NOC UN	Northrop Grumman Corp
CAH UN	Cardinal Health Inc	NOV UN	National Oilwell Varco Inc
CAM UN	Cameron International Corp	NRG UN	NRG Energy Inc
CAT UN	Caterpillar Inc	NSC UN	Norfolk Southern Corp
CB UN	Chubb Corp/The	NTAP UW	NetApp Inc
CBE UN	Cooper Industries PLC	NTRS UW	Northern Trust Corp
CBG UN	CBRE Group Inc	NU UN	Northeast Utilities
CBS UN	CBS Corp	NUE UN	Nucor Corp
CCE UN	Coca-Cola Enterprises Inc	NVDA UW	NVIDIA Corp
CCI UN	Crown Castle International Corp	NVLS UW	NOVELLUS SYSTEMS INC
CCL UN	Carnival Corp	NWL UN	Newell Rubbermaid Inc
CELG UW	Celgene Corp	NWSA UW	News Corp
CERN UW	Cerner Corp	NYX UN	NYSE EURONEXT INC
CF UN	CF Industries Holdings Inc	OI UN	Owens-Illinois Inc
CFN UN	CareFusion Corp	OKE UN	ONEOK Inc
CHK UN	Chesapeake Energy Corp	OMC UN	Omnicom Group Inc
CHRW UW	C H ROBINSON WORLDWIDE INC	ORCL UW	Oracle Corp
CI UN	Cigna Corp	ORLY UW	O'Reilly Automotive Inc
CINF UW	Cincinnati Financial Corp	OXY UN	Occidental Petroleum Corp
CL UN	Colgate-Palmolive Co	PAYX UW	Paychex Inc
CLF UN	Cliffs Natural Resources Inc	PBCT UW	People's United Financial Inc
CLX UN	Clorox Co/The	PBI UN	Pitney Bowes Inc
CMA UN	Comerica Inc	PCAR UW	PACCAR Inc
CMCSA UW	Comcast Corp	PCG UN	PG&E Corp
CME UW	CME Group Inc	PCL UN	Plum Creek Timber Co Inc
CMG UN	Chipotle Mexican Grill Inc	PCLN UW	priceline.com Inc
CMI UN	Cummins Inc	PCP UN	Precision Castparts Corp
CMS UN	CMS Energy Corp	PCS UN	MetroPCS Communications Inc
CNP UN	CenterPoint Energy Inc	PDCO UW	Patterson Cos Inc
CNX UN	CONSOL Energy Inc	PEG UN	PUBLIC SERVICE ENTERPRISE GROUP INC
COF UN	Capital One Financial Corp	PEP UN	PepsiCo Inc
COG UN	Cabot Oil & Gas Corp	PFE UN	Pfizer Inc
COH UN	Coach Inc	PFG UN	Principal Financial Group Inc
COL UN	Rockwell Collins Inc	PG UN	Procter & Gamble Co/The
COP UN	ConocoPhillips	PGN UN	Progress Energy Inc
COST UW	Costco Wholesale Corp	PGR UN	Progressive Corp/The
COV UN	Covidien PLC	PH UN	Parker Hannifin Corp
CPB UN	Campbell Soup Co	PHM UN	PulteGroup Inc
CRM UN	Salesforce.com Inc	PKI UN	PerkinElmer Inc
CSC UN	Computer Sciences Corp	PLD UN	Prologis Inc
CSCO UW	Cisco Systems Inc	PLL UN	Pall Corp

