CHALMERS

Interday news-based prediction of stock prices and trading volume

Master's thesis in Engineering Mathematics

CHRISTIAN SÖYLAND

Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2015

Master's thesis 2015:45

MASTER'S THESIS IN ENGINEERING MATHEMATICS

Interday news-based prediction of stock prices and trading volume

CHRISTIAN SÖYLAND

Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2015

Interday news-based prediction of stock prices and trading volume
CHRISTIAN SÖYLAND

Master's thesis 2015:45
ISSN 1652-8557
Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Chalmers Reproservice
Göteborg, Sweden 2015

Interday news-based prediction of stock prices and trading volume
Master's thesis in Engineering Mathematics
CHRISTIAN SÖYLAND
Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology

Abstract

This thesis investigates the predictive power of online news on one-day stock price up or down changes and high or low trade volume of 19 major banks and financial institutions within the MSCI World Index, during the period from January 1 2009 to April 16 2015. The news data correspond to news articles, press releases, and stock exchange information, and were obtained by a web-crawler, which scanned around 6000 online sources for news and saved them in a database. The news are partitioned and labeled into two classes according to which price change class, or trade volume class, it corresponds. A supervised automated document classification model is created and used for prediction.

The model does not succeed in predicting the one-day stock price changes, but the percentage of correctly labeled documents in the one-day trade volume experiment was 78.3%, i.e. a classification accuracy of 78.3% was achieved, suggesting that online news does contain some valuable predictive information.

Keywords: stock price prediction, document classification, text mining, trade volume prediction, financial prediction, news analytics

Acknowledgements

I would like to thank Mattias Wahde, David Fendrich, and Claes Ekman for their guidance and feedback during the project.

I would also like to thank Mikael Isik and Tobias Färdig for helping me with the Aitellu database API and Python.

Lastly, I would like to thank Gustav Ljungqvist and Johan Havås for their support in various ways.

Contents

1 Introduction

This thesis covers the related work, theory, implementation, and performance of a supervised automated document classification model[1] used to predict one-day up or down stock price movements and high or low trade volume, from online news. Instead of predicting the actual values of stock price return and trade volume, the data are partitioned into two categories, or classes, where one class corresponds to high, and the other class corresponds to low. The predictive model uses machine-learning in such a way that no manual investigation of the text data is needed. Instead, the model uses statistical tools to find the potential patterns between the text data and stock data. The first chapter covers the related literature within this area, as well as some general information about market theory and stock price prediction. The second chapter covers the theory behind the actual text document classification model, which was created and used for prediction. The third chapter covers the methodology and implementation of the model, and how the experiments were carried out. The fourth chapter covers the results and analysis of the experiments. The fifth and last chapter covers the conclusions and future work.

The project has mainly been conducted together with Aitellu, but also with guidance from the Department of quantitative strategies at the Second Swedish National Pension Fund.

Aitellu's main business lies in analyzing both internal and external big data, which arises naturally in today's business environment, for its customers. The difficulty lies in finding the relevant information hidden in the data. Since the volume of the data is very large, this task cannot be efficiently executed by humans. Instead, Aitellu uses genetic programming, predictive analysis, and artificial intelligence to find these market anomalies and making them an advantage for their customers.

The Second Swedish National Pension Fund is managing approximately 265 billion SEK of the financial buffer within the Swedish pension system. The Department of quantitative strategies uses mathematical tools to maximize the long-term relative return on these pension assets.

1.1 Aim

The aim of this thesis is to investigate if web based news can predict interday stock price movements and trade volume. Usually, quantitative capital management deals with data mining, where stock price prediction is based on past prices and other quantifiable factors. The methods are mathematical and commonly performed by computers. This thesis aims to use machine-learning based methods to evaluate the predictive power of text data, which is usually more unstructured and fuzzy than the numerical data.

1.2 Limitations

The news article data does not contain time stamps with minute precision and may differ in a couple of hours. However, since no intraday price changes will be used, this will not affect the outcome significantly.

[1]In supervised automated document classification, the task is to assign pre-defined class labels to unlabeled documents, automatically. The procedure is called supervised because when training the model, i.e. building up decision rules, the model receives feedback from correctly labeled documents. Unsupervised document classification is more commonly called document clustering. In the remainder of this thesis, document classification will refer to supervised automated document classification.

2 Literature review

The purpose of this thesis is to examine the predictability of news on stock prices. This section covers the major preliminaries within this field and is organized so that the first subsections give a short description of text mining, data mining, the stock market, and stock price prediction. The subsection 2.4 is the bulk of the literature review and covers the main work within text mining for market prediction. The last subsection summarizes the literature review.

2.1 Data mining and text mining

Data mining and text mining are the analytic part of the more abstract field: *knowledge discovery in databases*, which is the art of extracting valuable and implicit information from large volumes of data [34], [19], [17]. The amount of information contained in today's society is huge. It is estimated that the amount of information in the world doubles every 20 months [19]. The problem is that the usable information is hidden in the increasing volume of digital data. Therefore, the need for sophisticated data mining and text mining techniques to discover this knowledge has increased rapidly over the past years [17]. The classical method of data analysis relies on analysts becoming familiar with the data and serving as an interface between the data and the actual knowledge within the data. This procedure is both expensive and unreliable when handling large volumes of data. Hence, data mining and text mining techniques are used today in a lot of different areas such as marketing, finance, health care, retail, fraud detection, manufacturing, etc. [19].

The difference between text mining and data mining is that text data are often unstructured, whereas classical (numerical) data are structured. As the name implies, structured data must be prepared in a special way. The unstructured text data are often a collection of documents, with no specific requirements of how it is presented [70]. The first goal of text mining is thus to represent the documents in a structured way. The basic idea is to create numerical vectors where the dimensionality of the vector is the number of words in the document collection. Now, each document can be represented as a vector, called a feature vector, where each element is either 0 or 1. Here 1 represents the occurrence of a particular word, i.e. feature, and 0 represents the absence of that particular word. Alternatively, the vector elements can be natural numbers, representing the number of occurrences of a particular word in the document, or some other frequency measure [70]. Since the amount of distinct words occurring within a collection of documents can be very large, feature selection (e.g. the chi-squared method) and dimensionality reduction (e.g. random projection) is often needed [5]. Once the unstructured text data are represented by numeric vectors, machine-learning techniques can be applied.

2.2 The stock market

One of the most well-known theories about the stock market is the efficient market hypothesis (EMH), which states that the stock market is efficient. This means that the price of a stock fully reflects all available information of that particular stock [15]. If this were the case, no prediction of stock prices would be possible. This hypothesis is widely discussed, and many studies come to different conclusions [61], [39]. The EMH can be broken down into three forms; weak, semi-strong, and strong [15]. In weak form, the information set is only past prices. If this held, no technical analysis (i.e. prediction based on the price series itself) would be able to consistently generate excess return. In semi-strong form, the information set is the historical prices as well as all public information. This includes additional trading data such as trade volume, and also fundamental data such as sector specific information, or company specific information. If this held, technical analysis as well as fundamental analysis would not be able to consistently generate excess return. In strong form, the information set is all information, i.e. also insider information. A recent study[1] by Lee *et al.*, concludes that stock prices are stationary, which is inconsistent with the EMH [31]. Another theory, which is consistent with the EMH, is the random walk theory. The random walk theory states that stock price movements follow a random walk, i.e. they cannot be predicted [38]. The debate about the accuracy of the EMH has been going on for a long time, and for every paper that supports it, there seems to be one that does not. Hence, recent research have produced a counter-theory, by the name of adaptive market hypothesis (AMH). This theory tries to reconcile EMH with behavioral finance [37]. A recent study made a major empirical investigation and found evidence

[1]They test the stationarity of stocks in 32 developed, and 26 developing countries.

that supports the AMH [66]. In this study, it was found that long-term linear dependencies vary over time, but nonlinear dependencies are strong throughout. This finding suggests that stock prices can be predicted. The question is, how?

2.3 Stock prediction

There are two main approaches to stock price prediction; fundamental and technical. In technical analysis, only the stock price series itself is used to predict the future price [12]. In fundamental analysis, macroeconomic information, information about the sector which the company is in, or information about the company itself, is used to predict the stock price [65]. Fundamental analysis is often more related to fuzzy, or unstructured, data such as financial reports, than technical analysis, which strictly uses past price data.

