StockWatcher 2.0: Using Text Analysis to Predict Stock Market Trends

Alex Brojba-Micu
289585ab@student.eur.nl

Master Thesis

Department of Economics and Informatics
Erasmus School of Economics
Erasmus University Rotterdam

Supervisor: Dr. Flavius Frasincar
Co-supervisor: Dr. Viorel Milea

March 12, 2013

Acknowledgements

I am grateful for all the support I have received whilst researching and writing up this thesis. Of course I could not have done this work all alone, therefore I will gratefully thank all the people helping me.

First my former teacher and now thesis supervisor, Flavius Frasincar, for the valuable discussions and input. Furthermore, his revisions and corrections were crucial in creating a proper master thesis. And of course Viorel Milea, my thesis co-supervisor, who asked crucial questions which resulted in a more extensive and better theoretical background for my thesis. I am thankful to both for their long patience and support.

But one should never forget the people who had to deal with me during those months of work and exhaustion, especially my girlfriend Oana, which supported me every step of the way, and indulged in my mental absence.

Abstract

In this thesis we present a new application, StockWatcher 2.0, which predicts the stock price for listed companies by analyzing events in news items, with the assistance of Natural Language Processing. This is accomplished by applying a series of algorithms on news articles, in order to identify relevant information. The thesis first discusses several similar projects, by explaining their approach and results. We continue with a theoretical background on stock prediction, and a summary of Natural Language Processing techniques which are required for our application. Then we select the best performing techniques, and elaborate the development of our application. StockWatcher has been tested with an unique data set, producing very encouraging results. The application is able to predict stock prices for listed companies by analyzing news items regarding the company. The results are two-fold: BUY/SELL signal precision of 53.3% and actual investment excess returns (compared to the NASDAQ index return for the same time period) of 6.42% for the period of an year.

Keywords: natural language processing, word sense disambiguation, stock price prediction

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Being able to predict the evolution of financial markets is the key element for brokers and private individuals to earn money. This prediction process has been however very challenging, as the financial markets have a complex behavior. There are several distinguishable methods to approach this problem, mostly based on the analysis of structured numerical information. The most employed analysis and prediction method is based on statistical tools, where the focus is on previous price evolution. A second approach however, based on text information, is earning a higher degree of popularity the past years. The underlying concept behind it is that the prediction of the stock price for certain companies is possible by analyzing news items concerning that company. According to Ng and Fu [2003], stock prices are influenced by a contribution of key factors from several sources: from news items directly released by the companies to news items concerning the global economy and its trends.

The Web is the only medium where news forecasting is not restrained by time and place. Unlike printed media or television programs, on the Web news can be publicized as fast as an item becomes news and does not have to wait until it is being printed or broadcasted. At the same time more and more people have access to the Web, therefore news on the Web become more

accessible. News websites provide RSS-feeds facilitating the public to remain up-to-date about any topic of interest. Meanwhile, the new technological developments facilitating fast desktop computers for individual users, as well as the evolution in the field of knowledge discovery makes it possible to data mine the news items for critical information in real-time.

This form of knowledge extraction, also referred to as data mining, intends to employ computers and specific software on large amounts of digital data in order to find interesting patterns and relationships within the data. With the growth of the Web an increasing amount of textual data is available. This evolution has led the process of knowledge discovery to be expanded to text mining, where the focus lies on unstructured text data, as shown by Hearst [1997]. With the use of text mining techniques, aided by natural language tools, it becomes possible to extract relevant information from natural language text documents, and to use this information for other computer-based applications. In the context of this thesis, by creating an interaction between the discovery of key elements in textual data it becomes possible to build prediction models for the stock price towards a focus company. News items concerning certain companies can have several effects on the stock price towards that company (positive, negative and neutral). The key is to correctly identify these news items and extract the relevant information. For example, we would like to predict the influence of the news article illustrated in Figure 1.1 on the concerned companies. On the day this news item was published online, the stock prices of Delta Air and Northwest risen by 3.50% and 2.00%, respectively.

1.1.1 Natural Language Processing

Natural Language Processing (NLP) is a field created for the understanding of the natural human language. With the use of NLP techniques developers are trying to teach machines what we understand so easy, our human languages, according to Chowdhury [2003]. By natural language we mean a usual type of language generally used by humans. That excludes man-made languages such as Java, which falls in the realm of programming languages.

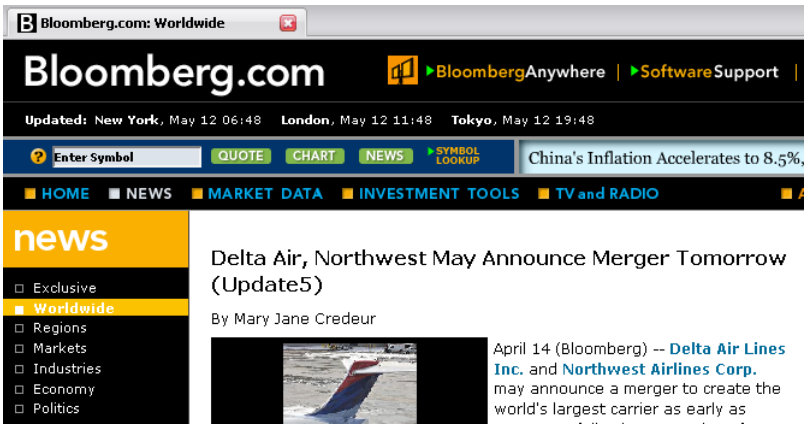

Figure 1.1: News article from Bloomberg.com

As a result, when we talk about NLP, we are addressing the efforts of using computers to process natural languages. NLP has three distinctive categories according to Jurafsky and Martin [2000], namely: speech, grammar and meaning. Our focus is only on grammar and meaning.

Most languages do have rules that need to be obeyed. These rules apply to the structure of the words (morphology), as well as to the structure of the sentences (syntax). In this way we have a generalized consensus on how to write a well-structured text, so that the reader is not confronted with trivial problems about the part-of-speech of a word or the sentence structure. This represents the grammar category. The second category of our interest is meaning. This category can be divided as well into semantics and pragmatics. While semantics is referring to the meaning of words and consequently the meaning of sentences, pragmatics is about what did the speaker or writer intended to express. Our focus in this case is to find the meaning behind the words.

1.2 Goals

We intend to employ the available NLP techniques, as discussed above, on financial market related news items representing companies from the NASDAQ-100 available on Internet RSS-feeds. The purposes of this process

it to first let our system identify key concepts which can influence the stock price concerning a given company. This would enable us to predict how future news items will impact the stock price. The main research question of the thesis is:

- Is it possible to predict stock prices for listed companies by analyzing events in news items?

The subquestions of our main research question are:

- What are the available methods for disambiguating words?
- How can we predict stock prices by making use of textual news articles?
- How reliable is a natural language processing technique involving event recognition in making stock price predictions?

1.2.1 NASDAQ

The NASDAQ stock exchange is one of the largest electronic financial market in the world. Over 3200 companies are listed and on average there are more shares traded on NASDAQ than any other market in the U.S. Furthermore, a stock market index was created for the 100 largest companies that are listed on NASDAQ, named NASDAQ-100[1]. Every year a rebalancing of the index takes place, where companies are delisted (merging, bankruptcy or declining market value of the company) with other companies taking the spot. Moreover, several large non-U.S. companies are also included in the NASDAQ-100.

1.3 Methodology

The first step in our research is a thorough literature study in this field. The focus is on existing projects with a degree of resemblance to ours. This

[1]NASDAQ - http://dynamic.nasdaq.com/dynamic/nasdaq100activity.stm

process will help us determine what has already been researched in regards to our own project and might even provide benchmarks for future validation of our software.

To accomplish our goal, the first step in our thesis is to design an application that is able to learn the importance of events. This is achieved by collecting news items regarding several companies on a time frame of several months. Each news item is rated by the difference between the opening and closing stock price for the subject company on the date of the news item appearence. Afterwards, we compare our own event concept list to the rated news items, and the application will learn what is the impact of the event on the stock price.

Certain requirements can already be specified for our application and research. First of all, a list has to be created with financial market relevant events. Extensive literature review and previous projects on this theme should provide insights for this problem. Furthermore, a reliable news service for the news items has to be identified. For this matter we look at the RSS-feeds of big financial websites, such as Google Finance[2], MSN Money[3] and Yahoo! Finance[4].

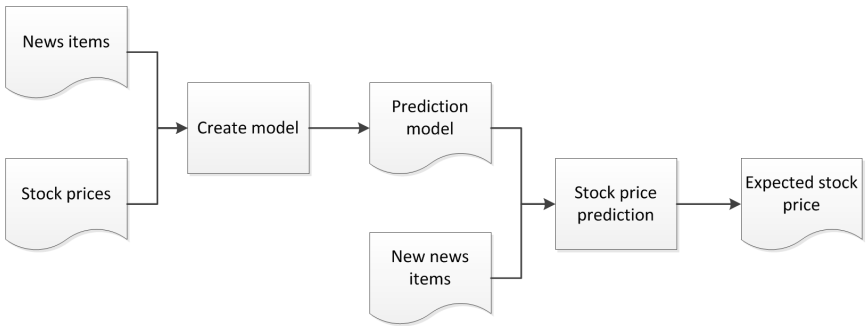

Figure 1.2: StockWatcher 2.0 framework

The second step of the research is to find if the system is able to predict the stock price given previously unseen news items. To validate this process, a second time frame is created. In this time frame a new batch of news

[2]Google - http://finance.google.com
[3]Microsoft - http://moneycentral.msn.com/home.asp
[4]Yahoo - http://finance.yahoo.com/

items will be collected. Our system will then process these news items and produce a stock price based on the identified events. This result will then be compared to to the actual stock price changes. In Figure 1.2 a conceptual model of our system is illustrated. The rectangles represent processes, which have documents or data as input / output. First a prediction model is build, as described above. Once this process is finished, this model is able to analyze news items and give a prediction concerning the stock price for the company.

1.4 Structure

In Chapter 2 of this thesis we discuss previous work work related to our field of interest. Relevant results and conclusions of other authors are pointed out for future reference. We also emphasize the limitations of similar approaches. In Chapter 3 we present the first part of our methodology. A technique and software survey for part of speech taggers and morphological analyzers is conducted. The survey enables us to select the necessary tools and techniques in order to create our application. Furthermore, in Chapter 4 we continue with the survey by comparing the available word sense algorithms.

Once the proper tools and techniques have been identified, in Chapter 5 we describe how they are used in our application. This chapter includes conceptual models and algorithms presented in pseudo-code. In Chapter 6 we test the developed application on our own news article corpus. A detailed analysis of the results, including precision, is presented. In Chapter 7 we present our conclusions and give recommendations for future work.

Chapter 2

Related Work

2.1 Introduction

In this chapter we investigate a number of tools and techniques which predict human sentiment towards companies and their stock price based on news items. We highlight the main characteristics and features of these applications, together with the obtained results as mentioned by the authors. We also give attention to the main differences between our goals and these applications.

2.2 Warren

Warren, as discussed by Seo et al. [2002], is an ensemble of intelligent agents, designed to assist humans in financial markets. The agents help users to track stock prices, give performance assistance, earning summaries and risks concerning companies of interest in the user's portfolio. One of the tasks performed by the agents is to analyze news items concerning related companies. The news analyzing agent classifies news items in one of the following five categories:

- **Good** News articles which show positive evidences of the company's

financial status explicitly, e.g., "the shares of ABC Company rose 2 percent on the NASDAQ to $24."

- **Good, uncertain** News articles which refer to predictions of future profitability, and forecasts, e.g., "ABC Company predicts fourth-quarter earnings will be high."
- **Neutral** News articles which did not explicitly mention anything about the financial well-being of the company, e.g., "ABC and XYZ Inc. announced plans to develop an industry initiative."
- **Bad, uncertain** News articles which refer to predictions of future losses, or no profitability, e.g., "ABC warned Tuesday that fourth-quarter results could fall short in expectations."
- **Bad** News articles which show negative evidences of the company's financial status explicitly, e.g., "the shares of ABC fell in early New York trading."

By making use of frequent collocated phrases (FCP), the agent is able to correctly categorize a news item. The FCP method searches for key words in the text, however it is not important how far these words are apart in the text (for example the words "shares" and "rose" in the same sentence would form a FCP). Once a FCP is recognized in an article, a Domain Expert is created, which can cast a vote on that article with a certain weight. Each Domain Expert is a term (word) with a weight. The vote (or votes) determine the classification of this article in the correct class (good, bad, etc.). If the agent would encounter the phrases "deal, good, gain", a vote would be cast with a high weight for the Good category. The weights are assigned to the Domain Experts by a naive Bayes classifier, with expectation maximization.

One of the conclusions of this research was the fact that they observed that nearly all the financial news articles were very ambiguous on the subject company. For example, it was hard for the algorithm to determine which company was the subject in the following phrase: "Company A share has gone up today. Competitors in the branch had a decline in share price.".

As we can see, the news item contains information about several companies, however, (if company A is the subject) the subject is mentioned in certain sentences. Therefore, the frequent collocated phrase method was restricted to only sentences where specific company names appeared.

The results obtained by the application were encouraging, scoring accuracies between 35% with previously unclassified articles, and 56% with previously classified articles. However, the number of FCP that the application can identify is very small (15 in total), and the focus is on machine learning techniques, with no consideration to NLP techniques.

2.3 OASYS

OASYS describes itself as an opinion analysis system, where text documents in a certain topic are scanned, in order to establish a general sentiment of those documents on that topic, according to Cesarano et al. [2006]. The results generated by the application can be either qualitative or quantitative. It is critical however that a scope is always provided with the documents, as the scores are determined by the documents in conjunction with the topic. The architecture of the application consists from five distinguishable components:

1. User specification: This allows the user to input the source of the documents, which can be URLs, directories and domain names. Furthermore, the topic has to be specified, and an interval time as well.

2. Web spider: This component has the task to retrieve the documents from the input sources. It is also in charge of the filtering, as specified by the user, of time intervals and topics. The algorithm facilitating the topic filtering is further described in Deerwester et al. [1990].

3. Scored opinion expressing word bank: This component represents a database with words that are more or less considered direct or indirect opinions. Scores have been assigned to these words, expressing the degree of positivism or negativism of the word. There are two methods

to score the opinions: an extensible library of opinion expressing word scoring methods and a restricted word bank with only specific types of words (e.g. nouns).

4. Quantitative opinion analysis algorithms: The system employs several algorithms, which analyze each document individually, and evaluate the opinion of that document with regard to the topic. By employing the scored opinion expressing word bank, the application is able to provide a quantitative score. Furthermore, the application allows for additional score information gained from other applications / algorithms. By making use of a hybrid algorithm, the system is able to merge the results gained with outside results.

5. Qualitative scoring module: This component allows the application to produce qualitative scores for the documents. This is accomplished by assigning a variety of adjectives (e.g. positive, neutral, negative) to the various ranges of the qualitative scores. The ranges for the adjectives are learned automatically by the system. Furthermore, the application can present users with a time evolution of the scores, for instance how the opinion towards the Iraq war changed during the past 3 years.

The scored opinion expressing word bank was created by allowing a panel of 16 individuals to rate 100 documents with a score from 0 to 1. The higher the score (towards 1), the harsher the document is on the specific topic. The lower the score (towards 0), the more positive the document is. Furthermore, the rated documents were analyzed to extract the words and build an opinion score, based on the relative proportion of a word and its synonyms in the document, and the human score assigned to that document.

Once the scored opinion expressing word bank has been created, there are several algorithms the application can be applied to retrieve a quantitative opinion analysis:

- **Topic-focused algorithm** - this method searches for all the sentences where a direct or indirect opinion is expressed about the topic. Per

sentence a score is calculated by summing all the word matches in the sentence with the word bank. The final score is then calculated by taking an average score over all sentences.

- **Distance-weighted topic focused algorithm** - this method applies the above discussed topic-focused algorithm to the document, by calculating an average score for every sentence for each word match with the word bank. The document is then divided in two parts, with one part representing sentences with a direct or indirect opinion (group 1) and the second part representing no opinion regarding the topic (group 2). By calculating the distance between sentences in group 1 and 2, the algorithm can determine the impact of sentences in group 2 on sentences in group 1. In other words, expression matches found in sentences in group 2 which are near sentences from group 1 have more weight on the final score compared to expression matches found in sentences in group 2 which are further away from sentences in group 1.

- **Template-based algorithm** - this algorithm employs a set of sentence templates, and only assigns scores to sentences matching the template.

- **Hybrid evaluation method** - this method combines the results generated by the above mentioned algorithms (however it is not restricted to them), to calculate an average score.

QualScore is the algorithm employed by the application to assign qualitative scores to documents, reflecting the opinion on certain topics. A rating scale was created, with a hierarchical order of adjectives (e.g. positive, harsh and very harsh). Threshold values are then assigned to the adjectives, indicating to what degree a document can be coupled with an adjective. For instance, a quantitative score between 0 and 0.4 can represent positive opinion, 0.4 and 0.7 harsh opinion, etc. The ranges are automatically computed by the system. The document receives a normal quantitative score, after which the nearest (closest qua quantitative score) user analyzed document is found.