Table A.1 – continued from previous page

Ticker	Company name	Ticker	Company name
CSX UN	CSX Corp	PM UN	PHILIP MORRIS INTERNATIONAL INC
CTAS UW	Cintas Corp	PNC UN	PNC FINANCIAL SERVICES GROUP INC
CTL UN	CenturyLink Inc	PNW UN	Pinnacle West Capital Corp
CTSH UW	COGNIZANT TECHNOLOGY SOLUTIONS CORP	POM UN	Pepco Holdings Inc
CTXS UW	Citrix Systems Inc	PPG UN	PPG Industries Inc
CVC UN	Cablevision Systems Corp	PPL UN	PPL Corp
CVH UN	Coventry Health Care Inc	PRGO UW	Perrigo Co
CVS UN	CVS Caremark Corp	PRU UN	Prudential Financial Inc
CVX UN	Chevron Corp	PSA UN	Public Storage
D UN	DOMINION RESOURCES INC	PWR UN	Quanta Services Inc
DD UN	E I DU PONT DE NEMOURS & CO	PX UN	Praxair Inc
DE UN	Deere & Co	PXD UN	Pioneer Natural Resources Co
DELL UW	Dell Inc	QCOM UW	QUALCOMM Inc
DF UN	Dean Foods Co	QEP UN	QEP Resources Inc
DFS UN	DISCOVER FINANCIAL SERVICES LLC	R UN	Ryder System Inc
DGX UN	Quest Diagnostics Inc	RAI UN	Reynolds American Inc
DHI UN	D R HORTON INC	RDC UN	Rowan Cos Inc
DHR UN	Danaher Corp	RF UN	Regions Financial Corp
DIS UN	Walt Disney Co/The	RHI UN	Robert Half International Inc
DISCA UW	Discovery Communications Inc	RHT UN	Red Hat Inc
DLTR UW	Dollar Tree Inc	RL UN	Ralph Lauren Corp
DNB UN	Dun & Bradstreet Corp/The	ROK UN	Rockwell Automation Inc
DNR UN	Denbury Resources Inc	ROP UN	Roper Industries Inc
DO UN	Diamond Offshore Drilling Inc	ROST UW	Ross Stores Inc
DOV UN	Dover Corp	RRC UN	Range Resources Corp
DOW UN	Dow Chemical Co/The	RRD UW	R R DONNELLEY & SONS CO
DPS UN	Dr Pepper Snapple Group Inc	RSG UN	Republic Services Inc
DRI UN	Darden Restaurants Inc	RTN UN	Raytheon Co
DTE UN	DTE Energy Co	S UN	Sprint Nextel Corp
DTV UW	DIRECTV	SAI UN	SAIC Inc
DUK UN	Duke Energy Corp	SBUX UW	Starbucks Corp
DV UN	DeVry Inc	SCG UN	SCANA Corp
DVA UN	DaVita Inc	SCHW UN	Charles Schwab Corp/The
DVN UN	Devon Energy Corp	SE UN	Spectra Energy Corp
EA UW	Electronic Arts Inc	SEE UN	Sealed Air Corp
EBAY UW	eBay Inc	SHLD UW	Sears Holdings Corp
ECL UN	Ecolab Inc	SHW UN	Sherwin-Williams Co/The
ED UN	Consolidated Edison Inc	SIAL UW	Sigma-Aldrich Corp
EFX UN	Equifax Inc	SJM UN	J M SMUCKER CO
EIX UN	Edison International	SLB UN	Schlumberger Ltd
EL UN	Estee Lauder Cos Inc/The	SLE UN	Sara Lee Corp
EMC UN	EMC Corp	SLM UW	SLM Corp
EMN UN	Eastman Chemical Co	SNA UN	Snap-on Inc
EMR UN	Emerson Electric Co	SNDK UW	SanDisk Corp
EOG UN	EOG Resources Inc	SNI UN	SCRIPPS NETWORKS INTERACTIVE INC
EP UN	El Paso Corp	SO UN	Southern Co/The
EQR UN	Equity Residential	SPG UN	Simon Property Group Inc
EQT UN	EQT Corp	SPLS UW	Staples Inc
ESRX UW	Express Scripts Holding Co	SRCL UW	Stericycle Inc
ETFC UW	E TRADE FINANCIAL CORP	SRE UN	Sempra Energy
ETN UN	Eaton Corp	STI UN	SunTrust Banks Inc
ETR UN	Entergy Corp	STJ UN	St Jude Medical Inc
EW UN	Edwards Lifesciences Corp	STT UN	State Street Corp
EXC UN	Exelon Corp	STZ UN	Constellation Brands Inc
EXPD UW	EXPEDITORS INTERNATIONAL OF WASHINGTON INC	SUN UN	Sunoco Inc
EXPE UW	Expedia Inc	SVU UN	SUPERVALU Inc
F UN	Ford Motor Co	SWK UN	Stanley Black & Decker Inc
FAST UW	Fastenal Co	SWN UN	Southwestern Energy Co
FCX UN	FREEPORT-MCMORAN COPPER AND GOLD INC	SWY UN	Safeway Inc
FDO UN	Family Dollar Stores Inc	SYK UN	Stryker Corp
FDX UN	FedEx Corp	SYMC UW	Symantec Corp
FE UN	FirstEnergy Corp	SYY UN	Sysco Corp
FFIV UW	F5 Networks Inc	T UN	AT&T Inc
FHN UN	First Horizon National Corp	TAP UN	Molson Coors Brewing Co
FII UN	Federated Investors Inc	TDC UN	Teradata Corp
FIS UN	FIDELITY NATIONAL INFORMATION SERVICES INC	TE UN	TECO Energy Inc
FISV UW	Fiserv Inc	TEG UN	INTEGRYS ENERGY GROUP INC
FITB UW	Fifth Third Bancorp	TEL UN	TE Connectivity Ltd
FLIR UW	FLIR Systems Inc	TER UN	Teradyne Inc
FLR UN	Fluor Corp	TGT UN	Target Corp
FLS UN	Flowserve Corp	THC UN	Tenet Healthcare Corp