2.4 Text based stock prediction systems

Klein and Prestbo [28] made one of the first systematic studies about the impact of textual information on financial markets. Even though their study suggests that news and financial movements are related, the study has been criticized since very few stories are used in each day [14]. However, this was the beginning of the field that could be called *text mining for market prediction*. Today, this area of research is still new but very promising. It connects the three disciplines *linguistics*, *behavioral economics*, and *data science* [46].

There have been many other studies that suggest that news affects the stock market [48], [6], [58], [9], [63], [4]. In the following part of this section, some of the most relevant automated text-based prediction systems in finance, with respect to this thesis, will be summarized. Since the area of text-based stock prediction systems is new, and growing fast, the following studies are presented separately, in chronological order, in an attempt to give the reader a good sense for each study, how the area has evolved in the past years, and also where it is going.

In 1998, Wuthrich *et al.*, presented the first online market prediction system that was based on textual web data. They predicted the daily closing values of the Nikkei 225 (NKY), Dow Jones Industrial Average (DOW), Financial Times 100 Index (FTSE), Hang Seng Index (HIS), and Singapore Straits Index (STI). They used keyword tuples that were said to influence stock markets and measured the frequency of these tuples each day. The tuple frequencies where used to create feature vectors. They generated probabilistic rules, used k-nearest neighbors[2] (k-NN), with $k = 9$, and feed forward neural network (FFNN) with back propagation for classifier learning. The average performance of their systems was 43.6%, which is above random guessing (33%). They also made a simple trading strategy that significantly outperformed the indices. They conclude that rule-based learning works best compared to FFNN and k-NN [71].

Lavrenko *et al.*, created a system called AEnalyst, which identified news stories that influenced the behavior of financial markets by correlating the content of news stories with trends in financial time series. They used piecewise linear fitting to identify the trends in the time series. AEnalyst is based on a more generalized system by Fawcett and Provost called activity monitoring [16], which involves monitoring a stream of data and issuing alarms for positive activity in the stream. Lavrenko *et al.*, collected 38469 news articles for 127 stocks in the period October 15 1999 to February 10 2000. They rank articles by the likelihood that they would be followed by a trend. The trends are created using a t-test based top-down procedure on the time series, where the significance of each trend is characterized by the regression statistics slope and R^2. Trends are then subjectively discretized using a simple binning procedure, where trend segments with slopes greater than or equal to 75% of the maximum observed segment slope are labeled *surge*. Greater or equal to 50% equal *slight+*. Analogously, the negative slopes were named *plunges* and *slight-*. They associated a document with a trend if its time stamp is h hours before the beginning of the trend. Values of h from 5 to 10 hours worked best. A naive Bayesian classifier was used to determine which words were significant to a trend. The performance of their system was measured by detection error tradeoff (DET) curves. As an example, they get 10% recall with false alarm rate around 0.5%.[3] They also present a simulated trading approach with significantly better profit than a

[2]The k-NN algorithm is a non-parametric classification algorithm used in pattern recognition [51].
[3]This means that 10% of the articles following a correct trend will be captured, and 0.5% of these articles will be wrong, or miss-classified by the algorithm.

randomized system [30].

The same year, Thomas and Sycara presented a study where they integrated genetic algorithms and text learning for financial prediction. They used two approaches; maximum entropy text classification for stock price prediction based on whole body of text, and genetic algorithms to learn rules based solely on numerical data (trading volume, number of messages posted per day, and total number of words posted per day). Both methods generate excess return, but a combination of the two worked best. The output from their system is up and down predictions, which they map on a trading strategy. The data they used were posts from the discussion boards on the financial website www.ragingbull.com, from January 1 1999 to December 31 1999, together with daily closing values of 22 stocks. They used the text produced on day t to predict whether the closing value of day $t+1$ would be higher or lower. For the text classification, they used rainbow package. The techniques are not presented in the report, but the rainbow package supports different learning techniques such as naive Bayes and k-NN to name a few [42]. The algorithm generates up probability, which they use to determine buy or hold positions in a simulated trading strategy. They measure their performance by excess return[4]. The performance is not very good unless they only use the stocks with more than 10000 posts, which are 12 stocks. In this case their trading strategy generates 6.9% excess return [64].

In 2001, Gidofalvi used news articles to predict intraday stock price movements (up, down or unchanged) of 12 stocks relative to an index. They use naive Bayes to compute the probability of every new stock specific news article belonging to one of the movement classes. Similar to Thomas and Sycara, Gidofalvi also uses the rainbow package [42], specifically the naive Bayes classifier with Witten-Bell smoothing method and uniform prior. Their data consist of around 4500 news articles in the training set, and 1500 articles in the test set. Stop word removal, stemming and feature selection based on mutual information were used in the text processing step. They find strong correlation between news articles and stock price 20 minutes prior and 20 minutes after the news article become publicly available, but that the predictive power is low, which according to them could be an effect of their way of modelling the index price [21].

Koppel and Shtrimberg constructed an automated news story labeling system according to changes in stock price. They align news stories posted on day t, with the changes in price from close of day $t-1$, to open on day $t+1$. They also did alignment of news stories on day t, with stock price change from day $t+1$ to $t+2$. They defined the change as being positive if the price increased by more than 10%, and negative if the price decreased by more than -7.8%. The reason for these high values was to not solely capture noise in the stock price, but the actual impact of new information. The feature selection was straight forward, using only words appearing at least 60 times in the corpus. They represented each text as a binary vector reflecting whether a feature was present or not in the story. Previous work indicates that presence information is superior to frequency information [50]. They further decreased feature dimensionality by selecting the 100 features with the highest information gain (IG). They use linear support vector machine (SVM), naive Bayes, and decision-tree, which all yielded similar results. With linear SVM and 10-fold cross validation, 70.3% accuracy was achieved for the first approach. The second approach yielded slightly over 52% accuracy [29].

Fung, Yu, and Lu investigated the immediate impact on stock prices from news articles based on the EMH, i.e. intraday stock prediction. The time series segmentation is done with a t-test based split and merge algorithm. They use chi-squared estimation on the keywords from the whole set of news to select the relevant ones. The documents are represented by a term frequency - inverse document frequency (tf-idf) weighting scheme and normalized to unit length. They try some different alignment approaches and use an SVM for classifying the articles into positive, neutral or negative stock price trend. They simulate a trading strategy and conclude that their system significantly outperform a randomized system at the 0.5% level [20].

In 2006, Schumaker and Chen developed the AZFinText System, which is a machine-learning based prediction system for stock prices, using financial news articles. They use three different textual representations; bag of words, noun phrases, and named entities. Their data set consist of 9211 financial news articles, and over 10 million stock quotes from the S&P 500 Index, collected during a 5-week period, generated on a 1-minute basis. Their goal was to estimate the stock price 20 minutes after a news release. Two research questions were formed; 1: *How effective is the prediction of discrete stock price values using textual financial news articles?* 2:

[4]The excess return measure is in their case the excess return you get from using their trading strategy instead of just longing the stocks. They also do not incorporate trading costs.

Which combination of textual analysis techniques is most valuable in stock price prediction? They examine the predictive power of only news articles, as well as news articles together with the stock price during article release. The best result was achieved using an SVM derivative called support vector regression (SVR) with both stock price and news articles as input. Independently of which text representation was used, this method consistently outperformed their benchmark method, which were linear regression estimate on stock price alone. They conclude that named entities was the best text representation method [56]. In a later study [57], the authors compare their system against existing quant funds and human stock pricing experts. The system uses sector-based training, which means that the news and stock prices are partitioned into sectors. Their sector-based prediction yielded mean squared error (MSE) score of 0.1954, predicted directional accuracy (DA) of 71.18% and a simulated trading return of 8.50%, compared to 5.62% for the S&P 500 Index. Their system performed better than many trading experts and professional quant funds. While AZFinText's success come from mainly financial news and stock quotes, most quant funds use sophisticated mathematical models on large sets of financial data. Schumaker and Chen believe that their research helps identify a promising research direction for future work. Mittermayer and Knolmayer constructed a similar system but used press releases instead of news articles [44].

Falinouss made an extensive research in the area of stock price prediction with news articles. He used stock price data from the Tehran Stock Exchange for the 20 most active stocks during the years 2004 and 2005, and collected time stamped news articles from various web pages for the Iranian companies, although he only found one stock (Ihran Khodro) with enough news data. He bases his research on the EMH and perform intraday stock price prediction by aligning news data with the trend (up or down) occurring at the same time the news is released. He uses a segmentation algorithm on the price data to produce 1811 trend segments (of which only 429 are aligned with news) during the two years. He mentions a lot of different document representation and feature selection techniques but argues that the chi-squared method is one of the better ones for his application. The documents are classified into a drop or rise category. He uses tf-idf term weighting in the vector space model used for document representation. The weights are normalized with cosine normalization to take into account the size differences between documents. He further reduce the dimensionality of the vector space model with random projection matrices, which projects the vectors into a lower dimensional space. The dimensionality is reduced from 4839 to 200. Python is used for text preprocessing, and R is used for time series segmentation. An SVM with radial basis function (RBF) kernel is used to learn the relationship between news stories and the up and down trends of the stock prices. The SVM is implemented in R with package e1071. The model predicts 1.62 times better than random labeling, i.e. an accuracy of 83% [14].