The qualitative score assigned to that document by the user is then also used for the computer analyzed document.

The results obtained by the application converge around 50%, with a computation time of 1 second per weblog post, by applying the hybrid evaluation method. It is important to point out the fact that this technique makes little to no use of natural language processing techniques, except synonyms for the adjective keywords.

2.4 AAC-based sentiment analysis

Benamara et al. [2007] proposes the use of an adverb-adjective combination (AAC) based system to denote the opinion of a certain article or phrase on a topic. The score produced by the system can range from –1 (fully negative opinion) to 1 (fully positive opinion). Most such applications, that try to extract the opinion from a digital text, concentrate only on adjectives. It is the opinions of the authors however, that by taking into account the influence of adverbs on adjectives, the systems can improve significantly. To begin with, a list of adverbs is created, and these adverbs are then categorized as followed by Lobeck [2000] and Quirk et al. [1985]:

- Adverbs of affirmation: certainly, exactly, totally, etc...
- Adverbs of doubt: roughly, apparently, seemingly, etc...
- Strong intensifying adverbs: astronomically, exceedingly, extremely, immensely, etc...
- Weak intensifying adverbs: scarcely, weakly, slightly, etc...
- Negation and Minimizers: hardly, barely, etc...

Every adverb receives a score of 0 or 1. In this case, an adverb with score 0 means that this adverb has no influence on the adjective, and 1 the opposite. The adjective list employed by the system, is the one from the OASYS application, previously discussed.

The system can employ three scoring algorithms, which all focus on different aspects of the word matches. First there is the *variable scoring algorithm*, where the score is calculated by simply adding the rating for the adjective with a multiplied rating of the adverb. Furthermore we have the *adjective priority scoring*, where a weight is introduced, which signifies the importance of an adverb compared to an adjective that it modifies, the larger the weight, the higher the impact. The last algorithm, *adverb first scoring*, is similar to the previous one, however the weight is now applied to the adjective, instead of the adverb. The results (precision and recall) obtained by this method were slightly higher compared to the results obtained by OASYS.

2.5 TOWL

The Time-determined Ontology Web Language (TOWL)[1] is a project with the focus on news content studies towards institutional equities services, investors and businesses, funded by the European Union. The main idea behind TOWL is, as the name already suggests it, adding a time constraint to the existing semantic web ontologies: OWL. Having a time property in an ontology will give the opportunity to migrate from the current static representations of the world, and move into a more dynamic content environment. Equipped with this new technology, a machine-based approach in knowledge extraction from business, market and global news, should provide the edge in making improved business decisions, resulting in increased performance.

To present the possibilities that TOWL has to offer, a semantic stock broker system is developed. Furthermore, this system acts as a benchmark for the project. By employing news messages from Reuters, the system can define and improve its ontology, to have a better representation of the real world. With every new article, the system tries to deduce how a certain stock price might fluctuate. The goal of this process is to generate accurate results in order to increase profitability of investments on a given portfolio. This semantic tool employs many NLP techniques, and its methods and goals

[1] TOWL - http://www.towl.org

resemble ours to a high degree: analyze news items to predict financial market trends.

The stock broker application developed for TOWL has three types of data:

- as first news items are the only input source for the prediction of the stock price;
- secondly, a rich ontology, focused on the stock domain has been created, where a lot of information regarding companies is stored;
- the results generated by the application, in the form of the development of a specific stock price.

At the moment a prototype of the stock broker application is available, which uses some plugins from GATE, as explained in Cunningham et al. [2002]:

1. **ANNIE Tokeniser and TOWL Sentence Splitter**

   These plugins are responsible for the breakdown of the text in words, or sentences.

2. **ANNIE POS Tagger**

   This plugin is responsible for identifying nouns, verbs, adjectives, adverbs, etc., in the text as corresponding to a particular part-of-speech.

3. **WordNet Lemmatiser**

   This plugin provides an interface to search the lemma of the words in the text (for example, for "running" the lemma is "run").

4. **Gazetteer**

   This is a tool designed to analyze a text, given an ontology, and finding matches between the ontology and the text.

5. **OWL Ontology Instantiator**

   This tool makes sure that the ontology is being updated with newly discovered instances.

The stock broker in development for TOWL has a lot of potential to become an effective application. By making use of different NLP techniques in combination with the tools available from the Semantic Web, the application is able to output fully annotated news articles. On another side note, the extensive use of NLP techniques can provide a foundation for our own project.

2.6 Other related work

In Cho [1999] the focus is on a system which employs probabilistic rules to predict financial market price trends. As framework, the authors make use of 400 hundred big word list, created by financial market experts. With this list, he then attempts to assign weights to the words, by analyzing news items about certain companies, and comparing the news items with the closing price of that stock. From these values, the probabilistic rules are generated. These rules are then used to predict where the stock price for a certain company is heading: up, down or stay the same. In Peramunetilleke and Wong [2002] the same probabilistic rule principle is applied, however the news analysis is constrained to the news titles. An accuracy of 51% was gained with this method in predicting the influence of news headlines on intraday currency exchange rate movements, which was a 6% increase compared to the previous method.

In Lavrenko et al. [2000] a Bayesian prediction model called Analyst is used to predict stock price trends. This application compares already described trends to news items and stock prices, in order to find out which news items coupled with stock prices lead to which trend. Once the learning process is done, the application is able to present users with relevant news items, which may be the starting point of a price trend. Furthermore, in Fawcett and Provost [1999] an approach is presented for predicting the influence of news articles on the stock prices by closely monitoring the stream of data (for example news items on RSS-feeds) for key words that may signal a positive trend. In Ahmad et al. [2002], the authors tried to link certain keywords (for example “rise”, “strong”) with the price trends on FTSE 100.

	NLP		Events
Application	Synonyms	WSD	Event restriction
Warren	no	no	no
OASYS	yes	no	yes
ACC	yes	no	yes
TOWL	yes	no	no

Table 2.1: Differences between existing prediction applications

Their conclusion was that the positive gradient in "good" news correlated well with the positive gradient in the FTSE 100 index value.

2.7 Discussion

As already mentioned in the introduction of this chapter, it is important to highlight the abilities of the above described applications, and at the same time compare it to our application needs. In Table 2.1 we try to illustrate the basic characteristics and differences of the applications. While Warren makes use of machine learning techniques, the other three systems focus more on the news articles by employing various NLP techniques. It is also important to mention the fact that both OASYS & ACC-based system have a restricted event list. Both systems are restricted to part of speech tags such as adjectives and adjectives & adverbs respectively. This event list could be expanded with a number of verbs for example. Last, the use of NLP tools is only restricted to synonyms in some cases, and none of the discussed applications employ word sense disambiguation (WSD).

2.8 Summary

Using specific learning techniques based on text classification, Warren attempts to identify a number of key events in news articles. However, the number of key events is very restricted, and no NLP technique is employed at all. The accuracy of the system ranges from 35% to 56%, based on the

news article input. Other machine learning techniques, where the authors attempt to find relationships between stock prices and news items, involve the use of probabilistic rules and Bayesian prediction models. The accuracies of such systems range between 45% and 51%. Again the use of NLP is very limited.

OASYS and the AAC-based systems approach the prediction problem from a different view, by placing the focus on the news articles themselves, instead of the machine learning techniques. Furthermore, NLP plays a (small) role, as the applications make use of synonyms. The problem however is the event identification restriction, as they only search for adjectives (in the AAC-based system they couple adjectives with adverbs). There are a great number of verbs which represent important events that may influence the opinion on stock prices. In both cases the accuracy converges around 50%.

The semantic stock broker created for TOWL resembles our project to a high degree, by making use of the latest NLP techniques in order to predict stock prices. In the following chapter we explore this, by discussing some of the tools already employed by the Semantic TOWL stock broker.

As we can see from the discussed applications and research in this chapter, there is a clear shift in the prediction systems from machine learning techniques to natural language processing based techniques. This change can be explained by the new developments in NLP techniques. Nevertheless, we observe that the use of NLP techniques can be expanded, by introducing word sense disambiguation in order to solve some of the ambiguity problems and increase accuracy. Furthermore, the number of key events can also be increased, however this has to happen without a loss in accuracy.

Chapter 3

Natural language processing

3.1 Introduction

In this chapter we present a more thorough analysis of the decisions made for our application combined with the proper literature to support them. This process helps us determine the requirements necessary for our application. Furthermore, a natural language processing technique and software survey is presented, where the advantages and disadvantages of each technique are discussed. With the survey, it is our intention to identify the right "building blocks" for our application.

3.2 Stock Markets

A stock market is essentially an entity that facilitates the trading of company stock, securities and derivatives. Stock exchanges are examples of such entities. The stock exchanges are specialized in bringing buyers and sellers of stocks and securities together. New York Stock Exchange (NYSE)[1], NASDAQ[2] and American Stock Exchange (AMEX)[3] are some of the biggest

[1]New York Stock Exchange - http://www.nyse.com

[2]National Association of Securities Dealers Automated Quotations - http://www.nasdaq.com

[3]American Stock Exchange - http://www.amex.com

stock exchanges in the world. Stock markets represent one of the most important sources for companies world wide to raise financial investments. At the same time it presents individuals with the opportunity to increase their economic wealth, by investing in companies and buying stocks.

With the large sums of money involved in stock market transactions, many have tried to identify patterns in stock prices, in order to predict their movement. This process has gone as far as analyzing the influence of weather on stock returns, as done by Hirshleifer and Shumway [2004]. They concluded that sunshine is strongly correlated with increased stock returns. This trend continued, and the answer was searched in the statistical field: from evolution strategy based on genetic algorithms Korczak et al. [2002] to adaptive belief systems where agents try to evolve in an Agent Based Model proposal by Hommes [2002]. In the literature there are several theories that attempt to describe the behavior of a stock market, in order to be able to predict it. The two most important are the efficient market hypothesis (EMH) developed by Fama [1965] and the random walk theory, created by Malkiel [1990].

3.2.1 Stock Market prediction theories

The efficient market hypothesis (EMH) introduced by Eugene Fama, argues that news and information are already incorporated in the price of a given stock.

> *Definition: A financial market is (informationally) efficient when market prices reflect all available information about value.*

From this argument we can deduce that an individual can not make any profits by using new information or news. So stock price prediction can not be based on information. Furthermore, the EMH theory can be shaped in three forms according to Brealey and Myers [2000]: weak, semi-strong, and strong efficiency.

- **Weak form efficiency**

In the current stock price only historical price information is incorporated, making it impossible to create excess earnings based only on past price analysis (technical analysis).

- **Semi-Strong efficiency**

  The current stock price is built from publicly available information and past prices, so there will be no excess earnings based on this information (fundamental analysis and technical analysis).

- **Strong form efficiency**

  The stock price now reflects all available information, public and private, so excess earnings are not possible.

The semi-strong form efficiency implies that a fundamental analysis of new publicly available information is incorporated in the stock price. However, technically a time interval must exist between the introduction of new information (fresh news article regarding a company) and the balancing of the stock price. Identifying and acting in this interval may be the key to generate increased earnings.

The random walk hypothesis argues that the stock price evolves according to a random walk. The underlying thought is that the current stock price is independent from previous price information, so creating excess earnings[4] is impossible. This argument is explained by the existing efficiency on a market. If individual investors are able to recognize patterns in stock prices, they would immediately take advantage of this. Once everyone is doing this, the recognized patterns will become worthless according to Brealey and Myers [2000]. However, this theory holds no argument against the influence of new information on stock prices. For example new articles with company information that was unknown beforehand can be used as a prediction tool.

3.2.2 Stock price relation to news articles

From the above discussed theories, we can conclude two hypotheses:

[4]Earnings over and above the expected rate of return.

- *certain news articles influence the stock price of a given company*;
- *the time interval to take advantage of the fundamental analysis of news articles is very short.*

In Chan et al. [2001] the authors confirm a relation between news items and stock price. Their second conclusion was that the time interval to take advantage of the knowledge is very small. In their research they discovered that salient news articles always have a positive or negative consequence for the stock prices and the volumes of traded stocks. Furthermore, the time interval to take advantage of this knowledge is very short, as the prices stabilize very fast. The same discoveries were made by Klibanoff et al. [1998] and Yue-Cheong [1996], where relationships were found between salient news articles from The New York Times and South China Morning Post respectively, and the trading volumes and prices. This is also confirmed by Mitchell and Mulherin [1994], where a strong and positive relationship was found between news headlines from Dow Jones and absolute price changes.

According to Patell and Wolfson [1984], a visible price change is observable after a news article release by Dow Jones. Immediate action must be taken once a news article was released, in order to be ahead of other investors and make profit.

3.3 Natural Language Processing

As already discussed in the first chapter, NLP can be defined with three categories: speech, grammar and meaning. The focus of this thesis is to determine the success of an application that predicts human sentiment, by making use of grammar and meaning in the text analyzation process.

According to Christian and Tsoukermann [1999], the grammar component can be divided in three distinguishable categories: lexicon, syntax, and morphology. Furthermore, the authors determined that each category relies on its predecessor. The morphology process can not be complete, without syntax, which can not be complete without lexical analysis. Lexical analysis

refers to tokenization, which is the process of converting data sequences in token sequences. Most used tokenization techniques are splitting big text articles into individual paragraphs, sentences or words. Syntax analysis refers to the process of finding out the part-of-speech tag of a given word in a sentence. This is necessary for the next process, the morphological analysis. The morphological analysis attempts to extract the lemma of a word, given its part of speech tag. Once the lemma has been identified, word sense disambiguation can be applied in order to find the meaning of the lemma in a sentence. In Figure 3.1 we depict the above described process. The input of the system is a list $D = (d_1,...,d_n)$, with d_i representing articles. The output of the system consists from a list $W = (w_1^{k_1},...,w_n^{k_n})$, where w_i represent the identified key events, and k_i is their meaning.

In the following section of this chapter we discuss several software solutions and algorithms, in relation to part-of-speech tagging and morphological analysis. As tokenization is a trivial process in our project, it is out of the scope of the research.

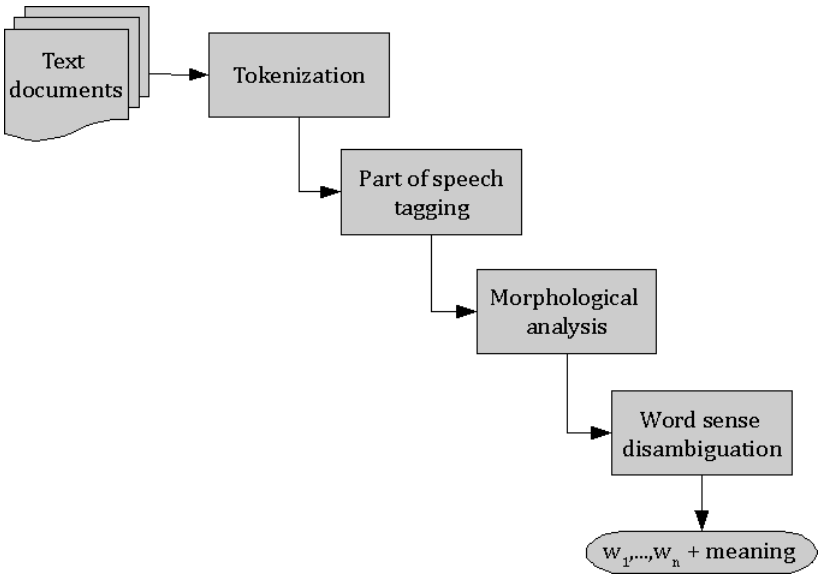

Figure 3.1: The intended natural language process for news articles

3.3.1 Part-of-speech taggers

Over the years part-of-speech (POS) tagging has been at the base of different NLP disciplines. POS tagging, also known as grammatical tagging, is the procedure of assigning the correct part-of-speech to the words in a text, based both on definition and context. In other words, POS tagging involves the right identification of words as nouns, verbs, adjectives, etc.

For a long period of time, POS tagging was considered an inseparable part of NLP. There are cases where assigning the correct part-of-speech depended on the semantics of the text. The algorithms designed determined the part-of-speech of words based on other NLP fields: syntax, morphology and semantics. However this notion was proven wrong by several algorithms such as the hidden Markov models according to Ghahramani [2002] and Lee et al. [1990] or the Viterbi algorithm in Viterbi [1967]. These findings convinced most people working in the field of POS tagging that it did not require being part of an entire NLP process, but can easily be separated from the other levels of the system.

In creating a relevant algorithm for POS tagging, there are two methods from which one can choose. The first one is called 'supervised learning', and it refers to the corpus on which the algorithm is trained on. In supervised learning the corpus requires to be tagged by humans. There are several such data sets available, with most popular ones being the Brown Corpus, which is further explained in Schubert and Tong [2003] and the Penn Treebank, Marcus et al. [1994]. The Brown corpus originates from 1961 and distinguishes as many as 87 different tags. The Penn Treebank is another corpus that consists from over 4.5 million words in American English and originates from 1989. In the Penn Treebank corpus we can find 36 different POS tags and 12 other tags, denoting punctuation and currency symbols.