Table A.1 – continued from previous page

Ticker	Company name	Ticker	Company name
FMC UN	FMC Corp	TIE UN	Titanium Metals Corp
FOSL UW	Fossil Inc	TIF UN	TIFFANY AND CO
FRX UN	Forest Laboratories Inc	TJX UN	TJX Cos Inc
FSLR UW	First Solar Inc	TMK UN	Torchmark Corp
FTI UN	FMC Technologies Inc	TMO UN	Thermo Fisher Scientific Inc
FTR UW	Frontier Communications Corp	TRIP UW	TripAdvisor Inc
GAS UN	AGL Resources Inc	TROW UW	T Rowe Price Group Inc
GCI UN	Gannett Co Inc	TRV UN	Travelers Cos Inc/The
GD UN	General Dynamics Corp	TSN UN	Tyson Foods Inc
GE UN	General Electric Co	TSO UN	Tesoro Corp
GILD UW	Gilead Sciences Inc	TSS UN	TOTAL SYSTEM SERVICES INC
GIS UN	General Mills Inc	TWC UN	Time Warner Cable Inc
GLW UN	Corning Inc	TWX UN	Time Warner Inc
GME UN	GameStop Corp	TXN UW	Texas Instruments Inc
GNW UN	Genworth Financial Inc	TXT UN	Textron Inc
GOOG UW	Google Inc	TYC UN	Tyco International Ltd
GPC UN	Genuine Parts Co	UNH UN	UnitedHealth Group Inc
GPS UN	Gap Inc/The	UNM UN	Unum Group
GR UN	Goodrich Corp	UNP UN	Union Pacific Corp
GS UN	Goldman Sachs Group Inc/The	UPS UN	United Parcel Service Inc
GT UN	Goodyear Tire & Rubber Co/The	URBN UW	Urban Outfitters Inc
GWW UN	WW Grainger Inc	USB UN	US Bancorp
HAL UN	Halliburton Co	UTX UN	United Technologies Corp
HAR UN	HARMAN INTERNATIONAL INDUSTRIES INC	V UN	Visa Inc
HAS UW	Hasbro Inc	VAR UN	Varian Medical Systems Inc
HBAN UW	HUNTINGTON BANCSHARES INC	VFC UN	VF Corp
HCBK UW	Hudson City Bancorp Inc	VIAB UW	Viacom Inc
HCN UN	Health Care REIT Inc	VLO UN	Valero Energy Corp
HCP UN	HCP Inc	VMC UN	Vulcan Materials Co
HD UN	Home Depot Inc/The	VNO UN	Vornado Realty Trust
HES UN	Hess Corp	VRSN UW	VeriSign Inc
HIG UN	HARTFORD FINANCIAL SERVICES GROUP INC	VTR UN	Ventas Inc
HNZ UN	H J HEINZ CO	VZ UN	Verizon Communications Inc
HOG UN	Harley-Davidson Inc	WAG UN	Walgreen Co
HON UN	Honeywell International Inc	WAT UN	Waters Corp
HOT UN	STARWOOD HOTELS & RESORTS WORLDWIDE INC	WDC UN	Western Digital Corp
HP UN	Helmerich & Payne Inc	WEC UN	Wisconsin Energy Corp
HPQ UN	Hewlett-Packard Co	WFC UN	Wells Fargo & Co
HRB UN	H&R Block Inc	WFM UW	Whole Foods Market Inc
HRL UN	Hormel Foods Corp	WHR UN	Whirlpool Corp
HRS UN	Harris Corp	WIN UW	Windstream Corp
HSP UN	Hospira Inc	WLP UN	WellPoint Inc
HST UN	Host Hotels & Resorts Inc	WM UN	Waste Management Inc
HSY UN	Hershey Co/The	WMB UN	Williams Cos Inc/The
HUM UN	Humana Inc	WMT UN	Wal-Mart Stores Inc
IBM UN	INTERNATIONAL BUSINESS MACHINES CORP	WPI UN	Watson Pharmaceuticals Inc
ICE UN	IntercontinentalExchange Inc	WPO UN	Washington Post Co/The
IFF UN	INTERNATIONAL FLAVORS & FRAGRANCES INC	WPX UN	WPX Energy Inc
IGT UN	INTERNATIONAL GAME TECHNOLOGY INC	WU UN	Western Union Co/The
INTC UW	Intel Corp	WY UN	Weyerhaeuser Co
INTU UW	Intuit Inc	WYN UN	Wyndham Worldwide Corp
IP UN	International Paper Co	WYNN UW	Wynn Resorts Ltd
IPG UN	INTERPUBLIC GROUP OF COS INC	X UN	United States Steel Corp
IR UN	Ingersoll-Rand PLC	XEL UN	Xcel Energy Inc
IRM UN	Iron Mountain Inc	XL UN	XL Group Plc
ISRG UW	Intuitive Surgical Inc	XLNX UW	Xilinx Inc
ITW UN	Illinois Tool Works Inc	XOM UN	Exxon Mobil Corp
IVZ UN	Invesco Ltd	XRAY UW	DENTSPLY International Inc
JBL UN	Jabil Circuit Inc	XRX UN	Xerox Corp
JCI UN	Johnson Controls Inc	XYL UN	XYLEM INC
JCP UN	J C PENNEY CO INC	YHOO UW	Yahoo! Inc
JDSU UW	JDS Uniphase Corp	YUM UN	Yum! Brands Inc
JEC UN	Jacobs Engineering Group Inc	ZION UW	ZIONS BANCORP
JNJ UN	Johnson & Johnson	ZMH UN	Zimmer Holdings Inc