Tang, Yang, and Zhou combine news mining with time series analysis to forecast interday stock prices. Their result indicated that the algorithm improved classical time series analysis in stock pricing forecast significantly. Each article is represented as a vector, where the features are the most meaningful words. They train an SVR on the feature vectors and stock price changes. To avoid sparseness in related news data, they choose three typical indices to be the forecasting destination, namely energy sources, computer and telecom, and real estate. They conclude that their combined news and time series algorithm outperforms moving average (MA) based time series algorithm. They also perform trend prediction that also outperforms the MA algorithm. The error rate for their combined algorithm was on average (over the three sectors) 37%. They conclude that news reports can provide additional information for stock price forecasting [62].

Kaya and Karsligil investigated the impact of news articles on stock price movement in the year of 2009. They used the Microsoft Stock on the New York Stock Exchange (NYSE), and a total of 982 articles. They aggregated all articles occurring on the same day. The articles released on day t, were labeled according to the *delta price* that day, i.e. $P_t - P_{t-1}$. They tried two feature representation techniques; single words, and word couples consisting of a noun and a verb. For example, the sentence *X company's sales increased by 20 percent from last year*, generates the pairs (X,increased), (company,increased), (sales,increased), etc. After carefully considering a lot of different feature selection methods, the chi-squared feature selection method was used. An SVM was used to train the classification algorithm. Different kernels were tried; linear, polynomial, and RBF. They also tried different feature count parameters; 100, 250, 1000, 5000, and 10000. The best result was obtained with word couples, RBF kernel, and 250 as feature count parameter, which yielded 61% precision and 87% recall. They conclude a strong relationship between news and stock price movements and argues that the system can be improved further. For example, some of the articles in the data set may relate to other companies, but within the same sector [27].

In 2010, Hafez constructed market- and industry-level sentiment indices based on a bottom-up approach for company-specific news events and their corresponding sentiment. A 90-day trailing window was used to create the indices, which were used to simulate a trading strategy taking 1-month long and short position in the market or the industries. He also combined the strategies. He only consider companies in the S&P 500 Index and select relevant news for each company. The data are obtained from RavenPack, which is a news analytics tool. He uses several metrics such as event sentiment score (ESS), event novelty score (ENS), and company relevance score (CRS), which are measures about the sentiment, novelty, and company relevance of the news events. He creates a trading strategy based on the indices that deliver double-digit positive returns in out of sample testing during the period May 2005 to December 2009. The industry strategy is market neutral and takes long (short) positions in the industries performing the best (worst). He finds that news sentiment is a better predictor than one-month price momentum when predicting future returns of the S&P 500 Index [22]. In a later study by Hafez and Xie, they conclude that market sentiment, i.e. news beta, is uncorrelated with traditional quant factors such as exposure to market, size, value, and momentum. The evidence in their research suggests that exposure to market sentiment is still an untapped source of alpha in financial markets [23].

In 2012, another automated stock price prediction system based on financial news was presented by Hagenau *et al.*. In their research, they compare different text preprocessing techniques. Their findings suggest that word combination methods (2-gram, noun-phrases, and 2-word-combination), together with SVM classifier and chi-squared feature selection, are the best approaches. The data set used comprises corporate announcements from Germany and the United Kingdom (UK), published between 1998 and 2010. They use daily abnormal return on the event day, i.e. the day the news was published and label the news as up or down according to the stock price change. They received 65% accuracy as best with 2-word combinations (2-gram) when testing the methods [24].

2.5 Summary

The level of interest in text mining for financial applications is increasing rapidly along with the increasing amount of available text data. The field is still in its infancy, and this chapter has covered the main contributions within the field so far. There is strong evidence that news and stock prices are in fact correlated. Several studies suggest that the information contained in news data can predict changes in intraday stock prices, but few succeed in performing well on interday level. Furthermore, most studies classify news articles according to if the article is followed by a price change of some sort, instead of aggregating all news in some fixed period, and using the news within this period to predict the following stock change. The majority of studies use price classes, e.g. up and down, to classify the news texts, but some systems use continuous prediction of the stock price. The most used machine-learning classifier is SVM. There are some systems that use decision-tree based learning [67], [55], but few that use random forest, which have become increasingly popular in recent years, since they have been proved to perform very well against other machine-learning methods and is also robust against overfitting [7]. Another advantage with random forests is the support of thousands of input variables [36], which most other machine-learning algorithms used in news analytics does not have. Instead, the use of feature dimensionality reduction techniques need to be applied, where the most used dimensionality reduction technique is the chi-squared method. Other dimensionality reduction techniques, e.g. random projection, are also used. The features most commonly used are words or 2-gram, and the feature representation techniques are mostly binary or tf-idf. In most cases, the features are learned from the corpus, but there are also studies where a pre-made feature set, designed by domain experts, is used. Since machine-learning methods are becoming more and more popular, and the amount of available information in text form is rapidly increasing, the future of news analytics in finance seems bright.

3 Theory

This chapter covers the document classification theory used in this study. The objective of document classification is to assign class labels to unlabeled text documents. The most common approach is to train a machine-learning algorithm with a set of labeled text documents [71], [16], [21], [50], [20]. This approach requires the text documents to be processed into numerical vectors or, more precisely, feature vectors before the training step, i.e. each document is represented by a feature vector. Since the number of common words used in the English language is around 50000 [33], and the number of distinct words occurring in a collection of documents is often even higher, only the most informative words or word combinations can be used when creating features. Otherwise, the vector space generated by those features, would be of too high dimensionality.

The document classification process can thus be broken down into four steps. First, text preprocessing, where noisy text and language dependent factors are eliminated as much as possible [69]. Second, document representation, where features are selected to represent the documents, and a feature vector space model is created. Third, dimensionality reduction, where the feature vector space dimensionality is reduced. Fourth, machine-learning, where the set of all feature vectors is used to train a machine-learning algorithm. The first four subsections within this chapter covers these four steps respectively. The last subsection covers the theory of how to evaluate the performance of the document classification model.

3.1 Text preprocessing

Text preprocessing is the transformation of raw text into a collection of words. According to [32], the process consists of tokenization, stop word removal, and stemming. However, there is no explicit definition of exactly which procedures should be included in text preprocessing. In this study, lemmatization, which is closely related to stemming, will also be covered.

Tokenization is the transformation of a stream of characters into tokens, which are words, phrases, or other meaningful elements of the text. A token is an instance of a type, and the defining properties of a type can differ depending on the context. When classifying Twitter, or social media text documents, punctuations, colons, parentheses, exclamation marks, or other non-letter characters can hold important information about the sentiment of the text (e.g. a smiley) and should therefore be a part of the token, whereas when classifying news articles, the purpose of these non-letter characters may not be as important. Hence, there are no distinct rules of how the tokens should be generated from the character stream [70].

As an example, the hypothetical text document,

{*'Company ABC's earnings rose by about 15% last year!!. It is a remarkable rise :)'*},

could be tokenized into the following set of tokens,

{*'company','abcs','earnings','rose','by','about','last','year','it','is','a','remarkable','rise'*}.

In this example, capital letters are transformed into lower letter characters, apostrophes are removed and non-letter characters are replaced by blanks. A token is defined to be one or more characters in between two blanks.

In stop word removal, words with low predictive capability, such as articles and pronouns, are removed to reduce the number of types in the text document. The collection of stop words has to be supplied manually, but are easily found free of charge online [2].

Continuing the example, the set of tokens after stop word removal, would be,

{*'company','abcs','earnings','rose','last','year','remarkable','rise'*}.

Stemming consists of transforming words into their word stem. When the process consists of regularizing grammatical variants such as present or past and singular or plural, it is called inflectional stemming [70]. For example, *wait*, *waiting*, and *waited* each has *wait* as their word stem. There are also more aggressive stemming algorithms that transform the word to a root form, which is not necessarily a correctly spelled word. One of the most well-known such stemming algorithms is the Porter stemmer, which for example stems the word *replacement* to *replac*. This kind of aggressive stemming is advantageous in applications such as automated text classification, since it can reduce the size and complexity of the number of types in the dictionary, making distributional statistics more reliable [52], [70].

