Master of Science Thesis

Augmented trading

From news articles to stock price predictions using syntactic analysis

Arthur Hugo van Bunningen
January 2004

Parlevink Language Engineering Group
Department of Computer Science
University of Twente
Enschede
The Netherlands

Graduation Committee:
Anton Nijholt
Mannes Poel
Martijn van Otterlo

Abstract

This thesis tries to answer the question how to predict the reaction of the stock market to news articles using the latest suitable developments in Natural Language Processing. This is done using text classification where a new article is matched to a category of articles which have a certain influence on the stock price. The thesis first discusses why analysis of news articles is a feasible approach to predicting the stock market and why analysis of past prices should not be build upon. From related work in this domain two main design choices are extracted; what to take as features for news articles and how to couple them with the changes in stock price. This thesis then suggests which different features are possible to extract from articles resulting in a template for features which can deal with negation, favorability, abstracts from companies and uses domain knowledge and synonyms for generalization. To couple the features to changes in stock price a survey is given of several text classification techniques from which it is concluded that Support Vector Machines are very suitable for the domain of stock prices and extensive features. The system has been tested with a unique data set of news articles for which results are reported that are significantly better than random. The results improve even more when only headlines of news articles are taken into account. Because the system is only tested with closing prices it cannot concluded that it will work in practice but this can be easily tested if stock prices during the days are available. The main suggestions for feature work are to test the system with this data and to improve the filling of the template so it can also be used in other areas of favorability analysis or maybe even to extract interesting information out of texts.

Preface

> Your project is too boring, van Bunningen. Hence, I will not mention it.
>
> Anton Nijholt

> This is a three years project.
>
> Jan Kuper

Three years of work on a boring project seems useless. Hence, it is not only time to finish the project but also to show that information extraction is not so boring at all. To show the magic of informatics and artificial intelligence which interested me sixteen years ago, and still fascinate me; that by having the right formulas and algorithms one can find very interesting results. Therefore this thesis is dedicated to the person who wrote the book on BASIC programming which contained my first words:

```
10 PRINT "HA HA HA"
20 FOR I=1 TO 1000: NEXT I
30 PRINT "TSJOE"
```

Of course I could not have done this work all alone, therefore I will gratefully thank the following people:

First my former roommate and now ft. Dr. Rutger Rienks, for the valuable discussions and making time at TKI not boring at all. And of course Dennis Reidsma with whom contacts became more informal each month and who helped me the last weeks, making time to give helpful reviews within an impossible time frame.

Priya Raghubir, for sending her article by letter, Victor Lavrenko, for his data-set.

Connexor Oy for letting me use Connexor FDG for this project.

Michael Bergmeijer of Dow Jones Newswires for the interview and access to the data set.

Martijn van Drunen, Ridder Daan Evers and Wilbert Menne at Saen options for the extensive information and tour.

My committee: Anton Nijholt, Martijn van Otterlo and Mannes Poel for their feedback and support.

Jaap van der Hart of Robeco (Financial aspects) Yeb Havinga (OpenCyc), Arun Bagchi (Financial aspects and contacts), Berend Roorda (Financial aspects), Jan Kuper (Template merging), Danny Lie (language technology), Paul van der Vet (ontologies and language technology), Roeland Ordelman (language

technology) and Marieke Smit (the TCW side of the problem) for their valuable discussions on specific subjects.

But one should never forget the people who had to deal with me during those months of exhaustion and cancelation of appointments, especially Joost, Jorien and Maarten, my housemates Martin and Jesper, my parents and my brother Steven. Thank you. To you I dedicate the first beer I will drink after I finish this.

Arthur van Bunningen

Contents

Chapter 1

Introduction to the problem

> One may be moderately optimistic, too, that IE may be the technology vehicle with which old AI goals of adaptive, tuned, lexicons and knowledge bases can pursued, IE may also be the only technique that will ever provide a substantial and consistent knowledge base from texts, as CYC has failed to do over twenty years.
>
> [Wilks, 2000]

1.1 Considerations

The continuous availability of more news articles in digital form, the latest developments in Natural Language Processing and the availability of faster computers lead to the question how to extract more information out of news articles.

In the field of trading, most analysis tools of the stock market still focus on statistical analysis of past price developments. This despite the assumption that the course of a stock price can be predicted much better by looking at appeared news articles. Some experts even say that this is the only way to predict the stock price. Because of the great amount of appearing articles (sometimes more than sixteen articles per company per day), the question comes up which articles will probably have a large influence and need to be looked at, and how to do this using Natural Language Processing.

There exist other fields in which Natural Language Processing could lead to the extraction of more information contained in news articles, yet the advantages of having an objective evaluation and an easy to explain, immediately available practical result made us choose the stock market as our domain. We explicitly do not want the techniques we develop to be limited to the stock market, for example by having our system completely rely on key phrases given by experts.