A.B Format of a news release distributed by LexisNexis

Listing A.1: Example for a news release distributed by LexisNexis

```
\begin{verbatim}
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "xhtml11-flat.dtd">
<html>
        <head>
                <title>Inventor, Richard P. Mettke of Columbus, Ohio Wins &#34;Round One&#34;
                      Against Hewlett-Packard in Federal Court in Their Attempt to Dismiss His
                      Lawsuit for $275,000,000  PR Newswire April 10, 2012 Tuesday   12:46 PM EST </
                      title>
                <meta content="FULL" name="_lnbillview"/>
                <meta content="00001" name="_lnminrev"/>
                <meta content="April 11, 2012" name="_loaddate"/>
                <meta content="55CP-0F91-F18Y-Y42D" name="_lndocid"/>
                <meta content=" PR Newswire Association LLC" name="_lncopyrightholder"/>
                <meta content="April 11, 2012" name="_eoptdate"/>
                <meta content="04:45:01 EDT" name="_eopttime"/>
                <meta content="e6874d82b9c4d010b7e764627821d6b0a7fdec475fe10de92cc9d928615fa1c48
                        e676ac20c8b0a03ca75247ca91c5d58d77abee492ef12ffe3719e76519fa049959f8ce349
                        a17e786187e07f9e6e45ed7e86d2a8fe54a0411adb28327cb7b00c2a9a84ee96f1b875aa7
                        e84b1b9fce853382265b1851990454a337d0b194c727a1083b7af064fef5b" name="
                              documentToken"/>
                <meta content="" name="docHeading"/>
                <meta content="" name="docLang"/>
                <meta content="" name="docCountry"/>
                <meta content="PR Newswire" name="sourceName"/>
        </head>
        <body>
                <div class="document-metadata" style="display: none">
                        <span class="doc-id"/>
                </div>
                <div class="PUB" style="text-align: center">
                        <br/>
                        <span class="hit">
                                <b>PR</b>
                        </span>
                        <span class="hit">
                                <b>Newswire</b>
                        </span>
                </div>
                <div class="DISPLAY-DATE" style="text-align: center">
                        <div class="PUB-DATE">
                                <span class="hit">
                                        <b>April</b>
                                </span> 10, 2012 Tuesday 12:46 PM EST</div>
                        <div class="TIME-RECEIVED"/>
                </div>
                <div class="HEADLINE">
                        <h1>Inventor, Richard P. Mettke of Columbus, Ohio Wins &#34;Round One&#34;
                              Against Hewlett-Packard in Federal Court in Their Attempt to Dismiss
                              His Lawsuit for $275,000,000</h1>
                </div>
                <div class="LENGTH">
                        <span class="header">
                                <strong>LENGTH: </strong>
                        </span>281 words</div>
                <div class="DATELINE">
                        <span class="header">
                                <strong>DATELINE: </strong>
                        </span>COLUMBUS, Ohio, April 10, 2012 </div>
                <div class="BODY">
                        <div class="REAL-LEAD">
                                <p>Inventor, Richard P. Mettke of Columbus, Ohio won &#34;round one
                                      &#34; in Federal Court against Hewlett-Packard (HP) in their
                                      attempt to dismiss his lawsuit against them for fraud and
                                      contract breach. The United States District Court, Southern
                                      District of Ohio ruled in a 19-page order on April 6, 2012
                                      that Mettke’s lawsuit (Case # No.2:11-cv-410) could move
                                      forward to trial. Mettke filed a $275,000,000 lawsuit against
                                      HP on April 13, 2011 for fraud and contract breach.</p>
                        </div>
                        <div class="BODY-1">
                                <p>HP walked away with a minor win. The court dismissed the fraud
                                      specification in the lawsuit because of Ohio's statute of
                                      limitations for fraud which is four years. The Federal Court
                                      ruled that the breach of contract specification against HP
                                      could move forward to trial.</p><p>The case file states that
                                      HP settled a patent infringement lawsuit with Mettke on April
                                      27, 1998. Mettke's patent was 5,602,905, &#34;On-Line
                                      Communication Terminal/Apparatus.&#34; The suit was settled
                                      based on HP's representation that they had no &#34;present
                                      plans to make, use or sell Internet Kiosks.&#34; The court
                                      filing states that in August of 1998 HP deployed &#34;Internet
                                       Kiosks&#34; to Circuit City, Best Buy and CompUSA to sell HP
                                      PC's and other related items. This was in direct contradiction
                                       to Mettke's settlement agreement. The court file further
                                      states that Hewlett-Packard clearly breached Mettke's
                                      settlement agreement; as well as committed fraud and bad faith
                                       in the negotiation and execution of the settlement agreement
                                      dated April 27, 1998.</p><p>Since, April 1998, HP and its
                                      partners have profited in the billions of dollars selling
                                      goods and services through their Internet capable kiosks.</p><
                                      p>Contact: Richard <span class="url" href="mailto:Mettke614
                                      -354-7062rmettke@aol.com">Mettke614-354-7062rmettke@aol.com</
                                      span>
                                </p><p>SOURCE Richard Mettke</p>
                        </div>
                </div>
                <div class="URL-SEG">
                        <span class="header">
                                <strong>URL: </strong>
                        </span>
                        <span class="url" href="http://www.prnewswire.com">http://www.prnewswire.
                              com</span>
                </div>
                <div class="SUBJECT">
                        <span class="header">
                                <strong>SUBJECT: </strong>
                        </span>
                        <div class="LN-SUBJ">
                                <span class="term">LITIGATION<span class="score"> (96%)</span>
                                </span>
                                <span class="term">; SETTLEMENT &#38; COMPROMISE<span class="score
                                      "> (90%)</span>
                                </span>
                                <span class="term">; SUITS &#38; CLAIMS<span class="score"> (90%)</
                                      span>
                                </span>
                                <span class="term">; DECISIONS &#38; RULINGS<span class="score">
                                      (90%)</span>
                                </span>
                                <span class="term">; PRESS RELEASES<span class="score"> (90%)</span
                                      >
                                </span>
                                <span class="term">; SETTLEMENTS &#38; DECISIONS<span class="score
                                      "> (90%)</span>
                                </span>
                                <span class="term">; LAW COURTS &#38; TRIBUNALS<span class="score">
                                       (90%)</span>
                                </span>
                                <span class="term">; BREACH OF CONTRACT<span class="score"> (90%)</
                                      span>
                                </span>
                                <span class="term">; ALTERNATIVE DISPUTE RESOLUTION<span class="
                                      score"> (89%)</span>
                                </span>
                                <span class="term">; PATENTS<span class="score"> (79%)</span>
                                </span>
                                <span class="term">; PATENT INFRINGEMENT<span class="score"> (78%)
                                      </span>
                                </span>
                                <span class="term">; PATENT LAW<span class="score"> (78%)</span>
                                </span>
                                <span class="term">; BAD FAITH<span class="score"> (77%)</span>
                                </span>
                                <span class="term">; CONTRACTS LAW<span class="score"> (77%)</span>
                                </span>
                                <span class="term">; INTELLECTUAL PROPERTY LAW<span class="score">
                                      (74%)</span>
                                </span>
                                <span class="term">; STATUTE OF LIMITATIONS<span class="score">
                                      (73%)</span>
                                </span>
                                <span class="term">; ELECTRONIC KIOSKS<span class="score"> (71%)</
                                      span>
                                </span>
                        </div>
                        <div class="PUB-SUBJECT">Mettke-Hewlett-Packrd; LAW Legal Issues</div>
                </div>
                <div class="COMPANY">
                        <span class="header">
                                <strong>COMPANY: </strong>
                        </span>
                        <div class="LN-CO">
                                <span class="term">
                                        <span class="hit">
                                                <b>HEWLETT</b>
                                        </span>-<span class="hit">
                                                <b>PACKARD</b>
                                        </span>
                                        <span class="hit">
                                                <b>CO</b>
                                        </span>
                                        <span class="score"> (<span class="hit">
                                                        <b>90%)</b>
                                                </span>
                                        </span>
                                </span>
                                <span class="term">; COMPUSA INC<span class="score"> (53%)</span>
                                </span>
                        </div>
                        <div class="PUB-COMPANY">Richard Mettke</div>
                </div>
                <div class="TICKER">
                        <span class="header">
                                <strong>TICKER: </strong>
                        </span>
                        <div class="LN-TS">
                                <span class="term">HPQ (NYSE)<span class="score"> (90%)</span>
                                </span>
                        </div>
                </div>
                <div class="INDUSTRY">
                        <span class="header">
                                <strong>INDUSTRY: </strong>
                        </span>
                        <div class="LN-IND">
                                <span class="term">NAICS511210 SOFTWARE PUBLISHERS<span class="
                                      score"> (90%)</span>
                                </span>
                                <span class="term">; NAICS334119 OTHER COMPUTER PERIPHERAL
                                      EQUIPMENT MANUFACTURING<span class="score"> (90%)</span>
                                </span>
                                <span class="term">; NAICS334111 ELECTRONIC COMPUTER MANUFACTURING<
                                      span class="score"> (90%)</span>
                                </span>
                                <span class="term">; NAICS443120 COMPUTER &#38; SOFTWARE STORES<
                                      span class="score"> (53%)</span>
                                </span>
                                <span class="term">; SIC5734 COMPUTER &#38; COMPUTER SOFTWARE
                                      STORES<span class="score"> (53%)</span>
                                </span>
                        </div>
                        <div class="PUB-INDUSTRY">PUB Publishing; Information Services</div>
                </div>
                <div class="GEOGRAPHIC">
                        <span class="header">
                                <strong>GEOGRAPHIC: </strong>
                        </span>
                        <div class="LN-CITY">
                                <span class="term">COLUMBUS, OH, USA<span class="score"> (92%)</
                                      span>
                                </span>
```

```
                        </div>
                        <div class="LN-ST">
                                <span class="term">OHIO, USA<span class="score"> (96%)</span>
                                </span>
                        </div>
                        <div class="LN-COUNTRY">
                                <span class="term">UNITED STATES<span class="score"> (96%)</span>
                                </span>
                        </div>
                        <div class="PUB-REGION">Ohio</div>
                </div>
                <div class="LOAD-DATE">
                        <span class="header">
                                <strong>LOAD-DATE: </strong>
                        </span>April 11, 2012</div>
                <div class="LANGUAGE">
                        <span class="header">
                                <strong>LANGUAGE: </strong>
                        </span>
                        <span class="hit">
                                <b>ENGLISH</b>
                        </span>
                </div>
                <div class="PUBLICATION-TYPE">
                        <span class="header">
                                <strong>PUBLICATION-TYPE: </strong>
                        </span>Newswire</div>
                <div class="COPYRIGHT" style="text-align: center">
                        <div class="PUB-COPYRIGHT">
                                <br/>Copyright 2012 PR Newswire Association LLC<br/>All Rights
                                     Reserved</div>
                </div>
        </body>
</html>
\end{verbatim}
```