The last part of the text preprocessing is lemmatization, and the purpose of this step is the same as with stemming; to reduce the number of types in the document by representing different words with similar meaning, by the same word. The problem with stemming is that it only removes, or chops off, a part of the original word. Lemmatization instead uses a vocabulary to map different types of a word into the same type. For example, *am*, *are*, and *is*, are transformed to *be*. Hence, lemmatization finds the base form or, more precisely, the lemma of a word. Hence, using lemmatization, a natural language processing (NLP) procedure that performs full morphological analysis of the text, can solve some problems arising from the more pragmatic method; stemming [40].

Continuing the example, the set of tokens after stop word removal and lemmatization would be,

{*'company', 'abcs', 'earnings', 'rise', 'last', 'year', 'remarkable', 'rise'*}.

Note that the type *rose* was lemmatized to *rise*. Hence, two tokens of the type *rise* occur in the text, but there is only one type associated with *rise*.

3.2 Document representation

After the text preprocessing step, each document is represented by a collection of tokens. The objective of document representation is to represent each document as a numerical vector, called feature vector, which lives in a Euclidian vector space. This model is called the vector space model [8]. An intuitive way of representing the documents as feature vectors, would be to consider the global set of types, which is called a dictionary, and let each axis in the vector space correspond to a distinct type. If the weighting of the feature vectors corresponds to the number of instances (tokens) of each type, the above approach is called the *bag of words* approach [70]. This method can be extended to word combinations, i.e. n-grams. Continuing the example, if $n = 2$, instead of having {*'earnings'*} and {*'rise'*} as two separate features, their combination {*'earnings rise'*} would also be a feasible feature. There are many other multi-word document representation schemes such as *named entities*, *proper nouns*, *noun phrases*, etc. [70], [74]. However, there are studies that suggest that the simple document representation schemes such as *bag of words* often perform just as good as the more advanced ones derived from NLP [45].

As mentioned, the vanilla version of the *bag of words* approach consists of weighting each element in the feature vector, according to the amount of tokens occurring in the document of the corresponding feature's type. Continuing the example, the *bag of words* approach would generate the feature vector,

{*'company', 'abcs', 'earnings', 'rise', 'last', 'year', 'remarkable'*},

with elements (weights),

{*'1', '1', '1', '2', '1', '1', '1'*},

since there are two tokens of the type *rise*, and only one token occurring from each of the other types, respectively. Note that if there were more text documents in this example, there would be a lot of zeros in the feature vectors, since the vector space axes are the global set of types occurring in all of the documents.

Apart from *bag of words*, there are other weighting schemes, such as tf-idf. This method modifies the count of a word by the perceived importance of that word [70]. The formula for the tf-idf weight can be found in the following equations,

$$\textit{tf-idf}(j) = \textit{tf}(t) \times \textit{idf}(j), \tag{3.1}$$

$$\textit{idf}(j) = \frac{N}{\textit{df}(j)}, \tag{3.2}$$

where *tf-idf*(j) is the tf-idf value of term j, *tf*(j) is the term frequency of term j, i.e. the number of times the term occurs in the document, *idf*(j) is the inverse document frequency of term j, *df*(j) is the document frequency of term j, i.e. the number of documents containing the term j, and N is the total number of documents.

Another simple, but effective, way of weighting the feature vectors is by using binary representation, where instead of a frequency value, a binary value of 1 or 0 is used, indicating the presence or absence of the feature within the document, respectively [29], [44]. As is also stated in the literature review, [50] suggests that presence information is sometimes superior to frequency information.

3.3 Dimensionality reduction

After the document representation step, each document is represented by a vector called feature vector, which lives in a Euclidian vector space. The problem with this model is that even after the domain dependent dimensionality reduction methods such as stop word removal and stemming or lemmatization, the number of features, i.e. dimensions, in the vector space is too big. Hence, domain-independent dimensionality reduction methods such as feature selection and feature extraction can be applied to further reduce the dimensionality of the vector space model needed for machine-learning [49].

The difference between feature selection and feature extraction is that feature selection reduces the dimensionality by selecting a subset of the original feature set, i.e. removes some of the axes in the vector space, while feature extraction reduces the dimensionality by linear algebra transformation of the original vector space into a more effective lower dimensional subspace [33], [72], e.g. by means of principal component analysis (PCA) [26] or linear discriminant analysis (LDA) [41]. Hence, in feature extraction, the new set of features are not of the same type as in the original feature set [49]. Since the large amount of features in document classification yields a very high dimensionality in the vector space model, feature extraction methods, even though proven to be very effective, often fails due to their high computational cost [72]. Hence, only feature selection will be used and covered in this study.

3.3.1 Feature selection

In document classification, the most used dimensionality reduction method is feature selection [49]. One of the reasons for this is that the selected features retain physical meaning [35], which is not the case for feature extraction. Feature selection can be carried out as a part of the machine-learning algorithm (the wrapper approach), or as an independent step prior to machine-learning (the filter approach) [49]. This thesis will only cover the later of the two.

The goal of feature selection is to select a subset from the original feature set, such that the subset is of lower dimensionality than the original feature set, but the least amount of relevant information is lost. If the cardinality of the original feature set is P, the individual best features approach selects the $p << P$ best features, according to a quality measure [33]. In supervised document classification, the quality measure of a feature reflects how good the feature is in discriminating between classes [49], i.e. the correlation of each feature and the class label [35]. There are several different quality measures, e.g. chi-squared, information gain, odds ratio, mutual information, correlation coefficient, etc. [18], [68], [59]. One of the most common choices in supervised learning is the chi-squared measure [68], and several studies suggest empirical evidence that this measure outperforms other measures, as well as being more computationally effective [73], [14], [47], [13], [60], [53].

The chi-squared feature selection method is based on the chi-squared test of independence from statistics, which tests the independence of two events. The two events A and B are defined to be independent if

$P(A \cap B) = P(A)P(B)$. In chi-squared feature selection, the event A corresponds to a term occurring, and the event B corresponds to a class occurring [40]. The chi-squared statistic is calculated by creating, for each term and class, a contingency table as in Table 3.1, and then plugging in the values in equation 3.3.

	# Documents having term j	# Documents not having term j
Class = 1	N_{11}	N_{10}
Class ≠ 0	N_{01}	N_{00}

Table 3.1: *Contingency table for the chi-squared feature selection method.*

$$\chi^2(j,c) = \frac{(N_{11}+N_{10}+N_{01}+N_{00}) \times (N_{11}N_{00} - N_{10}N_{01})^2}{(N_{11}+N_{01}) \times (N_{11}+N_{10}) \times (N_{10}+N_{00}) \times (N_{01}+N_{00})}. \tag{3.3}$$

The chi-squared (χ^2) statistic measures the difference between observed and expected counts and is based on the hypothesis that the term and class are independent. The higher the value of the chi-squared statistic, the more reason to doubt the hypothesis that the term and class are independent. Going back to the individual best features approach, the features are ranked with respect to their chi-squared values, and the top p are chosen. Hence, the chi-squared feature selection method selects the p features, out of the total set of P features, with the least amount of independence to the classes.

A weakness of the chi-squared feature selection method is that if there is a low number of observations in the bins, the test will fail to produce a robust measure [11].

3.4 Machine-learning classifier

In general, a machine-learning classifier uses vectors living in the feature space, with known class labels, and builds decision rules according to the relationship between the feature vectors and their corresponding class labels. In this study, a random forest classifier is used. A random forest is an ensemble method, consisting of several decision-trees. Hence, the first subsection covers decision-trees, and the second subsection covers random forests.