Eventually research in this area could lead to a system with the following capabilities:

- suggesting news articles which will probably have a great impact on the stock market,
- classifying news articles as “good” or “bad”,
- more understanding of the reactions of humans to news articles,

- extracting more information contained in news articles and
- statements about the effectiveness of different learning strategies in this domain.

1.2 The problem statement

Considering the assumption that news articles might give much better predictions of the stock market than analysis of past price developments, we want to develop a system which is able to use the latest most suitable developments in Natural Language Processing to model the reaction of the stock market to news articles and predict further reactions, by which we will keep in mind that we do not want the techniques being developed to be too depended on the stock market.

As input we take past news articles and stock prices. From these we build a model in which a correlation is made between certain features found in these articles and changes in stock price. We feed the system with new news articles and hope that the features found in these articles will cause the same reaction as in the past. In this case it will be possible to predict the reaction of the stock market as seen in figure 1.1.

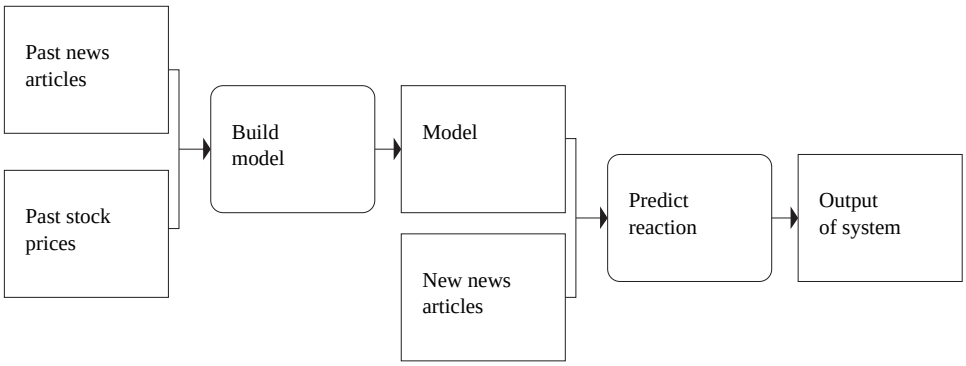

Figure 1.1: The idea of the system.

The reaction of the market to an article can be positive, negative or not relevant. This is why we want to discriminate between three kinds of reactions; *plunge*, *not relevant* and *surge* which respectively stand for a large drop in stock price, a non interesting reaction and a large rise in stock price[1].

We would for example like to predict the reaction on the following article, which appeared on January 13th, 2003:

```
Vodafone wil dochter Libertel geheel inlijven

AMSTERDAM (ANP) - Het Britse telecombedrijf Vodafone
wil een bod uitbrengen op de resterende aandelen van
```

[1]Lavrenko *et al.* discriminated between five reactions; *plunge*, *slight-*, *not relevant*, *slight+* and *surge*. However, the choice for these discriminations was not argued and the advantage of having a special category for slight reactions is not clear.

```
zijn Nederlandse dochterbedrijf Vodafone Libertel,
dat aan de beurs in Amsterdam staat genoteerd.
```

Out of this article we could extract an"inlijf" (take over) feature in which Libertel is taken over. It could be learned from past articles that this feature will most of the time have a positive reaction on stock price of the company that is taken over. The system could, based on this information, suggest that this article will lead to a rise in stock price and recommend it to traders for reading. As we can see in figure 1.2 this article actually led to rice in stock price of 3.74%.

Figure 1.2: A positive change in stock priced caused by a news event.

1.3 Main innovative aspects

The main innovative idea in this thesis is a suggestion of how to incorporate extensive features with both syntactic and domain knowledge into existing text classification procedures, how to implement the automatic extraction of these features and how it would be possible to use them to augment trading possibilities.

Furthermore this thesis looks at the financial background of predicting price changes based on news articles and suggests additions to existing methods. And finally this thesis suggests ways to contribute to ontology construction and describes future directions in the field of text classification and information extraction.

1.4 Outline of the thesis

Before the system described in our problem statement can be constructed, first we must have reason to believe that news articles do have an effect on the stock price. We make this hypothesis now and give arguments why it is viable in chapter 4.

As an introduction to the domain and to place our project in perspective we will first discuss related work. Based on this work we will discuss the design for our system in chapter 3.

This design leads to three main parts:

The stock market describes our domain, how stock prices can be related to news articles and how we should preprocess the articles and prices for use in our system (chapter 4).

Choosing features describes what to chose as features and how to extract these features. (chapter 5 and 6).

Model construction describes how to construct a model to get from these features in news articles to a prediction of the reaction of the stock market (chapter 7).