The second method is labeled 'unsupervised learning'. The main difference with the previous described method consists from the fact that the corpus is not humanly annotated. Such unsupervised learning techniques make use of an untagged corpus to train the data, and try to create POS tags by induction. These algorithms are not making use of POS tags such as verbs,

nouns, etc. Instead they try to observe patterns in the text, and create clusters for commonly detected patterns. Such techniques however require many iterations to categorize everything correctly. Nevertheless, the clusters of words created after many iterations are comparable to the same groups of POS tags being used in supervised learning (e.g. verbs, nouns, etc).

TnT Tagger

The Trigrams'n'Tags tagger as described in Brants [2000], developed by Thorsten Brants, makes use of a hidden Markov model (HMM). A HMM is characterized by a stochastic finite state method, where the transitions between different situations as well as the output of the model from the situations is associated with a degree of chance. In other words, it is a statistical model in which observable parameters can be used to find hidden parameters. The author of the TnT Tagger strived to apply the simplest form of a HMM algorithm, with the basic idea being: "the simplest is the best".

As we can observe in equation 3.1, the TnT Tagger attempts to calculate the output probability (in this case the tag for a given word) based on a sequence of given words $w_1...w_T$ of length T. Moreover, $t_1...t_T$ are the potential tags for the previously mentioned words, and t_{-1}, t_0 and t_T are beginning of sequence and end of sequence markers to improve the accuracy. Furthermore, the algorithm takes into account punctuations, assigning the appropriate tags if sentence boundaries are not marked, and capitalization, by changing the probability distributions for such words.

$$argmax_{t_1...t_T} \left[\prod_{i=1}^{T} P\left(t_i|t_{i-1}, t_{i-2}\right) P\left(w_i|t_i\right) \right] P\left(t_{T+1}|t_T\right) \tag{3.1}$$

The TnT Tagger was trained and tested on the Penn Treebank, making it a supervised learning algorithm. The initial testing of the algorithm reported an accuracy of 96.7%. Training and tests on other data by Mieskes and Strube [2006] sets reported a level of accuracy between 96.6% and 96.7%.

Stanford Bidirectional POS Tagger

The Stanford bidirectional POS tagger as described Toutanova et al. [2003] makes use of the same hidden Markov model (HMM) as the TnT tagger. However, where the previous algorithm looked at previous tags to calculate a probability for the new tag, this algorithm looks in both directions. The following formula gives the underlying thought behind the tagger.

$$P(w|t) = \prod_{i=0}^{T} P(t_i|t_{i-1}, t_{i+1}) \tag{3.2}$$

To calculate the POS tag t for word w, the algorithm looks first at the tag it assigned to the previous word t_{i-1} and to the next word t_{i+1}.

The bidirectional tagger was trained and tested on the Penn Treebank and at the same time on the same data sets as the TnT Tagger, in Mieskes and Strube [2006]. For the Penn Treebank, the reported accuracy was between 96.5% and 97.15%. For the external data sets the accuracy was at the same levels, between 96.7% and 97%.

Brill Tagger

The Brill tagger, as described in Brill [1993], is named after its creator, Eric Brill. The past few decades the academic field of part-of-speech tagging became more and more used with the idea that a stochastic approach the most efficient and accurate is. However, the Brill tagger proved that different approaches, where no probabilities are involved, also work as good, and maybe even better. Stochastic taggers have no direct linguistic information, only statistical data about the information. The Brill tagger however, works on a rule-base that has direct linguistic information stored in a non-stochastic way. While the previous two discussed algorithms make use of a hidden Markov model, the Brill tagger makes use of a transformation-based error-driven learning paradigm.

Transformation-based error-driven learning is designed to allocate a POS tag to each word in stage 1, and possibly improve the POS tag using a set of

predefined rules in the second stage of the process (possibly improve because not each rule leads to an improvement). The first stage of the tagger assigns tags to words based on the examination of a large corpus (specifically it picks the tag with the highest count as found for that word in the corpus). If a word is not known, it automatically receives the tag "noun". In the second stage of the process, the tagger applies a predefined set of rules which can change the initial tag. An example of such a rule is: if the word ends with "ed", change the tag to a verb. This rule is first tested on the corpus, and if the error level is dropping (compared to the error level before the rule was applied), the rule can be applied to your own text. The author reports that he has identified 66 rules which produce a reduction of error.

The Brill tagger was as well trained and tested on the Penn Treebank, having an accuracy between 96.7% and 97.2%. For different external data sets, the accuracy was between 96.1% and 96.5%.

3.3.2 Morphological analysis

Morphology analysis, also known as lemmatization, is a natural language processing field that is concerned with the structure of words. The principle idea behind it is that words are related to each other by rules. The use of morphological analysis for the NLP domain consists from reducing words to their base canonical form, also known as the lemma. For example a morphological analyzer should identify as the canonical form for the words *'running'*, *'ran'* and *'runner'* the word *'run'*. A morphological analyzer requires the part-of-speech tag for the word, in order to determine the proper lemma.

Stemming

Stemming, as described in Frakes [1992], is the procedure of reducing words to their stem, also known as the root of the word. Differently from morphology analysis, stemming does not require contextual information of the words, like the part-of-speech. The first research in the field of automated

root words extraction dates from 1968, when Martin Porter published the first stemming algorithm in van Rijsbergen et al. [1980]. The algorithm became very popular, and soon many implementations and extensions were added to improve it. This algorithm was a suffix stripping algorithm, which makes use of a rule list to strip down words to their stem:

- if ends with SSES ⇒ change to SS

  caresses ⇒ caress

- if ends with S ⇒ delete s

  cats ⇒ cat

- if ends with ATIONAL ⇒ change to ATE

  relational ⇒ relate

Such algorithms are very simple to create and very fast to run. However, the performance is very poor given words as *'sung'* and *'sing'*.

Another well-known stemming approach is the brute force algorithm, described in Xu and Croft [1998]. The brute force algorithm compares words to a database of words, trying to match characters. The algorithm stops at the first occurrence of the pattern. This approach is not very efficient and requires a lot of database space. Moreover in its original state, the algorithm is very resource consuming. This was later fixed by using hashing methods to speed up the process, by implementing the Rabin-Karp method, in [Cormen et al., 2001]. Instead of searching for patterns in the database, hash values would be calculated and compared.

Last, there are stochastic algorithms available for determining the stem of a word. Using probabilities, these algorithms can be trained on tables with root words and 'normal' words to develop a probabilistic model.

Lemmatization is closely related to stemming and the above mentioned approaches. As previously mentioned, a stemmer processes a word without information of the context. In practice this means that stemming algorithms are easier to implement and run faster but at the same time, with a reduced

accuracy than lemmatisers. Morphological analyzers are very accurate but take a while longer for processing. A simple example where some stemming algorithm would fail to identify the root word would be the string *'better'*, with the root of *'good'*. On another note, some lemmatization techniques make use of stochastic algorithms. The difference with the above mentioned stochastic algorithms would consist from an improved accuracy given the part-of-speech tag (more data is always better in probabilistic approaches).

WordNet

WordNet, as described in Fellbaum [1998], the lexical database of English developed by the University of Princeton, has a morphological analyzer called Morphy. According to Miller et al. [1990], Morphy is an add-on for the dictionary that enables users to query WordNet with derivational forms of words of interest. Morphy can process derivational word forms, reducing the words to their lemma. This feature provides a certain user friendliness (users are not required to know the lemma for each word they want to look up) and at the same time it makes it easier to use WordNet in larger natural language processing applications, which most of the time are dealing with non lemmas.

Noun		Verb		Adjective	
Suffix	Ending	Suffix	Ending	Suffix	Ending
s		s		er	
ses	s	ies	y	est	
xes	x	es	e	er	e
zes	z	es		est	e
ches	ch	ed	e		
shes	sh	ed			
men	man	ing	e		
ies	y	ing			

Table 3.1: WordNet recognized suffixes and suggested endings

The morphological analyzer, Morphy, is composed from two different steps.

In the first step exception files (one for each part-of-speech category, except adverbs)are used to identify certain English exception words. An example of this can be the string query *'better'* with the lemma *'good'*. The list is kept alphabetically ordered and the word matching is being done by a binary search algorithm. The second step consists from a suffix stripping algorithm, which attempts to replace word endings, in order to find the lemma. This process also requires the part-of-speech category for each word (except adverbs). The rules of suffix stripping, categorized by the part-of-speech, can be viewed in Table 3.1. When a query is being processed by WordNet, Morphy is firstly utilized to retrieve the lemma of that query. First the string is compared to the exception list, and if no hit is generated, the suffix stripping algorithm is being used. The morphological analyzing process employed by WordNet can be seen in Figure 3.2.

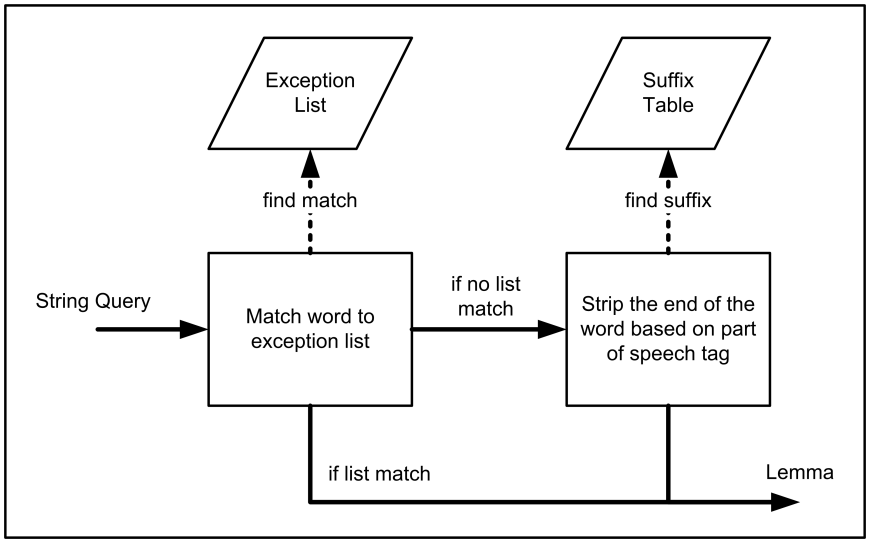

Figure 3.2: The WordNet morphological analyzer

3.4 Summary

In this chapter we described two popular theories that attempt to describe stock market behavior. The efficient market hypothesis (being one of the theories) argues that news and information is already incorporated in the

price of a given stock. The random walk hypothesis (the second theory) argues that one can not deduce the stock price from previous states of the company (such as company information, stock price information). However, there were several authors that were able to find a connection between news articles and a company's stock. There is an existing time interval between the release of a new news item and the impact of it on the stock price.

For the natural language process designed for our application, two components were highlighted in this chapter (with the third following in the next chapter). We recognized the importance of part-of-speech tagging and the morphological analysis of text documents. Three part-of-speech tagging algorithms were discussed: TnT tagger, Stanford bidirectional POS tagger and Brill tagger. The first two algorithms make use of hidden Markov models, with the main distinction being the direction in which the tagger looks, to calculate the tag. While the TnT tagger makes use of the previous two assigned tags to draw a conclusion about the concerning word, the Stanford tagger looks in both directions. The third tagger, the Brill tagger, uses a rule-based approach to assign tags. All three taggers scored very high accuracies, well above the 95%.

To conduct a morphological analysis of a given word, there are different approaches one can choose from. Stemming was one of the more popular approaches, where a suffix stripping algorithm reduced words to their stem. Other not so popular approaches (due to accuracy and processing time) were brute force algorithms and stochastic algorithms. To improve the accuracy of stochastic algorithms, lemmatization was introduced. By assigning the correct part-of-speech tag to a word, the accuracy of the stochastic algorithms improved significantly. Morphy is an implementation of such an algorithm, designed specifically for WordNet.

Chapter 4

Word sense disambiguation

4.1 Introduction

In the previous chapter we discussed the economic foundation for our applications. At the same time, we presented several natural language processing algorithms and tools, which can aid us in our goals. In this chapter we continue the latter, with the focus on word sense disambiguation techniques. This analysis should be used for building the word sense disambiguation module in our application.

4.2 History

Semantic ambiguity has been one of the greatest challenges in designing natural language processing systems. In every natural language, words can have different meanings, depending on the text. For humans this has proven to be a trivial process (in most cases), however, designing a computer algorithm to replicate this process seems to be a demanding task.

Already in the early history of WSD, the skepticism was great. In Bar-Hillel [1964], the authors argue that existing and future electronic computers could not resolve sense ambiguity, because that would first require to model

all world knowledge for computers. The authors presented the following example to prove their point:

> *Little John was looking for his toy box.*
>
> *Finally he found it.*
>
> *The box was in the* ***pen****.*
>
> *John was very happy.*

It was the opinion of the authors, that whether the word ***pen*** would possess two senses, a writing implement and enclosure (first and second sense according to WordNet), an electronic computer would not be able to pick the right one. In the following decades, the WSD process evolved from hand-coded rule-based applications to larger scale lexical resources, where knowledge extraction was crucial. This was followed by the supervised machine learning techniques breakthrough in the 1990s. In the last decade, the focus was on fine-tunement and domain adaptation to more specific topics.

4.2.1 Applications of WSD

In the literature, WSD has traditionally been associated with three tasks: machine translation, information retrieval and knowledge acquisition. It is important to mention that WSD has been perceived as an intermediary process which facilitates a final task.

- **Machine Translation**

  Scientists have always believed that WSD would revolutionize the machine translation field. In Hutchins and Somers [1992] the authors argue however that the problem might be more complex than it appears. A translation machine is confronted with two WSD problems: ambiguity in the first language, from which one attempts to translate from and ambiguity in the second language, in which one attempts to translate to. The first case of ambiguity is trivial, as one has to find the meaning from the context in the first language. The second case

however proves to be a challenge, where for instance a word has in the first language only one meaning and in the second language it might have multiple meanings. However, according to Brown et al. [1991], the accuracy of a machine translation system improves significantly (with over 8%) when WSD is employed.

- **Information Retrieval**

  With information retrieval we refer to search engines processing query inputs (e.g., Google, Yahoo!). Whether search engines would index documents by word senses, search efficiency could be greatly improved, as irrelevant documents would never be retrieved. In practice however, the accuracy improvements are very small. According to Krovetz and Croft [1992], by making use of a perfect WSD example, the accuracy improved only 2% (the authors manually disambiguated text documents). In Sanderson [1994], the author found a relationship between the length of the search query and accuracy: the shorter the search queries, the higher is the accuracy improvement with a WSD system. An explication for this phenomenon is that the standard statistical algorithms employed by information retrieval systems work on a co-occurrence base. The words in a search query then help disambiguate each other, when searched in documents.

- **Knowledge Acquisition**

  Word sense disambiguation might be the most important NLP task for knowledge acquisition. This would allow creating highly intelligent computer agents to find, reason and extract information from any digitalized source (e.g. the Internet). At the same time, WSD algorithms would facilitate an efficient transition to the Semantic Web, where automated annotation of documents is required.

4.3 Methods

There are a number of different approaches to implement a word sense disambiguation solution. However, most solutions can be classified in one of

the two following categories:

1. Corpus-based
2. Knowledge-based

We shall now discuss each approach individually.

4.3.1 Corpus-based approaches

In corpus-based approaches the focus is on employing machine learning techniques on large fragments of text. The underlying idea is that the context itself should provide enough information for word disambiguation. Machine learning models are trained on parts of the corpus, resulting in numerical classifiers. The classifiers are then used to disambiguate new fragments of text. The corpus-based methods can further be divided in two subclasses (typical to machine learning algorithms): supervised and unsupervised disambiguation.

Supervised algorithms

When the training corpus of a machine learning technique is already disambiguated (all words in the text were tagged with the correct sense), we speak of a supervised approach. In the training process, a classifier is constructed, which is then able to correctly label new words with their sense, based on the rest of the context. The earliest attempts to supervised disambiguation were Bayesian classifiers in Duda and Hart [1974], decision trees in Rivest [1987] and k-nearest neighbors and neural networks in Rumelhart et al. [1988].

One of the biggest problems with supervised algorithms is the necessity of large tagged corpus for the classifier training. At the moment there are two large corpora available. The SemCor corpus, as described in Landes et al. [1998], was developed by Princeton University and contains almost 700.000 running words. All the words are part of speech tagged, and more than 200.000 words were lemmatized and sense tagged. The second corpus,

Senseval, is a corpus made specifically for the evaluation of word sense disambiguation systems, according to Kilgarriff and Rosenzweig [2000]. The corpus is derived from the Hector dictionary, an artifact created by Oxford University Press.

There are however ways to surpass the above described problem, by making use of semi-supervised algorithms. The bootstrapping approach makes use of an initial seed (mostly a small number of manually tagged words) and a supervised learning method, to train an initial classifier. This classifier can then be used on the remaining untagged corpus, to classify the most confident instances (which score above a certain level). A new training set is then available (the initial manually tagged set and the new machine tagged set) to retrain the algorithm. Once the training is done, the classifier is used again on an even larger corpus. This process can be repeated a number of times, or until the corpus is exhausted. A number of bootstrapping approaches can be seen in Hearst [1991], Yarowsky [1995] and Mihalcea and Moldovan [2001].