A.C The Penn Treebank tag set

Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NP	Proper noun, singular
NPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PP	Personal pronoun
PP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner (wh = words starting with wh)
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

Table A.2: The Penn Treebank tag set

Appendix B

Training

B.A Stop word list used for text preprocessing

Table B.1: Stop word list used for text preprocessing

a a’s able about above according accordingly across actually after afterwards again against ain’t all allow allows almost alone along already also although always am among amongst an and another any anybody anyhow anyone anything anyway anyways anywhere apart appear appreciate appropriate are aren’t around as aside ask asking associated at available away awfully
b be became because become becomes becoming been before beforehand behind being believe below beside besides best better between beyond both brief but by
c c’mon c’s came can can’t cannot cant cause causes certain certainly changes clearly co com come comes concerning consequently consider considering contain containing contains corresponding could couldn’t course currently
d definitely described despite did didn’t different do does doesn’t doing don’t done down downwards during
e each edu eg eight either else elsewhere enough entirely especially et etc even ever every everybody everyone everything everywhere ex exactly example except
f far few fifth first five followed following follows for former formerly forth four from further furthermore
g get gets getting given gives go goes going gone got gotten greetings

h had hadn’t happens hardly has hasn’t have haven’t having he he’s hello help hence her here here’s hereafter hereby herein hereupon hers herself hi him himself his hither hopefully how howbeit however
i i’d i’ll i’m i’ve ie if ignored immediate in inasmuch inc indeed indicate indicated indicates inner insofar instead into inward is isn’t it it’d it’ll it’s its itself
j just
k keep keeps kept know knows known
l last lately later latter latterly least less lest let let’s like liked likely little look looking looks ltd
m mainly many may maybe me mean meanwhile merely might more moreover most mostly much must my myself
n name namely nd near nearly necessary need needs neither never nevertheless new next nine no nobody non none noone nor normally not nothing novel now nowhere
o obviously of off often oh ok okay old on once one ones only onto or other others otherwise ought our ours ourselves out outside over overall own
p particular particularly per perhaps placed please plus possible presumably probably provides
q que quite qv
r rather rd re really reasonably regarding regardless regards relatively respectively right
s said same saw say saying says second secondly see seeing seem seemed seeming seems seen self selves sensible sent serious seriously seven several shall she should shouldn’t since six so some somebody somehow someone something sometime sometimes somewhat somewhere soon sorry specified specify specifying still sub such sup sure
t t’s take taken tell tends th than thank thanks thanx that that’s thats the their theirs them themselves then thence there there’s thereafter thereby therefore therein theres thereupon these they they’d they’ll they’re they’ve think third this thorough thoroughly those though three through throughout thru thus to together too took toward towards tried tries truly try trying twice two
u un under unfortunately unless unlikely until unto up upon us use used useful uses using usually uucp
v value various very via viz vs

w want wants was wasn't way we we'd we'll we're we've welcome well went were weren't what what's whatever when whence whenever where where's whereafter whereas whereby wherein whereupon wherever whether which while whither who who's whoever whole whom whose why will willing wish with within without won't wonder would would wouldn't
x
y yes yet you you'd you'll you're you've your yours yourself yourselves
z zero

B.B Additional experimental results

In the following, we present a set of additional experimental results, conducted on different data sets. The prediction performance of each experiment is given in precision for each class (π^{buy}, π^{sell}, π^{hold}), recall for each class (ρ^{buy}, ρ^{sell}, ρ^{hold}), the balanced F-measure for each class (F_1^{buy}, F_1^{sell}, F_1^{hold}), the overall balanced F-measure F_1 calculated using macro-averaging, and the accuracy A. All F_1 and A values are given in $\%$. All results are evaluated by means of a 10-fold cross validation of the training set.