3.4.1 Decision-trees

A decision-tree is a machine-learning classifier which partitions the feature space into subspaces, and then fits a simple model in each subspace. The classifier is simple, yet powerful, and works well with data which are under the *curse of dimensionality* [25], which often is the case in document classification. There are several versions of decision-tree classifiers, e.g. C4.5 [54], but in this thesis a popular method called classification and regression tree (CART) is adopted, which will be covered now. Let the training data consist of N feature vectors of dimensionality p, i.e. $\mathbf{x}_i = (x_i^1, x_i^2, \ldots, x_i^p)$, with corresponding class labels y_i, for $i = 1, \ldots, N$. Suppose there are l class labels, i.e. $y_i \in 1, \ldots, l, \forall i = 1, \ldots, N$. To build a decision-tree from this data, the variables which should be splitted, their corresponding split points, and the topology of the tree, need to be determined. To obtain the optimum tree, assume the feature space is partitioned into M regions $R_1, R_2, \ldots, R_M$, and the response variable (class label) is modeled as a natural number $k_m \in 1, \ldots, l$ in each region, i.e.,

$$f(\mathbf{x}) = \sum_{m=1}^{M} k_m I(\mathbf{x} \in R_m). \tag{3.4}$$

Now, one can easily see that the optimum solution for $\hat{k}_m = \arg\min_k \sum (y_i - f(\mathbf{x}_i))^2$ becomes,

$$\hat{k}_m = \arg\max_k \frac{1}{N_m} \sum_{\mathbf{x}_i \in R_m} I(y_i = k), \tag{3.5}$$

where $N_m = \sum_{i=1}^{N} I(\mathbf{x}_i \in R_m)$. Now, to find the partition that minimize the sum of squared errors, a greedy algorithm is used. Suppose variable x^j is split at the point s, and two half-planes are defined as,

$$R_1(x^j, s) = \{\mathbf{x} | x^j \leq s\} \text{ and } R_2(x^j, s) = \{\mathbf{x} | x^j > s\}. \tag{3.6}$$

The splitting variable x^j and split point s is found by solving,

$$\min_{x^j,s}\left[\min_{k_1}\sum_{\mathbf{x}_i\in R_1(x^j,s)}(y_i-k_1)^2+\min_{k_2}\sum_{\mathbf{x}_i\in R_2(x^j,s)}(y_i-k_2)^2\right]. \tag{3.7}$$

The inner minimizations is solved by $\hat{k}_1$ and $\hat{k}_2$, respectively, for any choices of x^j and s. Determining the split point s is very quick for all splitting variables. Thus, the algorithm scan through all of the training data to determine the best split point s for each variable x^j. The algorithm then recursively partition each new region, into two new regions until a stopping criterion is met. This procedure will generate a decision-tree where the outermost partitions are called nodes. The question is when to stop the process since a large tree tends to overfit the data, but a small tree might miss valuable information contained in the data. The most common approach is to build a large tree, and stop creating more nodes when the number of feature vectors in the node is below some number, and then prune the tree, i.e. remove some of the nodes. The pruning used in CART is called *cost-complexity-pruning.* This pruning method minimizes a so-called cost complexity criterion, such that a subtree is obtained. The cost complexity criterion can be constructed in different ways, using different so-called node impurity measures, where the most used node impurity measures are gini index and cross-entropy. The *Cost-complexity-pruning* method, and a deeper description of decision-trees, can be found in [25].

3.4.2 Random forest

Random forest [7], is an ensemble method, i.e. a learning algorithm which takes a finite number of classifiers, and create decision rules based on the weighted average decision rules from these classifiers [10]. In particular, random forest is a modified version of bootstrap aggregation or bagging, which is a technique used in statistics to reduce the variance of an estimated prediction function. The bagging procedure consists of bootstrapping, i.e. sampling with replacement from the full data set, to create a set of samples which are identically distributed but dependent. Random forest then constructs a large number of decision-trees and uses one bootstrap sample for each tree, and averages the decision rules of the trees [25]. As an example, consider B number of decisions from decision-trees, each with a variance of σ^2, which are identically distributed but not necessarily dependent, since they are obtained from bootstrapped samples from the full data set. Let the correlation be ρ. Then, the variance of the average of the B decisions is

$$\rho\sigma^2+\frac{1-\rho}{B}\sigma^2. \tag{3.8}$$

Hence, when B increases, the variance of the average is reduced. However, if the correlation of the trees is big, the benefits of bagging are reduced. The idea in random forest is thus to reduce the variance by bagging. To minimize the correlation between the trees without increasing the variance too much, the input variables to consider when splitting a node is sampled at random from the full set of variables, i.e. instead of scanning through the p variables, a sampled subset of size $p' \leq p$ is used. A typical value for p' is $\sqrt{p}$. The main improvement obtained from using random forest instead of decision-trees, is that they do not tend to overfit the data as much. A deeper description about the theory of random forests is found in [25].

3.5 Evaluation metrics

To evaluate the document classification model, a test set is used. The test set is obtained by partitioning the whole set of documents into a training set and a test set. Each document in the test set has a true label and a predicted label. The number of different class labels can vary, but in this thesis, only the binary decision problem is covered. Hence, the labels are referred to as 0 and 1. The test set needs to be of sufficient size, usually no less than 50 documents [40]. After the classifier has predicted the class labels on the test set, a contingency table can be created with the predicted labels, and the correct labels, as in Table 3.2.

According to [40], the two most used basic performance measures for information retrieval are precision and recall. They are calculated for each class, as in equations 3.9 - 3.12,

$$\textit{precision (1)} = \frac{\mathrm{TP}}{TP+FP}, \tag{3.9}$$

	Predicted	
True	1	0
1	TP	FN
0	FP	TN

Table 3.2: *A contingency table or, more precisely, a confusion matrix for evaluating the performance of the classifier. The rows in the table correspond to the true labels of the test set, and the columns correspond to the predicted labels. TP, FN, FP, and TN are the number of true positives, false negatives, false positives, and true negatives, respectively.*

$$\textit{precision (0)} = \frac{\text{TN}}{FN + TN}, \tag{3.10}$$

$$\textit{recall (1)} = \frac{\text{TP}}{TP + FN}, \tag{3.11}$$

$$\textit{recall (0)} = \frac{\text{TN}}{FP + TN}, \tag{3.12}$$

where TP, FN, FP, and TN are defined as in Table 3.2. Another frequently used measure is the accuracy, which is defined as,

$$\textit{accuracy} = \frac{\text{TP + TN}}{TP + FP + FN + TN}. \tag{3.13}$$

Although accuracy is an intuitive measure, there is a good reason to avoid using it when the data are non-uniformly divided among the class labels, since if 99% of the data belong to class label 0, a classifier that only classifies all documents to class 0, could perform very well according to the accuracy measure. Hence, in many applications, it is favorable to have the two measures precision and recall [40]. To get a single measure that trades off precision versus recall, one can use the so-called F-measure, which is defined as,

$$F = \frac{1}{\alpha\frac{1}{p} + (1-\alpha)\frac{1}{r}} \tag{3.14}$$

where $\alpha \in [0, 1]$ is a parameter, p is the precision, and r is the recall. The F-measure is the weighted harmonic mean of precision and recall, and if the parameter $\alpha = \frac{1}{2}$, it is called the balanced F-measure, which equally weighs precision and recall.

4 Methodology and implementation

The goal of this thesis is to examine the predictability of stock prices and trade volume, from news articles. There are two major differences between this study and most other studies within this field. First of all, no intraday price movements are taken into account. Second, no price trend alignment with news articles is carried out. Hence, the labeled text documents are not single news articles but aggregated news articles. More precisely, a portfolio of stocks belonging to the same industry is created, and all news articles published on a particular day for a certain company, are aggregated to create a text document. This means that a text document is defined as all the text contained in the news articles published on a certain day, for a certain company. As an example, if the portfolio consists of two companies, with news articles published on each day over a period of two years, for both companies, the data set will contain $365 \times 2 = 730$ text documents. The model will not make any distinction regarding the company to which the text document pertains. Hence, an assumption is made: The type of content within a text document that gives rise to a price change is independent of which particular company it belongs to, as long as the company is in the portfolio. In this study, the portfolio consists solely of banks. This means that the model will capture news events that are correlated with stock price changes for banks.

4.1 Data

There are two types of data used in this study; news articles and stock data. The period used in this study spans from January 1 2009 to April 16 2015 and the companies used are found in Table 7.1. All the companies are banks appearing in the MSCI World Index within the period January 1 2009 to April 16 2015. The MSCI World Index represents large and mid-cap companies across 23 developed markets countries [1].