Unsupervised algorithms

A method which does not require large sense-tagged corpora is represented by unsupervised algorithms. One of the assumptions in WSD is that word senses depend on the context. This means that words should retain their sense, as long as the context is the same. By assigning a similarity degree to contexts, a computer would induce word senses. The most popular method for this process is cluster based analysis. For each encountered word sense a cluster can be created. New instances of the word are then assigned to the correct cluster, based on the similarity of the context with the cluster. Examples of such approaches can be seen in Pedersen and Bruce [1997].

4.3.2 Knowledge-based approaches

While the above described methods rely heavily on extensive sense tagged corpora, in knowledge-based approaches the focus is on a machine under-

standable specification of world knowledge (in our case word senses). These methods can primarily be categorized in machine readable dictionaries, thesauri and lexical repositories.

Dictionaries

The first attempt to employ machine readable dictionaries for word sense disambiguation purposes came from Lesk [1986]. The underlying thought was that word co-occurrences in documents are related to each other, and that word relations can be also found in the dictionary entry of each word (e.g., definitions, senses, etc.). The disambiguation process attempts to find a pair of two words with the most overlap in dictionary definition. The Oxford Advanced Learner's Dictionary was used as information source. The accuracy of the system was between 50% and 70%. In Cowie et al. [1992], the authors tried to improve the system by searching for overlaps of an entire sentence, instead of pairs of words. However, this process proved to be computationally inefficient, while there were slight improvements in accuracy.

Machine readable dictionaries are not very efficient in word sense disambiguation, because there is a lack of pragmatic information[1], according to Ide and Véronis [1998]. The dictionaries are primarily made for humans, and not computers.

Thesauri

Contrary to dictionaries, thesauri contain information about the synonyms and antonyms of words[2]. The most popular thesaurus is Roget's International Thesaurus, as described in Hullen [2003]. Several authors proposed WSD systems which make use of the semantic categories that can be found in Roget's Thesaurus. In Yarowsky [1992] and Walker [1987], the authors

[1] An example of pragmatic information is for example the co-occurence of *ash* and *tabacco*, which can not be found in a dictionary.

[2] While the dictionary's main purpose is to define words, the thesaurus tries to give you correct synonyms and antonyms.

attempt to classify words to the categories available in Roget's thesaurus (meaning that the categories act as senses). The first author recorded an accuracy of 92%, however he specified that the method works best with nouns. In Walker [1987], an accuracy of 50% was gained. Again, the same argument can be used as with dictionaries that thesauri are primarily made for humans, and not computers.

Lexical repositories

While the purpose of the previous two data sources is to be useful to humans, lexical repositories are the main information source for computers. Since researchers realized that traditional information sources are insufficient for computer algorithms to be efficient, the need for better data became obvious. Besides WordNet Fellbaum [1998], other important lexical repositories are EuroWordNet in Rodríguez et al. [1998], SIMPLE in Lenci et al. [2000] and ComLex in Grishman et al. [1994]. Since WordNet is the most popular lexicon, in the following section of this chapter we present several word sense disambiguation methods based on this repository.

4.4 WSD approaches based on WordNet

The strongest property that WordNet has to offer over the above described knowledge-based methods, is the word-to-word relationships that are coded in the database. Three databases are implemented in WordNet: for verbs, for nouns and one for adjectives & adverbs together. In each database there are lexical entries, which are associated with several synsets. A synset is a group of synonyms with the same definition, and each synset represents a different sense of a word. In the example bellow we illustrate the WordNet entry for the noun *fence*.

(n) **fence**, fencing: a barrier that serves to enclose an area

(n) **fence**: a dealer in stolen property

For each database, a set of semantic and lexical relations have been defined between the synsets. In Tables 4.1 and 4.2 we present the most important relationships defined by WordNet for two part of speech tags, nouns and verbs. The adjective and adverb relations are restricted to only derivations[3] and antonyms.

Relation		
Hypernym	Definition	Y is a hypernym of X if X is a (kind of) Y
	Example	*canine* is a hypernym of *dog*
Hyponym	Definition	X is a hyponym of Y if X is a (kind of) Y
	Example	*dog* is a hyponym of *canine*
Coordinate	Definition	Nouns or verbs that have the same hypernym
	Example	*wolf* is a coordinate *dog*
Holonym	Definition	Y is a holonym of X if X is a part of Y
	Example	*building* is a holonym of *window*
Meronym	Definition	Y is a meronym of X if Y is a part of X
	Example	*window* is a meronym of *house*
Antonym	Definition	Opposites of each other
	Example	*follower* is an antonym of *leader*

Table 4.1: Noun relationships in WordNet

Relation		
Hypernym	Definition	Y is a hypernym of X if X is a (kind of) Y
	Example	*travel* is an hypernym of *movement*
Troponym	Definition	X is a troponym of Y if to X is to Y
	Example	*to lisp* is a troponym of *to talk*
Entails	Definition	Y is entailed by X if you do X you must do Y
	Example	*to sleep* is entailed by *to snore*
Antonym	Definition	Opposites of each other
	Example	*increase* is an antonym of *decrease*

Table 4.2: Verb relationships in WordNet

Based on the semantic and lexical relations available in WordNet, several WSD approaches have been developed to extract the meaning of words from the context. We shall now discuss each approach individually.

[3]Adverbs are often derived from adjectives, and sometimes have antonyms; therefore the synset for an adverb usually contains a lexical pointer to the adjective from which it is derived.

4.4.1 Path-based similarity

By making use of the hierarchical order of concepts in WordNet (hypernym/hyponym relations), we are able to measure the similarity by calculating the path length between two words. However, there is one problem which is associated with path-based measures. The lexical database which is employed requires being relatively consistent in the interpretation of the relation. This is not the case with WordNet, where concepts lower in the hierarchy are less general than those higher in the hierarchy. This would translate to a different degree of similarity (even with the same path length) between general concepts and specific concepts[4].

Rada et al.

In Rada et al. [1989], the authors implemented a path-based solution, *"by defining the conceptual distance between two concepts as the shortest path through a semantic network"*, according to Pedersen et al. [2005]. The path was constructed with the hypernym/hyponym *(is-a)* and holonym/meronym *(part-of)* relations. By counting the edges in the shortest path between two concepts, the system was able to calculate the similarity between the concepts:

$$similarity_{path}\,(c1,c2) = 1/p \tag{4.1}$$

where p is the shortest path between the two concepts *(c1,c2)*.

Leacock and Chodorow

In Leacock and Chodorow [1998], the same approach as in Rada et al. is used, by calculating the shortest path between two concepts with the hypernym/hyponym relationship *(is-a)*. Once the path has been identified, it is scaled down by the depth D of the hierarchy. The total length between

[4]For instance, the path length between *rodent* and *mouse* is one, which is the same length between *fire iron* and *implement*.

a leaf node and the root node forms the depth of the hierarchy. Thus, the similarity of a path is the following:

$$similarity_{path}\,(c1,c2) = log\left(\frac{(2 * D)}{p}\right) \tag{4.2}$$

where p is the shortest path between the two concepts *(c1,c2)* and D is the maximum length in the hierarchy from the top root node to the lowest node.

Wu and Palmer

While the previous two methods focused on the path length between two concepts, in Wu and Palmer [1994], the authors focus on the path length between a concept and the root of the hierarchy. By finding the distance between two concepts to the nearest specific ancestor node, the algorithm is able to present the similarity between the concepts. This root node is also called the lowest common subsummer *(lcs)*. The similarity is calculated by the following formula:

$$similarity_{path}\,(c1,c2) = \frac{2 * depth\,(lcs\,(c1,c2))}{depth\,(c1) + depth\,(c2)} \tag{4.3}$$

where *depth* measures the distance between the node and the root node.

Hirst and St. Onge

In Hirst and St. Onge [1998], the authors developed an algorithm which takes into account more WordNet relationships, and does not focus only on the traditional *is-a*. This method is at the same time the only path-based method which is able to find the similarity between different part of speech words. To do this, the authors have assigned directions to the paths traveled between two concepts. These directions can be upward, downward and horizontal and are coupled with the different type of relationships found in WordNet[5]. By identifying the nature of the path that joins the concepts,

[5]For instance, *is–a* is an upward relation while *is–a–kind–of* is a downward relation and *antonyms* are horizontal relations.

the algorithm is able to measure the similarity. The similarity is stronger if the path is shorter and the number of direction changes is small, and the similarity is weaker with a long path with many direction changes. The similarity is calculated by:

$$path\_weight = C - path\_length - (k * \#\_changes\_in\_direction) \quad (4.4)$$

where C is 8 and k is 1, according to Budanitsky and Hirst [2001].

4.4.2 Information content similarity

The information content measure is a method which tries to calculate the specificity of a concept in a hierarchy. As already discussed with path-based methods, the similarity of two concepts is very different when the concepts are very general or very specific. This approach tries to identify the information content of a concept, by counting the number of times the concept is mentioned in a large corpus, and calculating a maximum likelihood estimate.

$$IC(concept) = -log(P(concept)) \quad (4.5)$$

The information content of the concept is defined in this case as the negative of the log likelihood *-log(P(concept))*, where *P(concept)* is the probability of encountering the concept in a text. The intuition of using the negative likelihood is that the higher the probability to find a concept in a text, the less information it contains. If the probability increases, the informativeness decreases, making the information content of this concept lower. In Figure 4.1 we can observe an *is-a* taxonomy tree from WordNet. The concept *money* has less information content when compared to *nickel*, because the *P(money) > P(nickel)* (the probability to encounter *money* is greater than the probability to encounter *nickel* in a text).

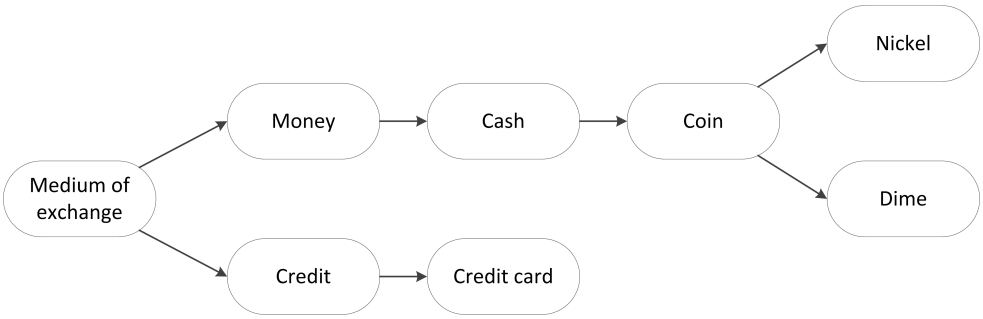

Figure 4.1: Taxonomy tree from WordNet

Resnik

In Resnik [1995], the author tries to combine the lowest common subsummer as described above in Wu and Palmer with information content. The basic idea is that two concepts are similar corresponding to the degree of information they share (the *lcs*). To formulate this:

$$similarity\,(c1,c2) = IC\,(lcs\,(c1,c2)) \tag{4.6}$$

where *lcs* is the lowest common subsummer. Again, the *is-a* relationships is used, however the path length is irrelevant in this case.

Jiang and Conrath

Further developing the method created by Resnik, in Jiang and Conrath [1997] the authors try to establish a link between the difference of information content of the individual concepts and the information content of the lowest common subsummer. They believe that the discrepancy between the two individual concepts and the *lcs*, measures the distance between the concepts in the hierarchy.

$$distance\,(c1,c2) = IC(c1) + IC(c2) - 2 * IC(lcs(c1,\ c2)) \tag{4.7}$$

This distance shows the similarity between the two concepts (if low, the

similarity is high, and vice versa). This can be translated to a similarity formula by taking the inverse according to Pedersen et al. [2005]:

$$similarity\,(c1, c2) = \frac{1}{distance\,(c1, c2)} \tag{4.8}$$

Lin

In Lin [1997], the author translated the method developed by Wu and Palmer, to take into account the information content of the concepts. At the same time, there is a big resemblance between this method and the one discussed previously, by Jiang and Conrath. In both cases the information content of the *lcs* is compared to the information contents of the two individual concepts. This has been formulated as following:

$$similarity\,(c1, c2) = \frac{2 * IC(lcs(c1, c2))}{IC(c1) + IC(c2)} \tag{4.9}$$

4.4.3 Graph-based similarity

While the previous two categories (path-based and information content similarity) are static of nature (the similarity is calculated individually without taking into account the rest of the sentence), graph-based similarity tries to determine the sense of the words collectively, by making use of the dependencies between the senses. Graph-based methods consist mainly from three steps Sinha and Mihalcea [2007]:

1. a graph has to be created, in which all word sense dependencies are calculated
2. calculate the sense scores according to the graph-based algorithms
3. select the highest score for the each word

For each sentence in a text, a new graph $G : V \rightarrow E$ has to be created. Each node $v \in V$ in G represents a sense of a word in that sentence. The

algorithm starts by identifying all nodes $v \in V$ and computing all edges $e \in E$ according to a dependency algorithm, as illustrated further. Once all edges are computed, the graph-based algorithm computes the best path in G. An example illustration of such a graph, with three words totaling six senses, can be observed in Figure 4.2 $v \in V$ represents a sense of each word, while the edge is computed with a similarity aglorithm. Each node contains the score computed by a graph-based algorithm.

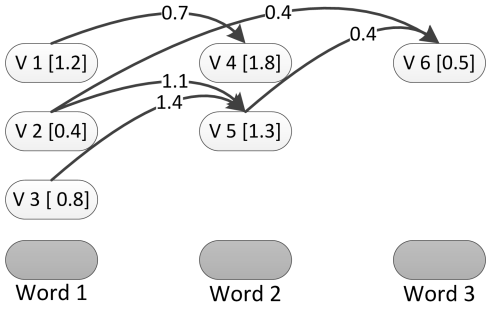

Figure 4.2: An example word sense disambiguation graph

Dependency algorithms

There are multiple metrics which can be used to calculate the word sense dependency within a graph. These metrics have two basic requirements in order to work for graph-based similarity. In the first place a machine readable dictionary is required, in order to automate the process. Furthermore the metrics should not rely on limited semantic relationships, which only apply to one part-of-speech category as with many WordNet similarity algorithm examples.

1) Hirst and St. Onge [1998] can be used as a dependency algorithm, as it is not limited to one specific part-of-speech category. In Mihalcea [2005] the author proposes a new metric, which computes the token sense overlap of two words. Taking a pair of words, the algorithm calculates the number of common tokens per sense between the words.

2) In Sinha and Mihalcea [2007] the authors combine the previously discussed similarity algorithms from subchapter 4.4.1 (Path-based similarity)

and 4.4.2 (Information content similarity) to calculate the dependency between words within a graph. In order to contain all the part-of-speech categories, they have developed a combined algorithm: Jiang and Conrath for nouns, Leacock and Chodorow for verbs and Lesk (as described in Lesk [1986]). For nouns and verbs the combination was determined by using the highest performing similarity measure per part-of-speech connection, and Lesk would measure all the remaining part-of-speech categories (adjectives, adverbs).

3) Navigli and Lapata [2010] propose a new method of building a graph, which has no weights assigned to the edges between the nodes. This method tries to recreate a WordNet graph with only the relevant nodes and edges. Given a sentence and all the senses of the words within that sentence, they search the WordNet graph using a depth-first strategy for each sense, until they find a new sense which exists in the current list of senses. If such a sense is found, that and all intermediate nodes and edges are added to the new relevant graph. As already mentioned, the WordNet edges only represent semantic relations between senses, without numeric weights, meaning the resulting relevant graph is unweighted.

Graph-based algorithms

The graph-based algorithms in this context calculate the influence of a node within a graph in relation to the other nodes. These algorithms can be further split in two categories: local and global. While local algorithms focus on the centrality of the node within a graph, global algorithms measure the overall structure of the graph instead of the node. In several tests (Navigli and Lapata [2010] and Navigli and Lapata [2007]) global graph-based algorithms performed relatively bad compared to the rest, so we will not discuss them.

1) Degree centrality: The simplest and fastest way to measure the centrality of a node is to count the number of edges connecting to a node in an undirected graph. This results in two algorithms however, one if the graph is unweighted and the edges have no 'distance' assigned to it, and the second

if the graph is weighted and each edge has a 'distance' assigned to it.

In the first case the degree of the node is calculated by summing the number of edges incident to the node, and normalize it by the maximum degree. In an undirected unweighted graph $G = (V, E)$, the degree of a node $deg(V_a)$ is calculated as follows:

$$deg\left(V_a\right) = \frac{|\{\{V_a, V_b\} \in E : V_a \in V\}|}{|V| - 1} \tag{4.10}$$

where $|\{\{V_a, V_b\} \in E : V_a \in V\}|$ represents the number of edges betewen nodes V_a and V_b in G.

In the second case, the degree of a node is calculated by summing the weights of the edges which are connected to a specific node. In an undirected weighted graph $G = (V, E)$, the degree of a node $deg(V_a)$ is calculated as follows:

$$deg\left(V_a\right) = \sum_{(V_a, V_b) \in E} w_{ab} \tag{4.11}$$

where w_{ab} represents the weight betewen nodes V_a and V_b. As we can see, this equation has no normalization as the previous one. This was due to the fact that normalization was already calculated in the weight of the edge.