In all results we use the abbreviations depicted in Table B.2. All settings that are not explicitly given are identical with the settings used in Chapter 5. For instance, we use k-NN with $k = 10$. In the column dimensionality reduction (DR), the parameters are given in the format [DR] $x\%$, or [DR] TH x. For instance, CHI 15% means that the dimensionality reduction method chi-squared is performed to reduce the total amount of features to 15% of the total amount of instances. CHI TH 0 means that chi-squared is performed to eleminate all features that obtain $\chi^2 = 0$. If the cell in the column DR is left blank, no dimensionality reduction is performed for this setting. In the column feature vector representation (VR), a blank cell means the boolean vector representation is chosen for this setting. In the column methods for handling unbalanced data (UD), the parameters are given in the format SS x, SM x, or MC x. For instance, SS 1 means that SpreadSubsample is performed with parameter 1, i.e. the majority class is undersampled until the distribution 1:1 is reached. SM 750 means that the minority classes are increased by 750%. SM even means that the minority classes are increased until an even distribution with respect to all classes is reached. MC 10 means that the MetaCost filter is applied with $C(i, j) = 0$ for $i = j$, $C(i, j) = 10$ for $i \in c_{\text{hold}}$, and $C(i, j) = 1$ in all other cases. A blank cell in the column UD means that no methods for handling unbalanced data are applied. In the column headline (HL), the parameter yes (y) means that only the headlines are used for training, instead of both headline and

Abbreviation	**Meaning**
B	Bigrams
BMA	Binary metalearning algorithm
BoW	Bag-of-words
CHI	Chi-squared
Class.	Classifier
DR	Dimensionality reduction
Feat.	Features
HL	Headline
IDF	IDF-TF
IG	Information gain
PB	POS bigrams
MC	MetaCost
NE	Named entities
S	Sentiment
SM	SMOTE
SS	SpreadSubsample
SVM	SVM with RBF kernel
SVM lin	SVM with linear kernel
TH x	Threshold $x\%$
UD	Methods for handling unbalanced data
VR	Feature vector representation
y	yes

Table B.2: Abbreviations used in the results

news body as usual. In the column binary learning algorithm (BMA), the parameter yes means that the binary learning algorithm is used to transform the 3-class problem into two binary prediction problems.

The results of the 3-class problem are shown in Table B.3, the results of the 2-class problem are shown in Table B.4. In Table B.5, we present a second 2-class problem that is constructed by copying the 3-class problem and eliminating all HOLD news. However, during the classification phase the HOLD news needs to be taken into account. Therefore, a considerably worse performance is expected compared to the 10-fold cross validation at the training set. In the column thesaurus (Thes.), the parameter yes means that we use the thesaurus to filter out irrelevant news like in both problems described previously. The rationale behind not performing any thesaurus filtering is that it increases the number of examples that can be trained. In the column threshold (TH), the profit threshold of classifying a news BUY or SELL is varied ($+/-0.3\%$, $+/-0.5\%$, $+/-0.2\%$).

Table B.3: Additional experimental results (2-class problem)

Features	**Classifier**	**DR**	**VR**	**UD**	$\pi^{\textbf{buy}}$	$\rho^{\textbf{buy}}$	$\pi^{\textbf{sell}}$	$\rho^{\textbf{sell}}$	$F_1^{\textbf{buy}}$	$F_1^{\textbf{sell}}$	F_1	A
BoW	Bayes				0.725	0.33	0.423	0.796	45.4	55.2	50.3	50.7987
B	Bayes				0.633	0.816	0.437	0.232	71.3	30.3	50.8	59.3184
B	Bayes			SS 1	0.663	0.325	0.401	0.732	43.6	51.8	47.7	48.0298
B	J48			SS 1	0.643	0.585	0.412	0.472	61.3	44	52.7	54.2066
B	SVM			SS 1	0.692	0.372	0.418	0.732	48.4	53.2	50.8	50.9052
B	SVM			SS 1	0.598	0.916	0	0	72.4	0	36.2	56.656
B/BoW	J48			SS 1	0.667	0.661	0.457	0.464	66.4	46	56.2	58.5729
B/BoW	Bayes			SS 1	0.675	0.435	0.418	0.659	52.9	51.2	52.1	52.0767
PB	J48			SS 1	0.667	0.516	0.425	0.581	58.2	49.1	53.7	54.1001
PB	Bayes			SS 1	0.637	0.58	0.405	0.464	60.7	43.2	52	53.5676
B/PB/BoW	J48			SS 1	0.643	0.508	0.404	0.542	56.8	46.3	51.6	52.0767
B/PB/BoW	Bayes			SS 1	0.666	0.546	0.43	0.556	60	48.5	54.3	54.9521
B	J48			SS 1	0.643	0.585	0.412	0.472	61.3	44	52.7	54.2066
B	Bayes				0.633	0.816	0.437	0.232	71.3	30.3	50.8	59.3184
B	SVM				0.627	0.957	0.519	0.075	75.8	13.1	44.5	62.0873
BoW	J48				0.655	0.633	0.435	0.458	64.4	44.6	54.5	56.656
BoW	J48				0.59	0.578	0.452	0.464	58.4	45.8	52.1	56.656
B	SVM				0.627	0.957	0.519	0.075	75.8	13.1	44.5	62.0873