4.1.1 News articles

The news articles are obtained from a proprietary database at Aitellu. In addition to news articles, the entries in the database consist of press releases, blogs, forum posts, Twitter, Youtube, Vimeo, Instagram, Flickr, and Facebook. To obtain this data, Aitellu uses a web-crawler which scans several thousand sources over the internet and saves the relevant content in the database. Each entry in the database is registered with a title, lead title, time stamp[1], body, language, source, and some additional information that are not used in this study. The body is the text that is analyzed. To select the relevant news articles from the database, one search profile for each company is created. Hence, each search profile corresponds to one of the companies in the portfolio. Each search profile is intended to collect all the relevant news articles for the corresponding company. When creating the search profiles, a boolean search is used. In a boolean search, search strings can be combined with boolean operators such as *AND*, *OR*, and *NOT*. The boolean search strings used are found in Table 7.1. Apart from the search strings, the search profiles need settings. The same settings are used for all search profiles and are described now. First of all, the search is only carried out in the title or the lead title of the database entry, e.g. if a search string is apparent in the body of an article, but not in the title or lead title, that article will not be included. Second, only *standard sources* are considered. Thus, the news articles in this study are entries in the Aitellu database that originate from *standard sources. Standard sources* correspond to news media, press releases, and stock exchange information. *Non-standard sources* are forum, Twitter, blogs, etc. For *standard sources* alone, the web crawler scans around 6000 sources over the internet. Third, no distinction between capital letters and non-capital letters is made. Fourth, only entries written in English are considered. After setup, the search profiles are filled[2] with entries with a time stamp within the period between January 1 2009 to April 16 2015. The number of entries found in each search profile is found in Table 7.1.

4.1.2 Stock data

The stock data correspond to daily time series with closing price and trade volume, obtained from the NYSE for the companies in Table 7.1 in the time January 1 2009 to April 16 2015. The data are downloaded from

[1]The time stamp is not the time when the text was published, but when the web crawler saved the text. The time stamp can differ in a couple of hours from the actual publishment.

[2]The profiles are filled automatically with Python and an API supplied by Aitellu.

Yahoo! Finance[3].

4.2 Experiments

Two experiments were carried out. One experiment where the stock price return is predicted, and one experiment where the stock trade volume is predicted. Both experiments were made with a binary document classification approach. This means that instead of predicting the actual values of return or trade volume, the samples were partitioned into two classes, where one class correspond to *high*, and one class correspond to *low*. Python programming language was used exclusively for these experiments. Since the time stamp on the entries in the database correspond to Gothenburg time, they were converted to New York time.

Both experiments boil down to a document classification task, as explained in the theory section. The only difference between the experiments is in what way the text documents were created and labeled. Hence, the first two subsections within this section, describe how the labeled text documents were generated, and the last section describes how the text document classification was carried out, since this was the same for both experiments once the labeled text documents were created.

4.2.1 One-day return

In this experiment, the one-day return was predicted. News articles were aggregated for each company and each day, and closing price return was calculated for each company and each day. The news articles with a time stamp on a weekend, were moved to the last Friday. This is because news occurring on weekends should also be considered when predicting the Monday return. The one-day closing price return was calculated for each company during the mentioned period, and each return was labeled as *high* or *low* according to if the return was positive (≥ 0) or negative (< 0)[4]. The aggregated news articles were then aligned with the stock price labels, such that the aggregated news articles for a certain day and a certain company, got the same label as the closing price return the day after the aggregated news articles time stamp. At this point, as mentioned at the beginning of this chapter, a dataset of labeled text documents were obtained, where a text document correspond to aggregated news articles for a certain company and a certain day. Hence, the task can now be defined as a binary text document classification task, and the methods described in the Theory chapter can be used. The implementation is described in the Text document classification implementation section below.

4.2.2 One-day volume

In this experiment, the one-day volume was predicted. In the same way as described in the one-day return experiment, news articles were aggregated, weekend dates were moved, but instead of using the one-day closing price return for each day, the trade volume was used for labeling the text documents. For each company, the trade volume median was calculated from all trading days within the period. Then each trading day trade volume was labeled as *high* or *low* according to if the traded volume that day was greater than or equal to the median, or lower than the median, respectively.

4.2.3 Text document classification implementation

The first step was to tokenize the text documents. The tokenization consisted of removing all HTML tags such as URL's etc., replacing all characters that are not letters, with blanks, and converting all capital letters to lowercase letters. All stop words in [2] were removed, and the remaining tokens were lemmatized[5] with the WordNet lemmatizer [43]. Next, the data set was divided into a training set and a test set. The whole data set was sorted in chronological order, and the first 90% of the data were used as the training set, and the remaining 10% were used as the test set. Every decision regarding document presentation, dimensionality reduction, and machine-learning, that were based on information within the text documents, was based solely on information contained in the training set.

[3]Pandas io.data package is used for automated data retrieval with Python. In this way, no manual stock data download is needed.

[4]An experiment where returns close to 0 were removed was also tried, without improvement of the results.

[5]Since the meaning of the word is of importance, lemmatization was deemed a more suitable procedure than stemming.

Before representing each text document as a feature vector, only the 50000 (the number of words commonly used in the English language [33]) most frequently occurring words were selected such that tokens that are wrongly spelled words, very unusual words, or simply tokens that are not words at all, were deleted. Also, the chi-squared feature selection method does not work for low-frequency words [11]. After some testing, a 2-gram binary representation scheme was used to represent each text document as a feature vector. Chi-squared feature selection was used as the only dimensionality reduction technique, since it has been proven to yield good performance in text classification, compared to other methods [73], [14], [47], [13], [60], [53]. In the same way as in [33], the individual best features approach was used, i.e. the features were sorted according to their chi-squared values, and the p best ones were used as the new features. In this study, $p = 200$, which was also the feature vector space dimensionality after the dimensionality reduction step.

A random forest classifier was trained with the feature vectors in the training set. After some parameter tuning with grid search, the following parameter settings were used,

Parameter name	Parameter description	Parameter value
Max features	The p' parameter	$\sqrt{p} = 70$
Min sample split	Minimum samples required in node to split a node	8
Min sample leaf	Minimum samples required in node leaves to split a node	4
Node impurity criterion	Pruning parameter criterion	entropy
Estimators	The number of trees in the forest	100

Table 4.1: *The parameter settings used when training the random forest.*

To measure the performance of the text document classification model, the random forest predicted labels for the test set, and precision, recall, accuracy, and F-measure were used. A randomized labeling algorithm was also generated in order to benchmark the model. The randomized model assigned *high* or *low* labels to the documents with equal probability. The results are found in the next chapter.

5 Results and discussion

In this chapter, the results from the two experiments is presented via confusion matrix, precision, recall and F-measure.

5.1 One-day return

In the one-day return experiment, the training set contained 13606 documents, and the test set contained 1512 documents. A 2-gram binary representation scheme is used, and a feature space dimensionality size of 200 after chi-squared feature selection. The results from predicting the test set labels with the random forest classifier can be found in tables 5.1 and 5.2.

	Predicted	
True	*high*	*low*
high	501	343
low	396	273

Table 5.1: *Confusion matrix for evaluating the performance of the one-day return classifier. The columns in the table correspond to the predicted labels of the test set, and the rows correspond to the true labels. The classifier parameter setting can be found in Table 4.1.*

Evaluation metric	Value
accuracy	51.1%
precision (*high*)	59.4%
precision (*low*)	40.8%
recall (*high*)	55.9%
recall (*low*)	44.3%
F-measure (*high*)	57.6%
F-measure (*low*)	42.5%

Table 5.2: *The metrics used for evaluating the performance of the one-day return classifier according to the confusion matrix in Table 5.1.*

5.1.1 Randomized text classification

In order to benchmark the text classification model created, a randomized text classification model is used. The randomized model assigns *high* and *low* labels to the text documents in the test set, with equal probability. The results from the randomized model are found in tables 5.3 and 5.4.

	Predicted	
True	*high*	*low*
high	411	433
low	320	348

Table 5.3: *Confusion matrix for evaluating the performance of the randomized one-day return classifier. The columns in the table correspond to the randomly predicted labels of the test set, and the rows correspond to the true labels.*

Evaluation metric	Value
accuracy	50.2%
precision (*high*)	48.7%
precision (*low*)	52.1%
recall (*high*)	56.2%
recall (*low*)	44.6%
F-measure (*high*)	52.2%
F-measure (*low*)	48.0%

Table 5.4: *The metrics obtained from the confusion matrix in Table 5.3, which correspond to the one-day return randomized classifier.*

5.2 One-day volume

In the one-day volume experiment, the training set contained 13606 documents, and the test set contained 1512 documents. A 2-gram binary representation scheme was used, and the feature space dimensionality size was 200 after chi-squared feature selection. The results from predicting the test set labels with the random forest classifier can be found in tables 5.5, and 5.6.