2) Betweenness centrality: This centrality method calculates how often a node is on the shortest path between other nodes. The more often, the more central. Again we have unweighted and weighted calculation for a undirected graph.

The weighted algorithm is:

$$betweenness\left(V_a\right) = \sum_{V_b \in V, V_c \in V} \frac{\sigma_{V_b, V_c}\left(V_a\right)}{\sigma_{V_b, V_c}} \tag{4.12}$$

where σ_{V_b, V_c} represents the number of shortest paths between V_b and V_c and $\sigma_{V_b, V_c}\left(V_a\right)$ the number of shortest paths between V_b and V_c that pass through V_a.

For the unweighted algorithm, a normalization is required:

$$normBetweenness\left(V_a\right) = \frac{betweenness\left(V_a\right)}{\left(|V|-1\right)\left(|V|-2\right)} \tag{4.13}$$

where $\left(|V|-1\right)\left(|V|-2\right)$ represents the number of pairs of vertices excluding V_a.

3) PageRank: PageRank is based on the eigenvector centrality, a method which measures the importance of a node within a graph. The algorithm assigns relative scores to a node based on the following principle: high scoring connected nodes add more value to the score of the selected node compared to low scoring nodes. This is done recursively based on a Markov chain model:

$$PageRank\left(V_a\right) = (1-d) + d * \sum_{(V_a,V_b)\in E} \frac{PageRank\left(V_b\right)}{|deg\left(V_b\right)|} \tag{4.14}$$

where d represents a constant for the random-walk model as recreated in a Markov chain model (the walker takes random steps in a graph). d is set to 0.85 based on Brin and Page [1998].

In a weighted graph, the importance of a node is replaced by the weight of the edge between two nodes:

$$PageRank\left(V_a\right) = (1-d) + d * \sum_{(V_a,V_b)\in E} \frac{w_{ab}}{\sum_{(V_c,V_b)\in E} w_{bc}} PageRank\left(V_b\right) \tag{4.15}$$

where w_{ab} represents the weight between node V_a and V_b.

4.4.4 Discussion

In Figure 4.3 we can see all the word sense disambiguation categories discussed in this chapter. The most important for our research is however WordNet, and algorithms based on WordNet. This can be observed in Figure 4.4. Based on this figure, we created Table 4.3 illustrating a general

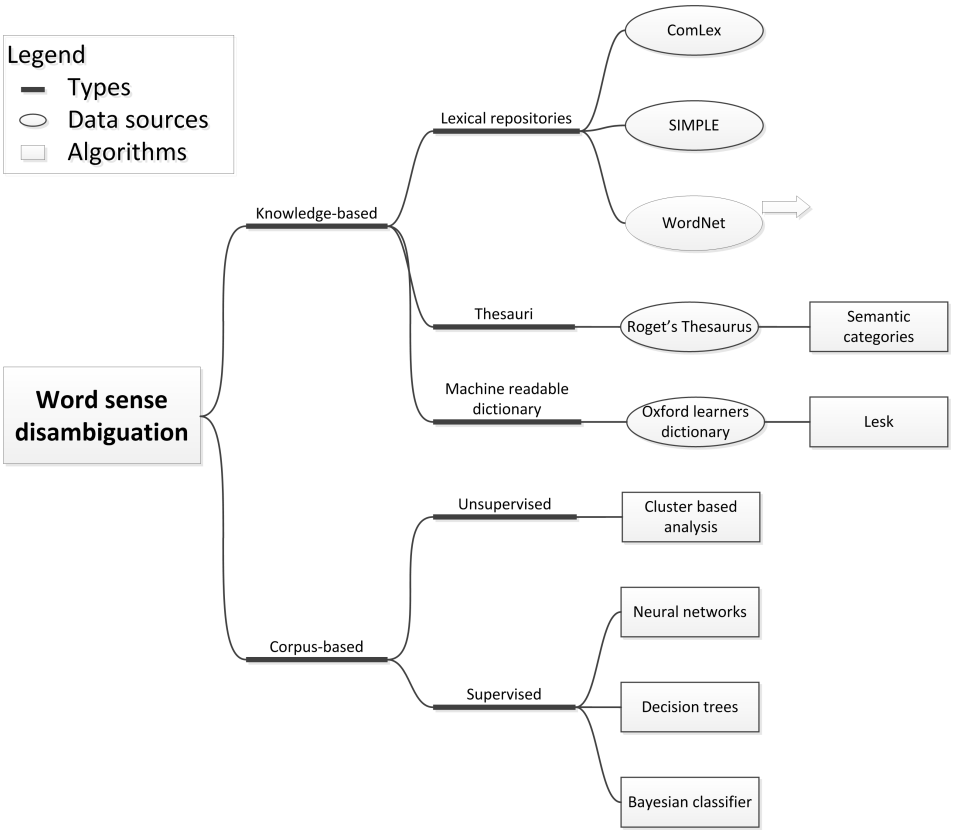

Figure 4.3: A representation of all types of word sense disambiguation, organized by type

classification of the similarity algorithms accompanied with their advantages, disadvantages and precision where possible (for some algorithms only the F-measure was reported). The results for the path-based and information content algorithms were reported by Pedersen et al. [2005], graph-based algorithms (except Navigli) by Sinha and Mihalcea [2007] and the rest by Navigli and Lapata [2010].

As we can observe, between path-based and information content algorithms, Jiang & Conrath performs best. When looking at graph-based algorithms, the degree measure performs the best for Sihna et al. and Navigli.

Type	Creator	Methodology	Advantage	Disadvantage	Precision
Path based	Rada et al.	Count edges in shortest path	Simple	Requires rich lexical knowledge base No multiple inheritence is-a relations only	F-measure Noun: 0.24 Verb: 0.16 Senseval 2
	Leacock & Chodorow	Shortest path between concepts, with log scaler	Simple Corrects for hierarchy depth	is-a relations only	F-measure Noun: 0.24 Verb: 0.13 Senseval 2
	Wu & Palmer	Path length to subsummer scaled by subsummer path to root	Simple	is-a relations only	F-measure Noun: 0.29 Verb: 0.05
	Hirst & St. Onge	Takes into account all WordNet information	Not only is-a relations	WordNet specific	F-measure Noun: 0.21 Verb: 0.09 Senseval 2
Information content	Resnik	Information content of the least common subsummer	Takes advantage of empirical statistics	Does nothing with individual lcs is-a relations only	F-measure Noun: 0.30 Verb: 0.05 Senseval 2
	Jiang & Conrath	extension on Resnik	Takes into account individual concepts	is-a relations only	F-measure **Noun: 0.40** **Verb: 0.19** Senseval 2
	Lin	extension on Resnik and Wu & Palmer		is-a relations only	F-measure Noun: 0.33 Verb: 0.06 Senseval 2
Graph based	Mihalcea & PageRank	Random walk on graph with encoded label dependencies and weights for the edges	Simple	Based on token overlap with no implicit connection Computational expensive	Noun: 57.5% Verb: 36.5% All: 54.2% Senseval 2
	Sihna et al. & PageRank	Combined similarity algorithm: jcn for nouns, lch for verbs, lesk for rest	Combines the best of all similarity algorithms	Computational expensive	Noun: 60.5% Verb: 41.7% All: 52.8% Senseval 2
	Sihna et al. & Degree	Same as above, only a different graph measure method	Combines the best of all similarity algorithms	Computational expensive	Noun: 61.1% **Verb: 43.3%** All: 53.4% Senseval 2
	Sihna et al. & Betweenness	Same as above, only a different graph measure method	Combines the best of all similarity algorithms	Computational expensive	Noun: 49.4% Verb: 20.7% All: 39.5% Senseval 2
	Navigli & PageRank	Random walk on a graph based on WordNet graph	Makes use of WordNet explicitly encoded connections	Computational expensive	All: 49.7% SemCor
	Navigli & Degree	Same as above, only a different graph measure	Makes use of WordNet explicitly encoded connections	Computational expensive	**Noun: 61.9%** Verb: 36.1% All: 50.0% SemCor
	Navigli & Betweenness	Same as above, only a different graph measure	Makes use of WordNet explicitly encoded connections	Computational expensive	All: 48.7% SemCor

Table 4.3: Comparison between different similarity measure methods

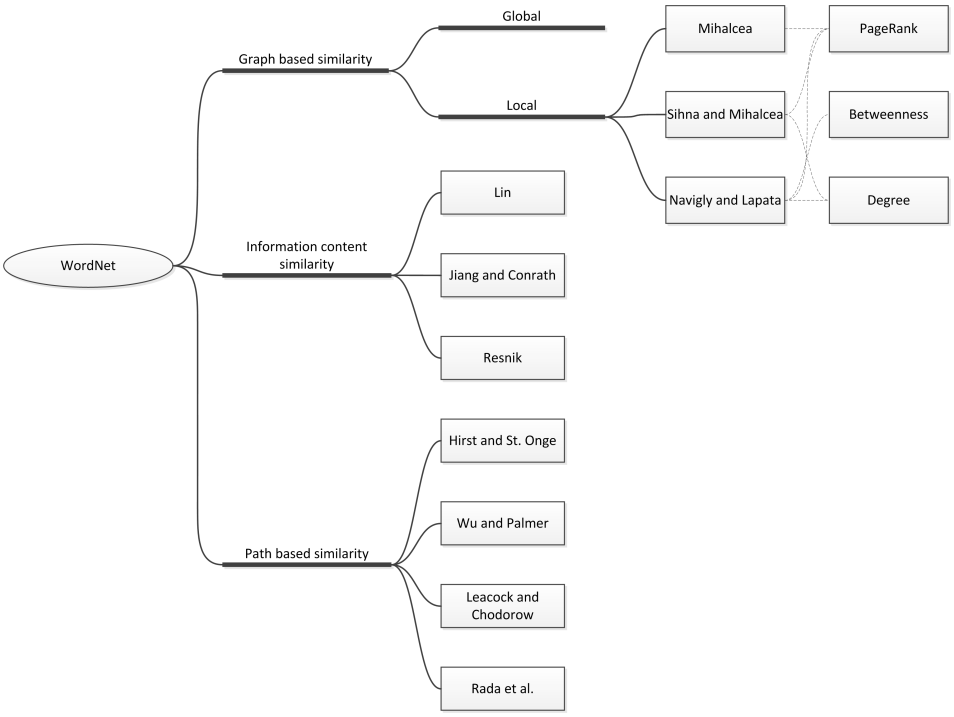

Figure 4.4: All similarity measure methods based on WordNet, organized by type

4.5 Conclusion

There are several methods in order to resolve the semantic ambiguity of words. In this chapter we discussed these methods, categorized on the approach employed: corpus-based and knowledge-based. Corpus-based approaches rely on machine learning techniques, where prediction models (Bayesian classifiers, decision trees and neural networks) are developed and trained on large disambiguated corpora. The unsupervised method of this approach does not require a corpus, but tries to identify the different meaning of the words on its own, with cluster classification. For every word meaning, a cluster is formed.

The knowledge-based approach relies on some form of machine understandable representation of the world. This can either be a dictionary, a thesauri or a lexical repository. However, the design of dictionaries and thesauri

proved to be inefficient in word sense disambiguation, as both are primarily made for humans, and not computers. The use of lexical repositories in combination with word sense disambiguation is a growing trend. WordNet is the most popular lexicon. The strongest property that WordNet has to offer over the other knowledge-based methods is the many types of word-to-word relationships that are coded in the database.

There are several approaches to the word sense disambiguation problem, using WordNet. In the path-based measures, the distance in the WordNet graph between the words is calculated. The first implementation of this was by Rada et al. This algorithm was extended by Leacock & Chodorow, by scaling the similarity with the depth of the hierarchy. Another path-based method is presented by Wu & Palmer, where the similarity of two words is given by finding the distance between the two concepts to the nearest specific intersection root node (lowest common subsummer). The last path-based algorithm, developed by Hirst & St. Onge, calculated the similarity between two words by the turns change in directions in the graph. The smaller the number of direction changes, the higher the similarity.

The information content (IC) approach tries to identify how specific or general a word is, by counting the number of times the concept is mentioned in a large corpus, and calculating a maximum likelihood estimate. The first implementation of this approach was done by Resnik, where the basic idea is that two concepts are similar corresponding to the degree of information they share in common (the lowest common subsummer). Jiang & Conrath expanded this, by taking the difference between of the individual ICs and two times the shared IC of the lowest common subsummer. A similar approach to the Wu & Palmer one was developed by Lin, where the similarity is represented by the IC of the intersection root node.

Graph-based similarity tries to determine the sense of the words collectively, by making use of the dependencies between the senses. Word senses are mapped to nodes, and relations between the senses become edges. Mihalcea created a framework for graph-based algorithms: 1) calculate sense dependencies in a sentence, 2) calculate sense scores based on graph-based algorithms, 3) select the correct sense for each word. For the first step

there are several choices: a token sense overlap between two words and a combination of path-based & IC similarity algorithms to calculate the dependencies between nodes. Navigli & Lapata proposed an alternative to the first step, by creating a weightless undirected graph based on the WordNet graph between two words. The second step of the framework can be filled in by a number of graph optimization algorithms: degree centrality, betweenness and PageRank. These algorithms measure the importance of the node within the graph.

Jiang & Conrath was reported to perform the best in the path-based and IC similarity pool, while degree centrality was the best performing graph-based algorithm. Furthermore, it is important to note that the computation time of graph-based similarity algorithms exceeds that of path-based and IC, as you need to disambiguate an entire sentence compared to two words of the same part-of-speech.

Chapter 5

Architecture

5.1 Introduction

In this chapter we explain the design of our application. Furthermore we elaborate on the choices made with respect to the different implemented modules.

We begin by giving an overall view of the application architecture. This is followed by a detailed examination of the text processing module (with the focus on the word sense disambiguation algorithm). We also give attention to the user interface, and more importantly, the manner in which the news are presented.

5.2 StockWatcher

Many of the ideas motivating this research have originated from a previous project, Micu et al. [2008]. In this project, a prediction system was developed which would rate news articles as having a positive or negative influence on the sentiment of a given company. However, the changes required to adapt this project to our current goals would prove inefficient, so we created a new application, called StoackWatcher 2.0.

In Figure 5.1 we can observe the overall process of our new application, and the flow of data. There are three distinct data sources required, in order for the application to run correctly. The events originate from a comma separated value (csv) file from the server. The financial database is also a csv file, and contains historical stock prices for NASDAQ-100 listed companies. This database gets automatically updated every day. The news sources represent RSS-feeds or locally stored news items (text format). We have preconfigured the application already with several popular RSS feeds, but the user has the ability to customize it with custom RSS feeds.

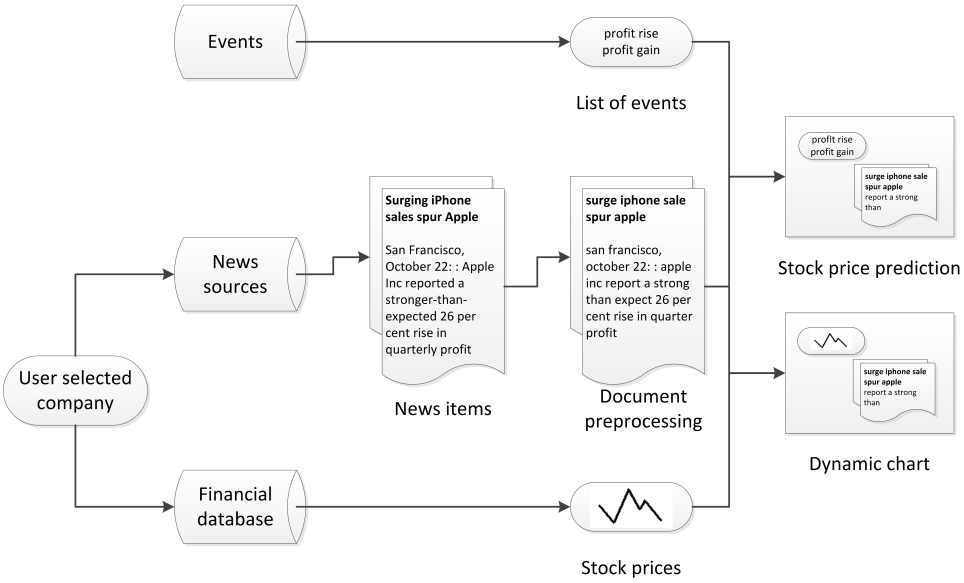

Figure 5.1: StockWatcher 2.0 conceptual model of the information flow.

Once a user engages the application, and the company has been selected, the application loads the three data sources. The news items undergo a series of text processing algorithms, after which they are rated based on event matches. Every article receives a stock price prediction, which indicates the effect of the news item on the closing price that day. Once all news items have been rated, they are presented to the user chronologically, together with a dynamic flash chart (which is generated from the historical stock prices). This provides a user friendly interface to interact between news items and previous stock prices.