Features	**Classifier**	**DR**	**VR**	**UD**	$\pi^{\textbf{buy}}$	$\rho^{\textbf{buy}}$	$\pi^{\textbf{sell}}$	$\rho^{\textbf{sell}}$	$F_1^{\textbf{buy}}$	$F_1^{\textbf{sell}}$	F_1	A
B	SVM lin				0.63	0.933	0.5	0.109	75.2	17.9	46.6	61.8743
BoW	SVM lin				0.62	0.974	0.423	0.031	75.8	5.8	40.8	61.4483
BoW	SVM lin	CHI			0.636	0.73	0.423	0.321	68	36.5	52.3	57.4015
BoW	SVM lin		IDF		0.623	0.966	0.474	0.05	75.7	9	42.4	61.6613
BoW	SVM lin				0.62	0.974	0.423	0.031	75.8	5.8	40.8	61.4483
PB	SVM lin				0.618	0.995	0	0	76.2	0	38.1	61.5548
B/BoW	SVM lin				0.616	0.962	0.313	0.028	75.1	5.1	40.1	60.5964
POS/S	SVM lin				0.617	0.986	0.273	0.008	75.9	1.6	38.8	61.3419
POS	SVM lin				0.618	0.983	0.333	0.014	75.9	2.7	39.3	61.3419
NE	SVM lin				0.617	0.962	0.333	0.031	75.2	5.7	40.5	60.7029
B/PB/BoW	SVM lin				0.624	0.964	0.5	0.059	75.8	10.6	43.2	61.8743
B/PB	SVM lin				0.62	0.976	0.417	0.028	75.8	5.2	40.5	61.4483
B/S	SVM lin				0.619	0.986	0.429	0.017	76.1	3.3	39.7	61.6613
B	SVM lin		IDF		0.626	0.921	0.459	0.109	74.5	17.6	46.1	61.1289
PB	SVM				0.625	0.966	0.512	0.059	75.9	10.6	43.3	61.9808
B/BoW/PB	SVM				0.624	0.955	0.469	0.064	75.5	11.3	43.4	61.5548
BoW	SVM				0.624	0.986	0.619	0.036	76.4	6.8	41.6	62.4068
BoW	SVM				0.656	0.466	0.411	0.603	54.5	48.9	51.7	51.8637
B	SVM			SS 1	0.657	0.32	0.398	0.729	43	51.5	47.3	47.6038

Features	**Classifier**	**DR**	**VR**	**UD**	π^{buy}	ρ^{buy}	π^{sell}	ρ^{sell}	F_1^{buy}	F_1^{sell}	F_1	A
B	SVM			SM 750	0.643	0.768	0.307	0.365	70	33.3	51.7	59.2119
B	SVM			SM even	0.636	0.824	0.452	0.235	71.8	30.9	51.4	59.9574
B	SVM	CHI TH 0		SM even	0.63	0.902	0.467	0.14	74.2	21.5	47.9	61.1289
B	SVM	IG TH 0		SM even	0.63	0.914	0.468	0.123	74.6	19.5	47.1	61.2354
B	J48	IG TH 0		SM even	0.618	0.959	0.351	0.036	75.2	6.5	40.9	60.7029
B	Bayes			SM even	0.626	0.929	0.461	0.098	74.8	16.2	45.5	61.2354
B	k-NN			SM even	0.841	0.164	0.412	0.95	27.4	57.5	42.5	46.3259
B	SVM	IG 15%		SM even	0.634	0.917	0.51	0.14	75	22	48.5	62.0873
B	SVM lin				0.626	0.921	0.459	0.109	74.5	17.6	46.1	61.1289
B	SVM lin				0.667	0.523	0.427	0.575	58.6	49	53.8	54.3131
B	SVM lin			SS 1	0.656	0.642	0.438	0.453	64.9	44.5	54.7	56.9755
B	Bayes				0.633	0.816	0.437	0.232	71.3	30.3	50.8	59.3184
B	k-NN				0.619	1	1	0.003	76.5	0.6	38.6	61.9808
B	J48			SM even	0.626	0.873	0.426	0.154	72.9	22.6	47.8	59.8509
B	J48				0.621	0.907	0.407	0.103	73.7	16.4	45.1	60.0639
B	SVM	IG 15%			0.624	0.957	0.479	0.064	75.5	11.3	43.4	61.6613
B	SVM	CHI 15%			0.624	0.957	0.479	0.064	75.5	11.3	43.4	61.6613

Table B.4: Additional experimental results (3-class problem)

Feat.	HL	**Class.**	**BMA**	**UD**	π^{buy}	ρ^{buy}	π^{hold}	ρ^{hold}	π^{sell}	ρ^{sell}	F_1^{buy}	F_1^{hold}	F_1^{sell}	F_1	A
BoW		SVM		MC 10	0.101	0.556	0.925	0.192	0.12	0.5	17.1	31.8	19.4	22.8	24.9201
BoW		SVM		SM 750	0.118	0.049	0.828	0.943	0.053	0.013	6.9	88.2	2.1	32.4	78.7007
POS		SVM		MC 10	0.089	0.457	0.913	0.202	0.109	0.475	14.9	33.1	17.7	21.9	24.7071
POS		SVM		MC 10	0.086	0.111	0.951	0.075	0.087	0.838	9.7	13.9	15.8	13.1	14.2705
POS		SVM			0.086	0.679	0.831	0.297	0.136	0.038	15.3	43.8	5.9	21.7	30.7774
S		SVM			0.086	0.58	0.84	0.216	0.094	0.225	15	34.4	13.3	20.9	24.8136
BoW		Bayes	y		0.104	0.185	0.842	0.703	0.152	0.275	13.3	76.6	19.6	36.5	62.1938
BoW		SVM	y	SS 1	0	0	0.83	0.95	0.163	0.088	0	88.6	11.4	33.3	79.4462
BoW		Bayes	y		0.107	0.247	0.847	0.599	0.144	0.363	14.9	70.2	20.6	35.2	54.8456
BoW		SVM			0	0	0.829	0.996	0.333	0.013	0	90.5	2.5	31	82.6411
B		SVM			0	0	0.828	0.996	0	0	0	90.4	0	30.1	82.5346
B		SVM		SS 1	0.094	0.679	0.837	0.343	0.108	0.05	16.5	48.7	6.8	24	34.7178
B		SVM		SS 2	0	0	0.829	0.981	0.214	0.038	0	89.9	6.5	32.1	81.5761
B		SVM		SM even	0.25	0.025	0.834	0.978	0.316	0.075	4.5	90	12.1	35.5	81.8956
B		SVM		SM 2	0.182	0.025	0.834	0.973	0.3	0.075	4.4	89.8	12	35.4	81.4696
B		J48			0	0	0.829	0.969	0.118	0.025	0	89.4	4.1	31.2	80.5112
B		Bayes			0.106	0.506	0.891	0.348	0.112	0.35	17.5	50.1	17	28.2	36.2087
B		k-NN			0	0	0.828	1	0	0	0	90.6	0	30.2	82.8541
B		Bayes		SS 1	0.091	0.531	0.87	0.231	0.104	0.338	15.5	36.5	15.9	22.6	26.6241