	Predicted	
True	*high*	*low*
high	683	184
low	145	501

Table 5.5: *Confusion matrix for evaluating the performance of the one-day volume classifier. The columns in the table correspond to the predicted labels of the test set, and the rows correspond to the true labels. The classifier parameter setting can be found in Table 4.1.*

Evaluation metric	Value
accuracy	78.3%
precision (*high*)	78.8%
precision (*low*)	77.6%
recall (*high*)	82.5%
recall (*low*)	73.1%
F-measure (*high*)	80.6%
F-measure (*low*)	75.3%

Table 5.6: *The metrics used for evaluating the performance of the one-day volume classifier according to the confusion matrix in Table 5.5.*

5.2.1 Randomized text classification

In order to benchmark the text classification model created, a randomized text classification model is used. The randomized model assigns *high* and *low* labels to the text documents in the test set, with equal probability. The results from the randomized model are found in tables 5.7, and 5.8.

5.3 Discussion

From the above results, one cannot, unfortunately, conclude that the the interday up or down stock price movements can be predicted from online news. However, there are some factors that could affect the above result. First of all, the web-based news data used in this study is extremely voluminous. The model used in

	Predicted	
True	*high*	*low*
high	437	430
low	332	313

Table 5.7: *Confusion matrix for evaluating the performance of the randomized one-day volume classifier. The columns in the table correspond to the randomly predicted labels of the test set, and the rows correspond to the true labels.*

Evaluation metric	Value
accuracy	[49.6%]
precision (*high*)	[50.4%]
precision (*low*)	[48.5%]
recall (*high*)	[56.8%]
recall (*low*)	[42.1%]
F-measure (*high*)	[53.4%]
F-measure (*low*)	[45.1%]

Table 5.8: *The metrics obtained from the confusion matrix in table 5.7, which correspond to the one-day volume randomized classifier.*

this thesis only uses statistical and machine-learning tools to filter out the irrelevant or noisy data, but no other method is applied for this purpose. For example, one could focus solely on press releases instead of what in this thesis is called *standard sources*, or use a so-called *relevant document selection*, as in [3]. However, the approach of this study has been to use unstructured big data and use automated text-mining methods to filter out the noise, but the results obtained suggests that if the stock price movements are in fact predictable by online news, a better filtering method for relevant articles could improve performance. Another fact that could affect the above result is the aggregation of companies. News events that give rise to a stock price change for one company, may not necessarily give rise to a stock price change for another company, even though the companies are similar in nature (in this study, they are all banks or financial institutions). To conclude, this study does not find evidence that contradicts the EMH.

The second part of the study was to use the same automated text document classification model, but instead using it to predict *high* or *low* occurrences of trade volume. The results from this experiment were far better than the stock price experiment, and all evaluation metrics indicate that the model performs well. The fact that the trade volume experiment yielded good results, suggests that the model in itself is working, but that the potential interday predictability of online news on stock prices is poor. It should also be stated that many different settings regarding feature space dimensionality, document representation schemes, number of class labels, filtering out returns close to 0, weekly price change (instead of daily), stemming, etc. have been tried, without improvement of the model. An SVM classifier was also tested, without improvement.

6 Conclusions and future work

The text classification model uses tokenization, stop word removal, lemmatization, 2-gram binary document representation, and chi-squared feature selection, before using a random forest machine learning classifier for the classification. The text documents consist of aggregated news stories (*standard sources*) for each day and company, yielding a total data set of 15118 text documents. In the one-day return experiment, the labels were created by considering whether the stock price for the corresponding company was *high* (≥ 0) or *low* (< 0) the day after the news were released. In the one-day trade volume experiment, the labels were created by considering whether the stock trade volume for the corresponding company was *high* ($\geq$ *sample median*) or *low* ($<$ *sample median*) the day after the news were released, where the *sample median* is the median trade volume, for each company, in the full dataset.

The model did not succeed in predicting the one-day stock price movements better than a randomized model. However, the performance was good when predicting the trade volume class. The results obtained suggest that the model works, but that there is no strong relationship between online news and one-day stock price movements. Or, if there is a relationship between online news and one-day stock price movements, the text data used in this study are too noisy to find the relationship with the created model.

The area of news analytics, or text mining for financial applications, is young [46], but the potential is big. Instead of predicting stock prices, one could predict movements in market or industry indices. Also, instead of using all the available news content available for the companies considered, one could limit the text data to press releases, quarterly statements, or more company-specific news. An interesting method for filtering out noisy news articles is applied by [3], where they use a clustering algorithm to try to remove news articles that do not correspond to the assigned label. There are also studies that, instead of online news, use twitter [67] or forum [42] posts, which is also interesting approaches. Another possible approach would be to consider central bank press releases to predict interest rates. Lastly, instead of using discrete class labels one could use continuous response variable for each text document, e.g. the actual stock price return, and train a random forest regressor model or SVR. As is pointed out in [23], another interesting area is to examine the role of sentiment in other financial markets such as the commodity, foreign exchange, or fixed income markets.

7 Appendix 1

Company name	NYSE	# Entries	Boolean search string
Bank of America	BAC	47615	”bank of america”
BB&T Bank	BBT	4125	”bb&t”
CIT Group	CIT	2965	”cit Group”
Citigroup	C	40557	”citigroup”
Comerica	CMA	2432	”comerica” NOT ”comerica park”
Fifth Third Bank	FITB	3364	”fifth Third”
First Republic Bank	FRC	442	”first republic bank”
Hudson City Bancorp	HCBK	469	”hudson city bancorp” OR ”hudson city savings bank”
Huntington Bancshares	HBAN	1544	”huntington bancshares” OR ”huntington bank”
J.P. Morgan	JPM	38740	”jpmorgan”
KeyCorp	KEY	1365	”keycorp”
M&T Bank	MTB	2592	”m&t”
New York Community Bank	NYCB	459	”new york community bancorp” OR ”new york community bank”
People’s United Financial	PBCT	498	”people’s united financial” OR ”peoples united financial”
PNC Financial Services Group	PNC	1739	”pnc banks” OR ”pnc financial”
Regions Financial Corporation	RF	1784	”regions banks” OR ”regions financial” OR ”regions financial”
SunTrust Banks	STI	1653	”suntrust banks” OR ”suntrust bank”
U.S. Bancorp	USB	7585	”u.s. bancorp” OR ”us bank” OR ”us bancorp”
Wells Fargo	WFC	24813	”wells fargo” NOT ”wells fargo center”

Table 7.1: *The companies used to create the portfolio, their corresponding NYSE stock symbol, the number of entries found when searching in the Aitellu database, and the search strings used for creating the search profiles.*

Bibliography

[1] Msci world index. `https://www.msci.com/world`. Accessed: 2015-04-23.

[2] Python stop-words. `https://pypi.python.org/pypi/stop-words/2014.5.26`. Accessed: 2015-04-06.

[3] Aase, K. G. Text mining of news articles for stock price predictions.

[4] Alanyali, M., Moat, H. S., and Preis, T. Quantifying the relationship between financial news and the stock market. *Scientific reports 3* (2013).

[5] Bingham, E., and Mannila, H. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* (2001), ACM, pp. 245–250.

[6] Bondt, W. F. M., and Thaler, R. Does the stock market overreact? *The Journal of finance 40*, 3 (1985), 793–805.

[7] Breiman, L. Random forests. *Machine learning 45*, 1 (2001), 5–32.

[8] Chakrabarti, S. *Mining the Web: Discovering knowledge from hypertext data.* Morgan Kaufmann, 2003.

[9] Davis, A. K., Piger, J. M., and Sedor, L. M. Beyond the numbers: An analysis of optimistic and pessimistic language in earnings press releases. *Federal Reserve Bank of St. Louis Working Paper Series*, 2006-005 (2006).

[10] Dietterich, T. G. Ensemble methods in machine learning. In *Multiple classifier systems.* Springer, 2000, pp. 1–15.

[11] Dunning, T. Accurate methods for the statistics of surprise and coincidence. *Computational linguistics 19*, 1 (1993), 61–74.

[12] Edwards, R. D., Magee, J., and Bassetti, W. H. C. *Technical analysis of stock trends.* CRC Press, 2007.

[13] Eyheramendy, S., and Madigan, D. A novel feature selection score for text categorization. In *Proceedings of International Workshop on Feature Selection for Data Mining: Interfacing Machine Learning and Statistics (in Conjunction with 2005 SIAM International Conference on Data Mining)* (2005), pp. 1–8.

[14] Falinouss, P. Stock trend prediction using news articles. *Master's thesis, Lulea University of Technology* (2007), 1653–0187.

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

[16] Fawcett, T., and Provost, F. Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining* (1999), ACM, pp. 53–62.