5.3 Document preprocessing and article rating

Before an article is rated, it must be processed to improve machine readability. As already discussed in chapter 3, a natural language processing system has to be developed, with the following components: a tokenizer, a POS-tagger and a morphological analyzer. Furthermore a word sense disambiguation component has to be present, in case disambiguation is required. We implemented all these requirements in our application, as depicted to Figure 5.2.

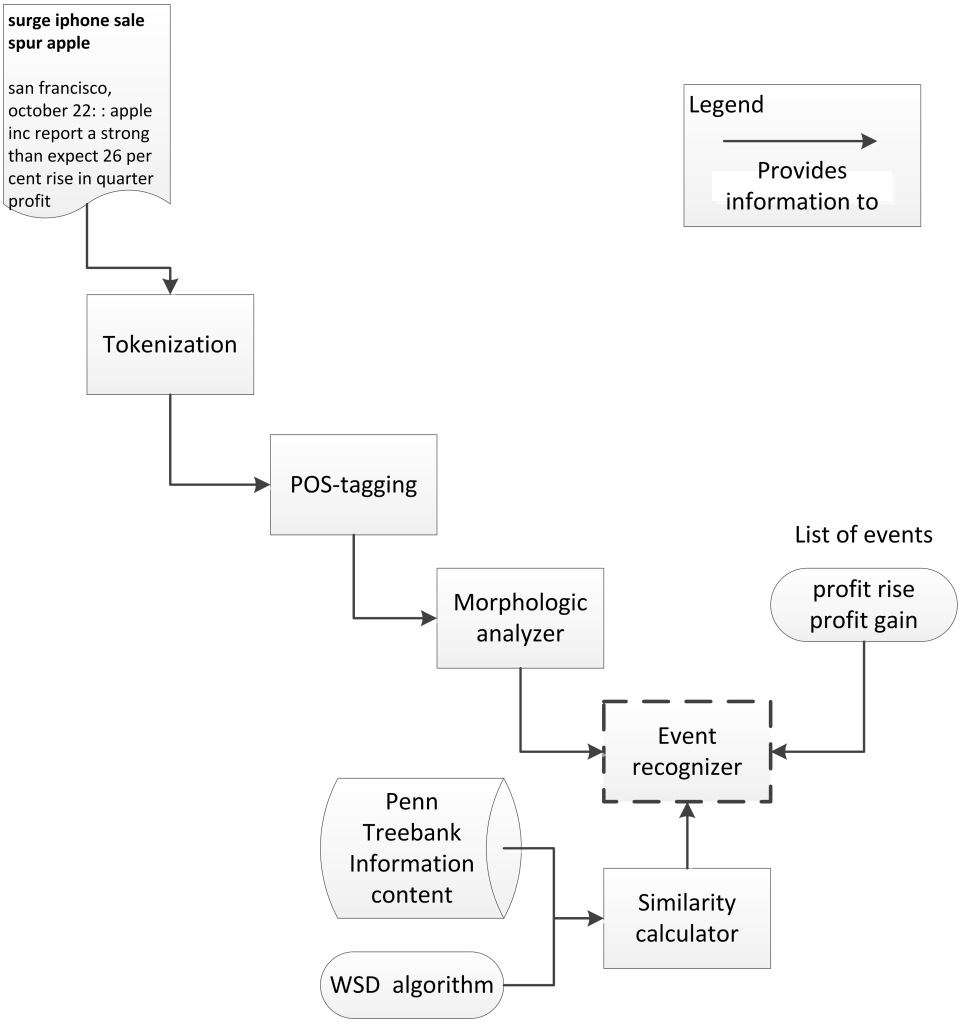

Figure 5.2: StockWatcher 2.0 NLP pipeline.

As we can observe in Figure 5.2, the first step in the sequence is the tokenization module. The tokenization process makes sure that the text is split in correct sentences, and unnecessary white spaces and non-word characters are deleted (for example punctuation marks). This is done by constructing regular expression-based patterns with the unnecessary characters, and upon identification within a sentence removing them.

For the part-of-speech tagging we implemented the Stanford POS tagger, released by the Stanford Natural Language Processing Group. The bidirectional tagger has the highest accuracy, and an English trained model is already available. For more information on the used types, the Stanford NLP Group website can be visited[1].

Once the sentence is tagged with the correct part-of-speech, the application finds the lemma of each word in the WordNet lexicon. This process is done by a module called the Java WordNet Library (JWNL), further described in Didion [2007]. JWNL is an application programming interface (API) that is required to access the WordNet dictionary. This module enables morphological processing and relationship discovery (which is useful for word sense disambiguation). The lemmatization takes place word for word, and once all the words have been processed, the news article is being rebuilt, with the new lemmas. The following example is taken from an article:

```
"_struggled and won the fiercest_"
```

would be transformed into

```
"_struggle and win the fierce_"
```

Transforming the article into its lemmatized representation enables us to keep the knowledge base of economic events minimal. There is no need to take into account different forms of the words (for example for the verb 'go' we don't need to store: 'goes', 'going', 'went', 'gone').

[1] http://nlp.stanford.edu/software/tagger.shtml

5.3.1 Event recognition

Most of the economic events in the csv file originate from Micu et al. [2008], with some minor improvements. The events are divided in groups, according to their economic origin and synonymity. For example: events regarding the stock price (and all their synonyms) were placed in one group, events regarding the income/earnings (and all their synonyms) were placed in another group, etc. Each line in the csv file represents an event and its characteristics. All the event groups can be found in Appendix A. Each event is encoded in the following manner:

$$event; word\_sense; POStag; minimal\_word\_match; weight \tag{5.1}$$

The first encoded block, *event*, represents the full event name (an event can have one or several words). The second block represents a sequence of senses for each word from the first block (coded as numbers, and the number of senses should be equal to the number of words). The numbers are related to the WordNet sense numbers. The third block is a string representation of the part-of-speech per event word. The fourth block is a number, which indicates what are the minimum number of words from each event that should be recognized in a sentence, for the event to be valid. For one word events, this is 1, but for multiple word events this number can be greater. The last block is the weight assigned to the event (positive or negative). Furthermore we have the following distinction between the events: ambiguous events which require word sense disambiguation (a word sense and POS-tag is given) and events where the semantic similarity is not important, as long as the full event can be found in a sentence. We will call the first group *AmbEvents* (ambiguous events) and the second *DisambEvents* (disambiguous events).

```
release;5;VB;1;1.88
```

In the example from above we have an event from the *AmbEvents* group. The event is *release*, with WordNet sense 5: *"prepare and issue for public*

distribution or sale", with POS-tag verb, only 1 word to match, and a weight of 1.9.

```
share rise;0 0;null null;2;4.7
```

In this example word disambiguation is not important, as it is reasonable to expect that if this exact word structure would be found in a sentence, the meaning would be the same. First we have the event *share rise*, with no meaning assigned (sense 0 does not exist in WordNet) and null POS-tag (the 0 and the null are based on our encoding scheme). Both words are required to be present in a sentence, and the weight assigned to it is 4.7.

The event distinction allows the event recognition module to differentiate between events which require disambiguation (*AmbEvents*) and events where the sense disambiguation is not an issue (*DisambEvents*). We illustrate the process for *DisambEvents* with the following sentence example for the *Apple* company: *"Shares of Apple rose 12 percent after posting a 48 percent gain in net income"*. The tokenization would split up the sentence in words and would remove common words such as *of*, *a*, *in* and other non text characters such as numbers and semicolons. The POS-tagging module assigns a correct tag to each word, making it possible to lemmatize each word. The result so far is as following: 1. *share*, 2. *apple*, 3. *rise*, 4. *percent*, 5. *after*, 6. *posting*, 7. *percent*, 8. *gain*, 9. *net*, 10. *income*. The application is now capable of matching the event *share rise* to the words in the sentence. If the *#minimal_word_match* condition has been satisfied, the rating can be calculated.

For the *AmbEvents* set, after the event has been identified and the *#minimal_word_match* condition has been satisfied, the algorithm proceeds to identify the WordNet sense number for the event. More on this subject in the next sub chapter.

Word sense disambiguation

All events in the *AmbEvents* group are either verbs or nouns. If we are to select from the word sense disambiguation algorithms from Chapter 4,

we have the following options: graph-based on degree Sihna et al. and Navigli. As we can observe in the results, the Sihna et al. graph-based degree approach yields almost identical results for nouns (Sihna et al. precision of 61.1% compared to Navigli 61.9%) however outperforms for verbs (Sihna et al. precision of 43.3% compared to Navigli 36.1%). For this reason we selected Sihna et al. graph-based degree algorithm as the word sense disambiguation technique.

For word sense disambiguation we employ GWSD: Unsupervised Graph-based Word Sense Disambiguation as described in Sinha and Mihalcea [2007]. The library contains the required WordNet similarity measures, Jiang & Conrath (JnC) and Leacock & Chodorow (LCh). Furthermore it contains the degree centrality algorithm to perform our tests. Based on the target word (in this case the event which was found in a sentence) we create a graph with all the words which share the same POS-tag with the target word. Based on the target word POS-tag, all the word sense dependencies are weighted according to JnC for nouns, and LCh for verbs. The second step is to calculate the degree centrality for each sense. This allows us to find the highest scoring sense for our target word.

In Algorithm 1 we can observe the pseudo code for the graph-based degree centrality similarity calculation. Based on the target word found in the sentence, we create an array containing all the words from the sentence of the same POS-tag type as the target word. From this array a graph is created where each vertice represents a sense $s^t_{w_i}$ for each word w_i. All the vertices are assigned weighted edges based on the similarity measure (JcN for nouns, LCh for verbs). For more information see chapter 4, figure 4.2. Once the graph G is ready, each vertice is assigned a score based on the degree centrality algorithm. This is done by summing the weights of the edges connected to each vertice. Now we can find the correct sense for our target word, by selecting the highest scoring vertice.

Algorithm 1 The word sense disambiguation based on graphs and degree centrality

Input: array with words w_i, where $i = 1$ to N (last word in sentence), where all words are the same POS-tag as the target word

Input: admissible senses of $s^t_{w_i}$, where $t = 1$ to N_{w_i} (last sense for w_i)

Output: Graph G

for $i = 1$ to N **do**
  for $j = i + 1$ to N **do**
    for $t = 1$ to N_{w_i} **do**
      for $y = 1$ to N_{w_j} **do**
        $weight \leftarrow Similarity(s^t_{w_i}, s^y_{w_j})$ {JcN for nouns, LCh for verbs}
        if $weight > 0$ **then**
          $AddEdge(G, s^t_{w_i}, s^y_{w_j}, weight)$
        end if
      end for
    end for
  end for
end for

Input: Graph G with vertices V

Output: Graph G with each vertice V scored according to degree centrality

for $V_a \in Vertices(G)$ **do**
  $Score(V_a) \leftarrow DegreeCentrality(V_a)$
end for

Input: vertice for target word x

Input: scored admissible senses of s^t_x, where $t = 1$ to N_x (last sense for x)

Output: $WinnerSense$ for target word x

for $t = 1$ to N_x **do**
  $MaxSimilarity = 0$
  $WinnerSense = 0$
  if $MaxSimilarity < Score(s^t_x)$ **then**
    $MaxSimilarity = Score(s^t_x)$
    $WinnerSense = t$
  end if
end for

Distance factoring

In Micu et al. [2008], one of the problems we encountered is that news items were being categorized correctly (positive or negative), however for the wrong company. In an article regarding a company, it is common that news about competitor companies are mentioned as well. If the application would find the event "income growth" in the text, and the company of interest is not mentioned in the neighborhood, it is unclear if this event has any relevance. One of the lessons learned in Micu et al. [2008] was that sentences where the target company was mentioned were relevant, including the preceding and succeeding sentence. All other sentences might be referring to a different company, and should be ignored. For this reason we applied a distance factoring filtering on each news article, which discards all sentences that are not in the vicinity of the target company based on the above described rule.

5.3.2 Event training

In order to assign the correct weights to the events, a training set with news articles is created. To determine the impact of an event on the stock price we made use of a method described in Cesarano et al. [2006]:

$$event\_weight^c(w) = \frac{\sum_{d \in D_{test}} \left(avsc^c(d) * \frac{n^c(w,d)}{\sum_{w' \in oew^c(D_{test})} n(w',d)} \right)}{\sum_{d \in D_{test}} avsc^c(d)} \quad (5.2)$$

with the following definitions:

1. $n^c(w,d)$ Denotes the number of occurences of event w (and its synonyms) on day d for company c.
2. $avsc^c(d)$ Indicates the percentual difference between the opening price and the closing price on the same day d for company c. In this case $d+2$ or $d+4$ indicates the percentual difference between opening price on day d and closing price on $d+2$ or $d+4$ respectively.

3. D_{test} Refers to the entire set of test days.

4. $oew^c(d)$ Denotes the total number of event occurences on day d for company c.

5. $event\_weight^c\,(w)$ The event weight is calculated per company c, indicating that the same event can have different weights depending on the company.

This method has its roots in statistics as discussed in Sheldon [2002]. The purpose is to calculate the contribution of the stock price change to the relative proportion of the event on a given day. If we average this contribution for the entire test case (all the days in the test set) we can calculate the event weight for the entire test frame. Applying this method will result in an event document for each company in the training set.

For example, the *Google* event document can have the following event entries (and their respective weight):

- stock decline, -0.321
- revenue increase, 0.731
- launch, 0.121

while the *Apple* event document can have the following entries:

- stock decline, -0.210
- revenue increase, 0.338
- sign deal, 0.443

As we can observe, both event documents have 2 events in common, however the event *launch* has only been trained for *Google*, and the event *sign deal* has only been trained for *Apple*.

Once the contribution of each event towards the stock price change for a certain company has been calculated, we can calculate an average event weight based on the individual company event weights:

$$event\_weight\,(w) = \frac{\sum_{c \in es(b)} event\_weight^{c}\,(w)}{occurences^{es(b)}\,(w)} \tag{5.3}$$

with the following definitions:

1. $event\_weight\,(w)$ The average event weight for event w.
2. $es\,(b)$ Denotes the the entire event set for all companies.

In this case, the system produces only one event document, and by extending the previous example we have the following entries:

- stock decline, (-0.321+-0.210)/2
- revenue increase, (0.731+0.338)/2
- launch, (0.121)/1
- sign deal, (0.443)/1

5.3.3 Article rating

Once all the events are weighted, the application is capable to rate new news articles. To assign ratings to the articles based on the event weights, the following method is used:

$$result^{c}\,(d) = \sum_{w \in eo(d)} event\_weight^{c}\,(w) \tag{5.4}$$

with the following definitions:

1. $result^{c}\,(r, d)$ Denotes the percentual stock price change for day d for a given company c. In this case $d+2$ or $d+4$ indicates the percentual difference between opening price on day d and closing price on $d+2$ or $d+4$ respectively.

2. $event\_weight^c\,(w)$ Refers to the event weight for event w for a given company c.

3. $eo(d)$ Denotes the total number of event occurences on day d.

This method identifies all events in the news articles on day d for a given company, and calculates the percentual stock price change expected at the end of the day (or over 2 days, etc. depending on the selection of d).

5.4 Conclusion

In this chapter we have presented the detailed design of our application. At the foundation of the application there are three important data sources. An encoded event list, where each event has several characteristics and a weight; a financial database with historical stock prices; the news articles that can be provided from RSS feeds or from the local disk. All these data sources are then used by the application to calculate a rating for news article, and present the results to the user.

Before a news article can be rated, a series of steps have to be followed first:

- the text is first tokenized in sentences;
- all the words are POS-tagged;
- all the words are lemmatized by the morphological analyzer.

After these steps are fulfilled, the rating of the article takes place. For this process a word sense disambiguation method is applied, provided by the Unsupervised Graph-based Word Sense Disambiguation package. The package contains a degree centrality algorithm, which calculates the correct sense of a word based on a graph built according to a sentence structure. The similarity algorithm is implemented in an event recognition module, which attempts to identify the event in a sentence, and assign the correct rating. The rating is based on the following factors: event ambiguity (if the

sense of an event in an article does not correspond to our event sense or is not clear) and weight of the event.

To assign weights to the events, a training set is used with news items. The application assigned weights to the events based on which events it found in articles per day, and the percentual change in closing price. For each company an event document containing all the events found in the training set for that company and the relative weight of the event based on the percentual stock price change. This results in multiple event documents, 1 for each trained company. Once this is finished, an average event weight based on the individual company event weights is calculated as well.

Chapter 6

Validation

6.1 Validation setup

In this chapter we discuss the validation of the results of our application. For this purpose we are testing the precision and excess return of our application with the following parameters:

- with $c(d)$, which represents same day closing price for company c;
- with $c(d+2)$, which represents the closing price in d plus two days;
- with $c(d+4)$, which represents the closing price in d plus four days;
- with $i(d)$ where $i(d) = c(d) - Index(d)$, where $Index$ represents the NASDAQ Index percentual closing price change;
- with $i(d+2)$, defined similarly as above;
- with $i(d+4)$, defined similarly as above;

These parameters are computed with word sense disambiguation (WSD), as well as no word sense disambiguation (NOWSD). Furthermore the company specific event weights (denoted as $event^c(w)$) are used as well as the average event weights (denoted as $event^a(w)$).