Feat.	HL	**Class.**	**BMA**	**UD**	$\pi^{\textbf{buy}}$	$\rho^{\textbf{buy}}$	$\pi^{\textbf{hold}}$	$\rho^{\textbf{hold}}$	$\pi^{\textbf{sell}}$	$\rho^{\textbf{sell}}$	$F_1^{\textbf{buy}}$	$F_1^{\textbf{hold}}$	$F_1^{\textbf{sell}}$	F_1	A
B		SVM lin			0	0	0.828	0.997	0	0	0	90.5	0	30.2	82.6411
B		Bayes	y		0.121	0.383	0.868	0.616	0.108	0.175	18.4	72.1	13.4	34.6	55.804
B		SVM	y	SS 1	0	0	0.837	0.343	0.108	0.05	0	48.7	6.8	18.5	34.7178
B		SVM	y	SM even	0.25	0.025	0.835	0.982	0.375	0.075	4.5	90.3	12.5	35.8	82.2151
B		J48	y		0.2	0.012	0.83	0.983	0.167	0.025	2.3	90	4.3	32.2	81.7891
B		J48		SS 1	0.111	0.333	0.859	0.416	0.091	0.363	16.7	56.1	14.6	29.1	40.4686
B		J48		SM even	0.08	0.049	0.829	0.889	0.056	0.038	6.1	85.8	4.5	32.1	74.4409
B		k-NN		SS 1	0.163	0.703	0.774	0.26	0.182	0.068	26.5	38.9	9.9	25.1	30.8036
B		SVM lin		SS 1	0.094	0.235	0.842	0.416	0.097	0.425	13.4	55.7	15.8	28.3	40.1491
B		SVM lin		SM even	0.083	0.012	0.832	0.965	0.25	0.075	2.1	89.4	11.5	34.3	80.7242
B		Bayes		SM even	0.085	0.506	0.839	0.323	0.082	0.163	14.6	46.6	10.9	24	32.4814
B		k-NN		SM even	0.106	0.531	0.891	0.283	0.094	0.338	17.7	43	14.7	25.1	30.8839
B		J48	y	SS 1	0.087	0.185	0.839	0.59	0.118	0.325	11.8	69.3	17.3	32.8	53.2481
BoW	y	SVM			0	0	0.829	1	0	0	0	90.7	0	30.2	82.8541
BoW	y	SVM		SS 1	0	0	0	0	0.085	1	0	0	15.7	5.2	8.5197
BoW	y	SVM		SS 2	0	0	0.832	0.943	0.143	0.1	0	88.4	11.8	33.4	79.0202
BoW	y	SVM		MC 10	0.099	0.63	0.919	0.147	0.089	0.338	17.1	25.3	14.1	18.8	20.4473
BoW	y	SVM		MC 5	0.059	0.012	0.832	0.955	0.138	0.05	2	88.9	7.3	32.7	79.6592
BoW	y	SVM		MC 7.5	0.084	0.173	0.845	0.667	0.145	0.288	11.3	74.6	19.3	35.1	59.2119

Table B.5: Additional experimental results (second 2-class problem)

Thes.	**TH**	**Features**	**Classifier**	$\pi^{\textbf{buy}}$	$\rho^{\textbf{buy}}$	$\pi^{\textbf{sell}}$	$\rho^{\textbf{sell}}$	$F_1^{\textbf{buy}}$	$F_1^{\textbf{sell}}$	F_1	A
y	0.30%	NE	SVM	0.476	0.123	0.493	0.863	19.5	62.8	41.2	49.0683
y	0.30%	POS	SVM	0.469	0.568	0.444	0.35	51.4	39.1	45.3	45.9627
y	0.30%	POS	Bayes	0.544	0.531	0.537	0.55	53.7	54.3	54	54.0373
y	0.30%	NE	SVM	0.537	0.444	0.521	0.613	48.6	56.3	52.5	52.795
y	0.30%	BoW	Bayes	0.528	0.58	0.528	0.475	55.3	50	52.7	52.795
	0.30%	BoW	Bayes	0.542	0.549	0.491	0.485	54.5	48.8	51.7	51.8443
	0.50%	BoW	Bayes	0.54	0.587	0.386	0.342	56.3	36.3	46.3	48.0874
	0.50%	POS	Bayes	0.595	0.635	0.472	0.43	61.4	45	53.2	54.6448
	0.50%	POS	SVM	0.575	1	1	0.025	73	4.9	39	57.9235
	0.50%	POS	Bayes	0.565	0.624	0.511	0.451	59.3	47.9	53.6	54.3093
y	0.30%	POS	Bayes	0.53	0.543	0.526	0.513	53.6	51.9	52.8	52.795
y	0.30%	B	Bayes	0.57	0.654	0.588	0.5	60.9	54	57.5	57.764
y	0.30%	B	J48	0.618	0.679	0.639	0.575	64.7	60.5	62.6	62.7329

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Masterarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Mannheim, den Unterschrift