[17] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. From data mining to knowledge discovery in databases. *AI magazine 17*, 3 (1996), 37.

[18] Forman, G. An extensive empirical study of feature selection metrics for text classification. *The Journal of machine learning research 3* (2003), 1289–1305.

[19] Frawley, W. J., Piatetsky-Shapiro, G., and Matheus, C. J. Knowledge discovery in databases: An overview. *AI magazine 13*, 3 (1992), 57.

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

[21] Gidófalvi, G., and Elkan, C. Using news articles to predict stock price movements. *Department of Computer Science and Engineering, University of California, San Diego* (2001).

[22] Hafez, P. A. How news events impact market sentiment.

[23] Hafez, P. A., and Xie, J. Factoring sentiment risk into quant models. *Available at SSRN 2071142* (2012).

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

[25] Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J., and Tibshirani, R. *The elements of statistical learning*, vol. 2. Springer, 2009.

[26] Jolliffe, I. *Principal component analysis.* Wiley Online Library, 2002.

[27] Kaya, M. I. Y., and Karsligil, M. E. Stock price prediction using financial news articles. In *Information and Financial Engineering (ICIFE), 2010 2nd IEEE International Conference on* (2010), IEEE, pp. 478–482.

[28] Klein, F. C., and Prestbo, J. A. *News and the Market.* H. Regnery Co., 1974.

[29] Koppel, M., and Shtrimberg, I. Good news or bad news? let the market decide. In *Computing attitude and affect in text: Theory and applications.* Springer, 2006, pp. 297–301.

[30] Lavrenko, V., Schmill, M., Lawrie, D., Ogilvie, P., Jensen, D., and Allan, J. Language models for financial news recommendation. In *Proceedings of the ninth international conference on Information and knowledge management* (2000), ACM, pp. 389–396.

[31] Lee, C. C., and Lee, J. D. Stock prices and the efficient market hypothesis: Evidence from a panel stationary test with structural breaks. *Japan and the world economy 22*, 1 (2010), 49–58.

[32] Lee, L. W., and Chen, S. M. New methods for text categorization based on a new feature selection method and a new similarity measure between documents. In *Advances in Applied Artificial Intelligence.* Springer, 2006, pp. 1280–1289.

[33] Li, Y. H., and Jain, A. K. Classification of text documents. *The Computer Journal 41*, 8 (1998), 537–546.

[34] Liddy, E. D. Text mining. *Bulletin of the American Society for Information Science and Technology 27*, 1 (2000), 13–14.

[35] Liu, T., Liu, S., Chen, Z., and Ma, W. Y. An evaluation on feature selection for text clustering. In *ICML* (2003), vol. 3, pp. 488–495.

[36] LLC, M. Random forests.

[37] Lo, A. W. Reconciling efficient markets with behavioral finance: the adaptive markets hypothesis. *Journal of Investment Consulting 7*, 2 (2005), 21–44.

[38] Malkiel, B. G. *A random walk down Wall Street: including a life-cycle guide to personal investing.* WW Norton & Company, 1999.

[39] Malkiel, B. G. The efficient market hypothesis and its critics. *Journal of economic perspectives* (2003), 59–82.

[40] Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to information retrieval*, vol. 1. Cambridge university press Cambridge, 2008.

[41] Martínez, A. M., and Kak, A. C. Pca versus lda. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 23*, 2 (2001), 228–233.

[42] McCallum, A. K. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering, 1996.

[43] Miller, G. A. Wordnet: a lexical database for english. *Communications of the ACM 38*, 11 (1995), 39–41.

[44] Mittermayer, M. A., and Knolmayer, G. F. Newscats: A news categorization and trading system. In *Data Mining, 2006. ICDM'06. Sixth International Conference on* (2006), Ieee, pp. 1002–1007.

[45] Moschitti, A., and Basili, R. Complex linguistic features for text classification: A comprehensive study. In *Advances in Information Retrieval*. Springer, 2004, pp. 181–196.

[46] Nassirtoussi, A. K., Aghabozorgi, S., Wah, T. Y., and Ngo, D. C. L. Text mining for market prediction: A systematic review. *Expert Systems with Applications 41*, 16 (2014), 7653–7670.

[47] Ng, H. T., Goh, W. B., and Low, K. L. Feature selection, perceptron learning, and a usability case study for text categorization. In *ACM SIGIR Forum* (1997), vol. 31, ACM, pp. 67–73.

[48] Niederhoffer, V. The analysis of world events and stock prices. *Journal of Business* (1971), 193–219.

[49] Novovicova, J., and Malik, A. Information-theoretic feature selection algorithms for text classification. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on* (2005), vol. 5, IEEE, pp. 3272–3277.

[50] Pang, B., Lee, L., and Vaithyanathan, S. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10* (2002), Association for Computational Linguistics, pp. 79–86.

[51] Peterson, L. E. K-nearest neighbor. *Scholarpedia 4*, 2 (2009), 1883.

[52] Porter, M. F. An algorithm for suffix stripping. *Program 14*, 3 (1980), 130–137.

[53] Prabowo, R., and Thelwall, M. A comparison of feature selection methods for an evolving rss feed corpus. *Information processing & management 42*, 6 (2006), 1491–1512.

[54] Quinlan, J. R. *C4. 5: programs for machine learning*. Elsevier, 2014.

[55] Rachlin, G., Last, M., Alberg, D., and Kandel, A. Admiral: A data mining based financial trading system. In *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on* (2007), IEEE, pp. 720–725.

[56] Schumaker, R. P., and Chen, H. Textual analysis of stock market prediction using financial news articles. *AMCIS 2006 Proceedings* (2006), 185.

[57] Schumaker, R. P., and Chen, H. A quantitative stock prediction system based on financial news. *Information Processing & Management 45*, 5 (2009), 571–583.

[58] Schuster, T. News events and price movements. price effects of economic and non-economic publications in the news media. *Economic Publications in the News Media (May 2003). Leipzig University Working Paper*, 03-02 (2003).

[59] Sebastiani, F. A tutorial on automated text categorisation. In *Proceedings of ASAI-99, 1st Argentinian Symposium on Artificial Intelligence* (1999), Buenos Aires, AR, pp. 7–35.

[60] Seo, Y. W., Ankolekar, A., and Sycara, K. Feature selection for extracting semantically rich words. Tech. rep., DTIC Document, 2004.

[61] Sewell, M. History of the efficient market hypothesis. *RN 11*, 04 (2011), 04.

[62] Tang, X., Yang, C., and Zhou, J. Stock price forecasting by combining news mining and time series analysis. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on* (2009), vol. 1, IET, pp. 279–282.

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

[64] Thomas, J. D., and Sycara, K. Integrating genetic algorithms and text learning for financial prediction. *Data Mining with Evolutionary Algorithms* (2000), 72–75.

[65] Thomsett, M. C. *Getting started in fundamental analysis.* John Wiley & Sons, 2006.

[66] Urquhart, A., and Hudson, R. Efficient or adaptive markets? evidence from major stock markets using very long run historic data. *International Review of Financial Analysis 28* (2013), 130–142.

[67] Vu, T. T., Chang, S., Ha, Q. T., and Collier, N. An experiment in integrating sentiment features for tech stock prediction in twitter.

[68] Wang, Q., Guan, Y., Wang, X., and Xu, Z. A novel feature selection method based on category information analysis for class prejudging in text classification. *International Journal of Computer Science and Network Security 6*, 1A (2006), 113–119.

[69] Wang, Y., and Wang, X. J. A new approach to feature selection in text classification. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on* (2005), vol. 6, IEEE, pp. 3814–3819.

[70] Weiss, S. M., Indurkhya, N., and Zhang, T. *Fundamentals of predictive text mining.* Springer Science & Business Media, 2010.

[71] Wuthrich, B., Cho, V., Leung, S., Permunetilleke, D., Sankaran, K., and Zhang, J. Daily stock market forecast from textual web data. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on* (1998), vol. 3, IEEE, pp. 2720–2725.

[72] Yan, J., Liu, N., Zhang, B., Yan, S., Chen, Z., Cheng, Q., Fan, W., and Ma, W. Y. Ocfs: optimal orthogonal centroid feature selection for text categorization. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval* (2005), ACM, pp. 122–129.

[73] Yang, Y., and Pedersen, J. O. A comparative study on feature selection in text categorization. In *ICML* (1997), vol. 97, pp. 412–420.

[74] Yilmazel, O., Finneran, C. M., and Liddy, E. D. Metaextract: an nlp system to automatically assign metadata. In *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries* (2004), ACM, pp. 241–242.