To validate our approach, the application was trained according to the above mentioned parameters resulting in 12 event weights per company (WSD with the six parameters as described above, and NOWSD with same parameters). The training set consists from the 48 largest companies listed on NASDAQ. The articles for the training set are gathered from Dow Jones Newswires[1]. The time frame for the training set was established between 1 January 2010 and 31 December 2010. In total 16685 articles were scanned (348 articles on average per company, with a standard deviation of 495). More details on the news distribution per company can be found in the appendix.

To validate the training results a second time frame was used for the same 48 companies, from 1 January 2011 to 31 December 2011. The articles were again gathered from Dow Jones Newswires and the same parameters were used as described above. The predicted rating indicates the expected percentual change from the opening price on the date of the article apparition. This allows us to compare the results with the actual closing prices and calculate the precision of the application with the different parameters used. More information regarding the validation set can be found in the appendix.

To compare our results with actual closing prices we employ two rating systems. The first one is based on BUY / SELL indicators, where our application (based on the calculated rating) gives a BUY (if the rating is above 0) / SELL (if the rating is below 0) signal. We then proceed to compare these indicators with actual market data for the dates and companies in our validation set. The precision in this case is calculated by: number of correctlly identified signals related to the number of identified signals. We average the precision over all companies.

The second rating system is based on the excess returns generated by our BUY / SELL indicators. We define excess returns as the investment returns generated by our portfolio that exceed the NASDAQ index returns for the same time period. For the period of the validation set (2011), the NASDAQ

[1]Dow Jones Newswires is a real-time news organization with a network of 2,100 journalists, with more than 19,000 daily news items: *http* : *//www.dowjones.com/factiva/index.asp*

index produced negative returns of -1.80%[2]. When the application calculates a BUY indication for a certain time period (d, $d+2$ or $d+4$) the strategy is to go long in shares for that company for the indicated time period. When the application calculates a SELL indication, the strategy is to go short in shares for that company for the indicated time period. The starting capital for the investment is 1000$ per company (resulting in a total of 48000$). For the period of 1 January 2011 to 31 December 2011 the capital is invested chronologically per company, per BUY / SELL signal.

To illustrate the approach lets consider the following example. For the date of 6 January 2011 the application calculates an expected 0.652 percentual rise in the stock price for *Apple*, for the $c(d)$ parameter (this is the first chronological observation for *Apple*, so the investment capital is 1000$).

1. 0.652 is positive, resulting in a BUY indication;
2. Go long in *Apple* shares for the value of 1000$;
3. The price of the shares increase with 4% at the end of the day;
4. Sell the shares of *Apple* resulting in a profit of 40$. The current investment capital is 1040$;

For the date of 7 January 2011, the application calculates an expected -0.731 percentual decline in the stock price for *Apple*, for the $c(d)$ parameter (current investment capital is 1040$).

1. -0.731 is negative, resulting in a SELL indication;
2. Go short in *Apple* shares for the value of 1040$;
3. The price of the shares declines by 6% at the end of the day.
4. Sell the shares of *Apple* resulting in a profit of 62.4$. The current investment capital is 1102.4$.

[2] $http://www.1stock1.com/1stock1\_142.htm$

If we were to step out of the market at this moment, the total investment return after 2 transactions would be of 10.24%. Following this example, transactions are carried out for all the companies within the validation set. At the end of the time frame the total investment capital is summed over all companies and the percentual return is calculated. Dividend dates, borrowing fees and stop-loss orders (an order to buy or sell a stock once the price of the stock reaches a specified price, known as the stop price, to manage your risks) are out of the scope in the current calculations. Furthermore, for the $d + 2$ and $d + 4$ parameters, once an investment has been placed for a company, no new investments can be placed for the same company as long as the current investment period is active. The final excess return is the difference between the percentual return for the entire portfolio (let's assume 10.24% in this case) and NASDAQ performance (-1.80%), resulting in 12.04%.

We also created a random result set per parameter, based on randomly (positive and negative) generated numbers, which should indicated the percentual stock price changes. These results act as a baseline for our precision and investment return measures. Furthermore we discuss the computation time difference between using WSD and NOWSD.

6.2 Results

As already described, the news articles for validation are from the same pool of data, but from a different time frame. In total 17589 articles were scanned (366 articles average per company, with a standard deviation of 598). The precisions for validation method 1 can be found in Table 6.1 for company specific event weights and Table 6.2 for average event weights, while the second validation method results can be found in Table 6.3 for company specific event weights and Table 6.4 for average event weights.

Method	WSD	NOWSD	Baseline
$c(d)$	0.506	0.502	0.491
$c(d+2)$	0.500	0.500	0.486
$c(d+4)$	0.493	0.505	0.494
$i(d)$	0.502	0.501	0.496
$i(d+2)$	0.503	0.501	0.499
$i(d+4)$	**0.512**	0.508	0.505

Table 6.1: StockWatcher 2.0 - BUY/SELL precision for company specific event weights.

Method	WSD	NOWSD	Baseline
$c(d)$	0.509	0.509	0.491
$c(d+2)$	**0.533**	0.530	0.486
$c(d+4)$	0.497	0.503	0.494
$i(d)$	0.493	0.495	0.496
$i(d+2)$	0.502	0.510	0.499
$i(d+4)$	0.484	0.492	0.505

Table 6.2: StockWatcher 2.0 - BUY/SELL precision for average event weights.

Method	WSD	NOWSD	Baseline
c(d)	5.13	2.95	0.92
c(d+2)	0.18	-3.09	-1.65
c(d+4)	-1.14	-0.74	0.77
i(d)	3.10	4.82	1.77
i(d+2)	4.17	3.18	0.63
i(d+4)	**5.63**	3.23	1.89

Table 6.3: StockWatcher 2.0 - Excess returns for company specific event weights, expressed in percentages.

Method	WSD	NOWSD	Baseline
c(d)	5.91	5.22	0.92
c(d+2)	**6.42**	6.31	-1.65
c(d+4)	4.03	3.70	0.77
i(d)	1.81	-0.77	1.77
i(d+2)	0.55	4.89	0.63
i(d+4)	-0.23	3.69	1.89

Table 6.4: StockWatcher 2.0 - Excess returns for average event weights, expressed in percentages.

6.3 Computation time

An important performance measure for the application in a real-time deployment is the computation time. For this purpose we tested how long it takes for the system to rate an article on an average desktop computer[3]. In Table 6.5 we can observe the average computation time per article in seconds (measurement was done on the entire validation set, consisting of 17589 articles).

WSD	1.21
NOWSD	1.19

Table 6.5: StockWatcher 2.0 - Average computation time per article in seconds

As we can observe, the average time to process an article is highest for WSD, and lowest for NOWSD, which is natural due to the additional computation time performed by WSD.

[3]Windows 7 64bit, Intel Core2Duo E8500 @3.17GHz, 4GB RAM

6.4 Discussion

With this project we aimed to create an application which can predict the stock price of listed companies by the interpretation of human natural language. To compare the results of the WSD, NOWSD and random baseline approaches, we also address the statistical significance of these results. For the first validation effort we make use of the Fisher's exact test[4] (a test which calculates if the differences between two binary observation sets is the result of chance, or were due to other factors). The null hypothesis is defined by: there is no significant difference between the result sets. For the second validation we make use of the Wilcoxon paired signed rank test[5] (a statistical hypothesis test used when comparing two related samples). The null hypothesis is defined by: the median difference between pairs of observations is zero. If we can reject the null hypothesis, the compared results are significantly different. In both cases we want to determine whether the degree of the observed difference (for example between WSD and NOWSD) reflects anything more than some lucky guessing.

The results for the first validation effort (predict precision of BUY / SELL signals) show that for the company specific event weights ($event^c(w)$), the parameter $i(d+4)$ with WSD performs the best with a precision of 0.512 (Fisher test p-value of 0.39 when compared to NOWSD and 0.27 when compared to the random baseline, indicating that there is no statistical significance in the difference between WSD and NOWSD / random baseline results). Furthermore almost all precision results perform better with WSD, when compared to NOWSD results or the random baseline (with the exception of $c(d+4)$). When we look at the average event weights ($event^a(w)$), the parameter $c(d+2)$ gives the best result with a precision of 0.533 (Fisher test p-value of 0.40 when compared to NOWSD and 0.00001 when compared the random baseline).

The interpretation of these results is that having a higher number of events to match to news articles improves the overall precision, even if these events

[4] $http://graphpad.com/quickcalcs/contingency1.cfm$

[5] $http://www.wessa.net/rwasp_Reddy-MooresWilcoxonMann-WitneyTest.wasp$

were trained for different companies. Furthermore, the use of word sense disambiguation provides the best results for both weight methods ($event^c(w)$ and $event^a(w)$). We can conclude that having a higher number of events increase the precision, as long as the sense of the event is relevant.

In the second validation effort (calculating the excess investment returns) we can observe that for $event^c(w)$ the parameter $i(d+4)$ with WSD performs the best, with a significant excess return of 5.63% (Wilcoxon p-value of 0.000 when compared to both NOWSD and random baseline). When we look at $event^a(w)$, the parameter $c(d+2)$ provides the best result, with a significant excess return of 6.42% (Wilcoxon p-value of 0.000 when compared to both NOWSD and random baseline).

Again we can imply that events trained for different companies carry their weight impact to other related companies as well (hence $event^a(w)$ outperforming $event^c(w)$). The results also indicate that the news item impact on the stock price is noticeable after a delay.

To summarise the conclusions of the two validation tests:

- there is a noticeable news item impact on the stock price, however after a delay;
- events which were trained to predict the stock price for company A can carry their prediction impact to the stock price for company B;
- making use of WSD increases the performance overall, for precision and return results;

Chapter 7

Conclusion

In this thesis we laid out the stock price prediction framework for our application, StockWatcher 2.0. By combining different Natural Language Processing (NLP) tools, we created a prediction system for stock prices based on text articles. As discussed in the first chapters, the use of NLP has increased over the years, as new tools and information sources became available, for example WordNet. However, word sense disambiguation (WSD) is fairly new, and has not been employed on a large scale for news analytics.

To be able to interpret the core subject of an article, one must know the intended meaning of the words. For humans this task is trivial, but for machines this has been proven to be an obstacle. For this reason we implemented a word sense disambiguation module in our application, in order to improve the machine understanding of an article. We have chosen for the graph-based degree WSD algorithm according to the Sihna et al. This measure has the better precision compared to other techniques.

One of the more important steps in our research was to determine the impact of an event on the stock price for a company. A training set from Dow Jones Newswires articles was composed for the time period of 1 January 2010 - 31 December 2010, for the top 48 non-financial companies rated at the NASDAQ. The training facilitated the calculation of the event weight (each event identified within an article received a weight based on the percentual difference between the opening price and closing price). With each event

weighted, the system is able to calculate new stock prices based on new articles for the same company. We also created an average weighted training set from all company specific events, in order to be able to predict stock prices for companies which are not part of the training set.

We then proceeded to measure the application precision on new articles. A test set was created, with the same parameters, for the time period of 1 January 2011 - 31 December 2011. The validation was done in two steps: in the first step the system calculates the precision by predicting the BUY / SELL signal for a day given a company, in the second step we calculate the investment excess return based on the actual returns generated by our BUY / SELL indicators (by following a long / short stock selling strategy based on the indicator) compared to the NASDAQ index returns for the same time period. The first method produced the best precision at 0.533% with average weighted events with word sense disambiguation enabled. The second method produced the best significant investment excess return of 6.42% in favour of average event weights with word sense disambiguation. In both validation steps the results were significantly better than a random predicted result set (based on randomly positive and negative generated numbers, which should indicated the percentual stock price changes).

Our research question was: *"Is it possible to predict stock prices for listed companies by analyzing events in news items?"*. Our tests concluded that our application, making use of word sense disambiguation based on graph degree similarity, produces higher precision and investment return results when compared to a random baseline or no sense disambiguation.

7.1 Future work

The article rating system could be improved in two ways. The current application framework could be changed to relate event weights to human sentiment (in addition of the stock prices). By conducting tests with human annotated articles (what is the human sentiment towards the article) the prediction model can make use of a hybrid model (event weights calculated

by stock price and human sentiment) to predict the stock price.

At the same time, the article rating could be reprogrammed to run on multiple threads, as multithreaded processors are gradually becoming the standard. This would improve the computation time significantly in a real-time environment. Furthermore new database techniques, such as and MapReduce[1] could improve the processing time for our large data sets.

[1]Programming model to process large data sets, to be used on a distributed computing clusters. Available as Apache Hadoop: *http : //hadoop.apache.org/*

Appendix A

Events

stock rise	share up	dividend increase
stock ascent	share grow	dividend boost
stock climb	share ascend	dividend raise
stock increase	share heighten	dividend climb
stock lift	share jump	income gain
stock boost	share propel	income growth
stock advance	launch	income rise
stock up	release	income jump
stock grow	bring out	income increase
stock ascend	publish	income boost
stock heighten	issue	income raise
stock jump	new application	income climb
stock propel	new product	earning rise
share rise	new technology	earning gain
share ascent	new service	earning boost
share climb	new software	earning growth
share increase	dividend gain	earning jump
share lift	dividend growth	earning increase
share boost	dividend rise	earning climb
share advance	dividend jump	revenue rise

revenue growth	market increase	under expectation
revenue earn	business deal	below expected value
revenue gain	sign deal	under expected value
revenue boost	sign contract	forecast loss
revenue jump	sign agreement	forecast losings
revenue increase	get contract	forecast losses
revenue raise	new contract	expect loss
revenue climb	partnership	expect losings
profit raise	expand	expect losses
profit rise	alliance	anticipate loss
profit gain	close deal	anticipate losings
profit growth	get deal	anticipate losses
turn profit	new deal	widespread loss
profit jump	finalize deal	bad result
profit increase	attain contract	company loss
profit boost	acquire contract	company losses
profit climb	gain contract	company losings
good result	gain deal	lost contract
solidify position	acquire deal	lose contract
strengthen position	exceed expectation	miss contract
market portion up	surpass expectation	reject deal
market portion grow	exceed expected value	reject bid
market part up	surpass expected value	reject contract
market part grow	forecast earning	turn down deal
market percentage up	forecast profit	turn down bid
market percentage grow	expect earning	turn down contract
market portion rise	expect profit	refuse deal
market part rise	anticipate earning	refuse bid
market percentage rise	anticipate profit	refuse contract
market gain	below expectation	block deal

postpone deal	income decrease	profit loss
delay deal	income sink	profit down
stock drop	income fell	profit decrease
stock fall	income drop	profit sink
stock sink	income decline	profit fell
stock fell	income dip	profit bad
stock decrease	income trim	profit drop
stock down	income reduce	profit decline
stock descend	income slice	profit dip
stock decline	earning loss	profit trim
stock dip	earning down	profit reduce
stock slice	earning decrease	lawsuit
share drop	earning sink	case
share fall	earning fell	suit
share sink	earning drop	anti trust
share fell	earning decline	monopoly
share decrease	earning dip	lose case
share down	earning trim	lose suit
share descend	earning slice	copyright violation
share decline	revenue loss	copyright infringement
share dip	revenue down	trademark violation
share slice	revenue decrease	security problem
loss dividend	revenue sink	security flaw
low earning	revenue fell	software flaw
low income	revenue bad	software defect
low revenue	revenue drop	product defect
low profit	revenue decline	product flaw
dividend decrease	revenue trim	global crisis
income loss	revenue reduce	recession
income down	revenue dip	bad economy

Appendix B

Training results

		WSD		NOWSD	
Company	Articles	Event occurences	Dates	Event occurences	Dates
activision	96	108	30	151	38
adobe	198	280	51	383	59
adp	50	84	21	100	21
alexion	21	15	7	31	10
amazon	411	337	107	502	127
amgen	186	104	48	190	62
apple	2319	1823	230	3010	236
applied	71	92	26	126	27
arts	105	103	31	148	43
baidu	129	198	37	247	43
biogen	194	160	49	288	65
bmc	33	62	12	68	13
broadcom	124	135	41	200	52
celgene	89	110	22	157	28
cerner	27	28	10	39	12

		WSD		NOWSD	
Company	Articles	Event occurences	Dates	Event occurences	Dates
cisco	351	324	77	436	102
cme	240	220	52	328	61
cognizant	31	70	12	86	13
comcast	316	328	89	475	105
costco	142	81	22	127	31
dell	586	631	110	868	129
directv	111	126	44	175	54
dish	102	126	38	193	48
ebay	304	384	74	578	96
express	81	113	21	135	28
gilead	76	89	16	110	21
google	2137	2002	235	3199	246
intel	874	893	137	1220	166
intuit	56	86	14	103	22
intuitive	44	44	10	66	16
kraft	829	562	75	783	93
liberty	104	148	28	212	38
microsoft	1079	1193	197	1762	213
news	1251	1159	163	1655	186
oracle	516	760	97	1025	112

		WSD		NOWSD	
Company	Articles	Event occurences	Dates	Event occurences	Dates
paccar	43	35	9	52	9
powershares	9	2	2	9	5
priceline	89	132	24	158	30
qualcomm	340	380	71	479	82
randgold	73	30	16	40	19
starbucks	258	215	58	316	72
symantec	114	210	27	244	30
texas	108	79	35	102	41
verizon	727	598	153	944	176
viacom	209	310	60	419	72
vodafone	820	541	127	785	152
whole	65	56	14	69	16
yahoo	547	701	143	936	155
TOTAL	**16685**	**16267**	**2972**	**23729**	**3475**
Average	**348**	**339**	**62**	**494**	**72**
Std Dev	**495**	**438**	**59**	**685**	**64**

Appendix C

Validation results

		Average based events			
		WSD		NOWSD	
Company	Articles	Event occurences	Dates	Event occurences	Dates
activision	86	125	33	182	39
adobe	125	206	31	284	37
adp	58	125	23	147	27
alexion	13	28	6	36	7
amazon	651	587	139	904	169
amgen	145	158	37	242	52
apple	2930	2673	231	4483	244
applied	70	88	12	111	16
arts	120	84	25	129	40
baidu	184	368	53	484	59
biogen	127	161	32	217	39
bmc	21	59	9	61	9
broadcom	107	171	34	210	38
celgene	59	88	17	136	28
cerner	10	27	6	27	6

		Average event weights			
		WSD		NOWSD	
Company	Articles	Event occurences	Dates	Event occurences	Dates
cisco	341	416	78	517	96
cme	565	499	130	764	155
cognizant	27	58	14	69	15
comcast	296	232	62	301	82
costco	139	134	29	153	36
dell	341	324	72	474	90
directv	76	94	31	118	37
dish	142	147	36	213	47
ebay	390	401	77	534	85
express	162	275	44	337	49
gilead	97	101	25	144	34
google	2700	2470	236	3760	243
intel	643	710	122	977	141
intuit	57	88	20	103	24
intuitive	20	44	8	48	9
kraft	331	356	66	439	77
liberty	105	156	29	186	40
microsoft	1245	1295	195	1842	213
news	1381	1258	141	2328	166
oracle	467	568	103	786	120

		Average based events			
		WSD		NOWSD	
Company	Articles	Event occurences	Dates	Event occurences	Dates
paccar	30	49	9	53	10
powershares	9	1	1	1	1
priceline	48	122	16	128	17
qualcomm	315	445	67	541	78
randgold	77	39	15	45	15
starbucks	355	335	72	524	91
symantec	53	74	17	87	19
texas	175	129	54	163	57
verizon	639	571	126	841	148
viacom	168	202	46	312	67
vodafone	758	589	115	811	145
whole	42	69	14	80	18
yahoo	689	744	99	990	103
TOTAL	**17589**	**17943**	**2857**	**26322**	**3338**
Average	**366**	**374**	**60**	**548**	**70**
Std Dev	**598**	**545**	**58**	**882**	**63**

		Company based events			
		WSD		NOWSD	
Company	Articles	Event occurences	Dates	Event occurences	Dates
activision	86	12	5	169	39
adobe	125	72	5	283	37
adp	58	9	3	128	26
alexion	13	0	0	29	6
amazon	651	249	44	897	167
amgen	145	66	6	241	52
apple	2930	1994	167	4475	244
applied	70	8	1	110	16
arts	120	24	4	128	40
baidu	184	80	9	473	59
biogen	127	54	10	216	39
bmc	21	12	2	49	9
broadcom	107	38	9	206	38
celgene	59	0	0	129	26
cerner	10	13	3	25	6

		Company based events			
		WSD		NOWSD	
Company	Articles	Event occurences	Dates	Event occurences	Dates
cisco	341	166	27	513	96
cme	565	59	20	735	153
cognizant	27	2	1	64	15
comcast	296	62	19	297	82
costco	139	19	3	153	36
dell	341	172	34	474	90
directv	76	5	2	117	37
dish	142	25	4	209	47
ebay	390	100	22	532	85
express	162	21	5	326	49
gilead	97	36	9	142	34
google	2700	1995	174	3759	243
intel	643	408	58	973	141
intuit	57	13	2	100	23
intuitive	20	9	2	46	9
kraft	331	123	16	437	77
liberty	105	38	4	180	40
microsoft	1245	1005	128	1842	213
news	1381	538	74	2327	166
oracle	467	237	38	777	119

		Company based events			
		WSD		NOWSD	
Company	Articles	Event occurences	Dates	Event occurences	Dates
paccar	30	0	0	49	10
powershares	9	0	0	1	1
priceline	48	43	4	123	15
qualcomm	315	259	19	538	77
randgold	77	11	2	43	15
starbucks	355	92	18	517	91
symantec	53	5	3	85	19
texas	175	14	8	162	57
verizon	639	327	68	841	148
viacom	168	63	9	309	67
vodafone	758	294	46	804	144
whole	42	0	0	79	17
yahoo	689	404	45	979	102
TOTAL	**17589**	**9176**	**1132**	**26091**	**3322**
Average	**366**	**191**	**24**	**544**	**69**
Std Dev	**598**	**421**	**40**	**882**	**63**

Bibliography

K. Ahmad, P. de Oliveira, P. Manomaisupat, M. Casey, and T. Taskaya. Description of events: An analysis of keywords and indexical names. In *Proceedings of the Third International Conference on Language Resources and Evaluation: Workshop on Event Modelling for Multilingual Document Linking*, pages 29–35. European Language Ressources Association, 2002.

Y. Bar-Hillel. *Language and Information.* Addison-Wesley, 1964.

F. Benamara, C. Cesarano, A. Picariello, D. Reforgiato, and V. Subrahmanian. Sentiment Analysis: Adjectives and Adverbs are better than Adjectives Alone. In *IADIS Applied Computing*, pages 203–206. Association for Computational Linguistics, 2007.

T. Brants. Tnt: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pages 224–231. Morgan Kaufmann Publishers Inc., 2000.

R. A. Brealey and S. C. Myers. *Principles of Corporate Finance with Cdrom*, pages 344 – 375. McGraw-Hill Higher Education, 2000.

E. Brill. Automatic grammar induction and parsing free text: a transformation-based approach. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 259–265. Association for Computational Linguistics, 1993.

S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Network ISDN System*, 30(1-7):107–117, 1998.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. Word-sense disambiguation using statistical methods. In *Proceedings of the 29th Annual Meeting of the ACL*. Association for Computational Linguistics, 1991.

A. Budanitsky and G. Hirst. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources at the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 29–34. Carnegie Mellon University, 2001.

C. Cesarano, B. Dorr, A. Picariello, D. Reforgiato, A. Sagoff, and V.S. Subrahmanian. Oasys: An opinion analysis system. In *Proceedings the Spring Symposia on Computational Approaches to Analyzing Weblogs*, pages 21–26. Association for the Advancement of Artificial Intelligence, 2006.

Y. Chan, A. C. W. Chui, and C. C. Y. Kwok. The impact of salient political and economic news on the trading activity. *Pacific-Basin Finance Journal*, 9(3):195–217, 2001.

W. Cho. *Knowledge discovery from distributed and textual data*. PhD thesis, Hong Kong University of Science and Technology, 1999.

G. G. Chowdhury. Natural language processing. *Annual Review of Information Science and Technology*, 37(1):51–89, 2003.

J. Christian and E. Tsoukermann. *Natural Language Processing for term variant extraction: synergy between morphology, lexicon, and syntax.*, pages 25–74. Dordrecht: Kluwer Academic Publishers, 1999.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, 2001.

J. Cowie, J. Guthrie, and L. Guthrie. Lexical disambiguation using simulated annealing. In *Proceedings of the 14th conference on Computational linguistics*, pages 359–365. Association for Computational Linguistics, 1992.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

J. Didion. The Java WordNet Library, 2007. `http://jwordnet.sourceforge.net/`.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*, pages 44–64. John Wiley and Sons, 1974.

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

T. Fawcett and F. Provost. Activity monitoring: noticing interesting changes in behavior. In *Proceedings of the fifth ACM international conference on Knowledge discovery and data mining*, pages 53–62. Association for Computational Linguistics, 1999.

C. Fellbaum, editor. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, 1998.

W. B. Frakes. Stemming algorithms. In *Information retrieval: data structures and algorithms*, pages 131–160. Prentice-Hall, Inc., 1992.

Z. Ghahramani. An introduction to hidden markov models and bayesian networks. In *Hidden Markov models: applications in computer vision*, pages 9–42. World Scientific Publishing Co., Inc., 2002.

R. Grishman, C. Macleod, and A. Meyers. Comlex syntax: building a computational lexicon. In *Proceedings of the 15th conference on Computational linguistics*, pages 268–272. Association for Computational Linguistics, 1994.

M. A. Hearst. Noun homograph disambiguation using local context in large corpora. In *Proceedings of the 7th Annual Conference of the University of Waterloo Centre for the New Oxford English Dictionary*, pages 1–22, 1991.

M. A. Hearst. Text data mining - issues, techniques and the relationship to information access. In *UW/MS Workshop on Data Mining*, pages 623–645, 1997.

D. Hirshleifer and T. G. Shumway. Good day sunshine: Stock returns and the weather. Technical report, EconWPA, 2004.

G. Hirst and D. St. Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In *WordNet: An Electronic Lexical Database*, pages 305–332. MIT Press, 1998.

C. H. Hommes. Modeling the stylized facts in finance through simple nonlinear adaptive systems. *Proceedings of the National Academy of Sciences*, 99(90003):7221–7228, 2002.

W. Hullen. *A History of Roget's Thesaurus*. Oxford University Press, New York, 2003.

W. J. Hutchins and H. L. Somers. *An Introduction to Machine Translation*, pages 81 – 97. New York: Academic Press, 1992.

N. Ide and J. Véronis. Introduction to the special issue on word sense disambiguation: the state of the art. *Computational Linguistics*, 24(1): 2–40, 1998.

J. J. Jiang and D. W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *International Conference Research on Computational Linguistics*, pages 19–33, 1997.

D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, 2000.

A. Kilgarriff and J. Rosenzweig. Framework and results for English SENSEVAL. *Computers and the Humanities*, 34(1-2), 2000.

P. Klibanoff, O. Lamont, and T. A. Wizman. Investor reaction to salient news in closed-end country funds. *The Journal of Finance*, 53(2):673–699, 1998.

J. J. Korczak, P.r Lipinski, and P. Roger. Evolution strategy in portfolio optimization. In *Selected Papers from the 5th European Conference on Artificial Evolution*, pages 156–167. Springer-Verlag, 2002.

R. Krovetz and W. B. Croft. Lexical ambiguity and information retrieval. *ACM Transactions on Information Systems*, 10(2):115–141, 1992.

S. Landes, C. Leacock, and R. I. Tengi. Building semantic concordances. In *WordNet: An Electronic Lexical Database*, pages 199–216. MIT Press, 1998.

V. Lavrenko, M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen, and J. Allan. Language models for financial news recommendation. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 389–396. Association for Computational Linguistics, 2000.

C. Leacock and M. Chodorow. Combining local context and WordNet similarity for word sense identification. In *WordNet: An Electronic Lexical Database*, pages 265–283. MIT Press, 1998.

K. Lee, H. Hon, M. Hwang, and X. Huang. Speech recognition using hidden markov models: a cmu perspective. *Speech Commun.*, 9(5-6):497–508, 1990.

A. Lenci, N. Bel, F. Busa, N. Calzolari, E. Gola, M. Monachini, A. Ogonowski, I. Peters, W. Peters, N. Ruimy, M. Villegas, and A. Zampolli. Simple: A general framework for the development of multilingual lexicons. *International Journal of Lexicography*, 13(4):249–263, 2000.

M. Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of*

the 5th annual international conference on Systems documentation, pages 24–26. Association for Computational Linguistics, 1986.

D. Lin. Using syntactic dependency as local context to resolve word sense ambiguity. In *Proceedings of the 35th annual meeting on Association for Computational Linguistics*, pages 64–71. Association for Computational Linguistics, 1997.

A. C. Lobeck. *Discovering grammar : an introduction to English sentence structure*. Oxford University Press, 2000.

B. G. Malkiel. *A Random Walk Down Wall Street*. Norton, New York, 1990.

M. Marcus, G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. The penn treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology*, pages 114–119. Association for Computational Linguistics, 1994.

A. Micu, L. Mast, V. Milea, F. Frasincar, and U. Kaymak. Financial news analysis using a semantic web approach. In *Semantic Knowledge Management: an Ontology-based Framework*, pages 311–328. Idea Group, 2008.

M. Mieskes and M. Strube. Part-of-speech tagging of transcribed speech. In *Proceedings of the 5th Conference on Language Resources and Evaluation*. European Language Resources Association, 2006.

R. Mihalcea. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 411–418. Association for Computational Linguistics, 2005.

R. Mihalcea and D. I. Moldovan. A highly accurate bootstrapping algorithm for word sense disambiguation. *International Journal on Artificial Intelligence Tools*, 10(1-2):5–21, 2001.

G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to wordnet: An on-line lexical database*. *Int J Lexicography*, 3 (4):235–244, 1990.

M. L. Mitchell and J. H. Mulherin. The impact of public information on the stock market. *Journal of Finance*, 49(3):923–50, 1994.

R. Navigli and M. Lapata. Graph connectivity measures for unsupervised word sense disambiguation. In *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*, pages 1683–1688. Morgan Kaufmann Publishers Inc., 2007.

R. Navigli and M. Lapata. An experimental study of graph connectivity for unsupervised word sense disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:678–692, 2010.

A. Ng and A. Wai-Chee Fu. Mining frequent episodes for relating financial events and stock trends. In *Advances in Knowledge Discovery and Data Mining*, pages 569–569, 2003.

J. M. Patell and M. A. Wolfson. The intraday speed of adjustment of stock prices to earnings and dividend announcements. *Journal of Financial Economics*, 13(2):223–252, 1984.

T. Pedersen and R. Bruce. Distinguishing word senses in untagged text. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 197–207. Association for Computational Linguistics, 1997.

T. Pedersen, S. Banerjee, and S. Patwardhan. Maximizing Semantic Relatedness to Perform Word Sense Disambiguation. Research Report UMSI 2005/25, University of Minnesota Supercomputing Institute, 2005.

D. Peramunetilleke and R. K. Wong. Currency exchange rate forecasting from news headlines. *Australian Computer Science Communications*, 24 (2):131–139, 2002.

R. Quirk, S. Greenbaum, G. Leech, and J. Svartvik. *A comprehensive grammar of the english language*. Longman, 1985.

R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, 19(1):17–30, 1989.

P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 448–453. Morgan Kaufmann, 1995.

R. L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

H. Rodríguez, S. Climent, P. Vossen, L. Bloksma, W. Peters, A. Alonge, F. Bertagna, and A. Roventini. The top-down strategy for building eurowordnet: vocabulary coverage, base concepts and top ontology. In *EuroWordNet: a multilingual database with lexical semantic networks*, pages 45–80. Kluwer Academic Publishers, 1998.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Neurocomputing: foundations of research*, pages 673–695. MIT Press, 1988.

M. Sanderson. Word sense disambiguation and information retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 142–151. Springer-Verlag New York, Inc., 1994.

L. Schubert and M. Tong. Extracting and evaluating general world knowledge from the brown corpus. In *Proceedings of the HLT-NAACL 2003 workshop on Text meaning*, pages 7–13. Association for Computational Linguistics, 2003.

Y. Seo, J. A. Giampapa, and K. P. Sycara. Text classification for intelligent agent portfolio management. In *International Joint Conference on Autonomous Agents and Multi-Agent Systems*. Springer, 2002.

R. Sheldon. *A First Course In Probability, 6/E*. Pearson Education, 2002. ISBN 9788177583618.

R. Sinha and R. Mihalcea. Unsupervised graph-basedword sense disambiguation using measures of word semantic similarity. In *ICSC '07: Proceedings of the International Conference on Semantic Computing*, pages 363–369. IEEE Computer Society, 2007.

K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 173–180. Association for Computational Linguistics, 2003.

C. J. van Rijsbergen, S. E. Robertson, and M. F. Porter. New models in probabilistic information retrieval. *British Library Research and Development Report, no. 5587*, 1980.

A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.

D. E. Walker. Knowledge resource tools for accessing large text files. In *Machine Translation: Theoretical and Methodological Issues. Sergei Nirenberg*, pages 247–261. University Press, 1987.

Z. Wu and M. Palmer. Verb semantics and lexical selection. In *32nd. Annual Meeting of the Association for Computational Linguistics*, pages 133 –138. Association for Computational Linguistics, 1994.

J. Xu and W. B. Croft. Corpus-based stemming using cooccurrence of word variants. *ACM Transactions on Information Systems*, 16(1):61–81, 1998.

D. Yarowsky. Word-sense disambiguation using statistical models of roget's categories trained on large corpora. In *Proceedings of the 14th conference on Computational linguistics*, pages 454–460. Association for Computational Linguistics, 1992.

D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for*

Computational Linguistics, pages 189–196. Association for Computational Linguistics, 1995.

C. Yue-Cheong. Political risk and stock price volatility: The case of hong kong. *Pacific-Basin Finance Journal*, 4:259–275(17), 1996.