Results of our system are given in chapter 8, based on which conclusions are drawn in chapter 9. These conclusions are interpreted for future work in chapter 10.

Chapter 2

Related work

2.1 Introduction

As an introduction and to place our research in perspective this chapter introduces other research on predicting the influence of new news articles on the stock price by relating previous news articles to previous stock prices.

2.2 A system predicting sentiment using frequent collocated phrases

The first publication about using automatic classification of news articles to say something about the sentiment around a company was by [Seo *et al.*, 2002]. They classified the training set into five categories:

Good News articles which show good evidences of the company's financial status explicitly e.g. *The shares of ABC Company rose 1/2 or 2 percent on the Nasdaq to $24-15/16.*

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 bad evidences of the company's financial status explicitly e.g. *The shares of ABC fell in early New York trading.*

Because classifying news articles as saying something *good* or *bad* about the performance of a company is a relatively objective task when limited to a certain company (at least not as difficult as classifying them into categories such as

agriculture or health), it is argued that it is viable to classify the articles by hand.

In these articles frequent collocated phrases (FCPs) are identified; these are bags of nearby, but not necessary consecutive, words which are extracted after removing the stop-words. An example could be "shares, rose" or "product, deal, good". Each FCP is a so called *Domain Expert* for a category with a certain weight. If a new news article is seen, all appearing FCPs *vote* with a certain weight for their category and the category with the highest vote is chosen.

Only the 200 most informative FCPs are used to classify between the categories. The selection of these 200 is based on their Information Gain. This means that when a FCP has a higher Information Gain, and therefore is more discriminative between categories, it is considered better[1]. The weights of the FCPs are trained using several methods. They compared their approach to the word-based Naive Bayes (see chapter 7.3 for details on this method) on which they report improvement.

Interesting for our research is that they observe that most financial articles contain news about multiple companies and that therefore most of the time company names or *ticker symbols*[2] will be mentioned explicitly in the sentences. Based on this and the fact that most interesting sentences are the ones in which the company name appears, they only take sentences containing a company name or symbol into account for extracting FCPs.

2.3 Two complete systems using probabilistic rules

In [Cho, 1999] a strategy is described based on probabilistic rules with as variable the time. These are an extension of first-order rules with the ability to handle uncertain or weighted facts. An example of a probabilistic rule is:

```
HSI_UP(T+1)  <--  GOLD_SELL(T), ~BOND_UP(T)
```

In this example the different predicates are weighted, for example *GOLD_SELL(T)* is the weighted occurrence of a certain word sequence over some articles at time T.

As advantage of probabilistic rules Cho mentions the very comprehensible models and although the time of training is exponential to the number of rules to be generated, he argues that four or five rules are sufficient and therefore the training time would be relatively fast. He even argues that these four or five rules will be more accurate than a larger rule set, which would overfit the data. He therefore constraints on this maximum number of rules.

As input for his learning algorithm he takes four hundred sequences of words provided by a domain expert such as "property weak", "dow rebound" and "technology rebound strongly". These are for the last 100 days transformed into weights depending on the closing prices of the stock markets and the news articles which appeared. From these values probabilistic rules are generated.

[1] This selection of features is also called *term space reduction*. Other functions which are useful for this task are described in [Sebastiani, 2002].

[2] Ticker symbol: A symbol which is usually used for representing a companys name briefly in stock trade market and is for each company and market unique.

The mean square error over the training examples is used to measure the accuracy of a rule. This is an appropriate goodness measure for applications when the classification problem is expected to be relatively difficult or no perfect solution exists. For a new specialized rule s and existing rules R this measure is calculated as follows:

$$mse(R \cup s) = \sum_t (up(t) - output_{R \cup s}(t))^2 \tag{2.1}$$

Where $output(t)$ is the output of the rules at time t and $up(t)$ is 1 if the market price is raised at time t and 0 otherwise. For each step in the algorithm the rule with the smallest mean square error is added until the maximum number or rules is reached. More about their rule generating algorithm can be found in [Savasere *et al.*, 1995].

These generated rules applied to news of the day are then used to predict if the market will go up (rises at least 0.5 %), go down (drops at least 0.5%) or remains steady. Tested with 60 stock trading days, Cho reports an accuracy of 45% for the dow Jones, where 46.7% of predictions are slightly wrong and 8.3% are wrong.

[Peramunetilleke and Wong, 2002] describe a same sort of algorithm for the prediction of currency exchange rates. Here the probabilistic rules which are extracted look like:

```
DOLLAR_UP(T)  <--  STOCK_ROSE(T-1), BOND_STRONG(T-1)
```

In this rule, the STOCK_ROSE and BOND_STRONG are indicators of the appearance of word sequences. Other examples of the word sequences mentioned by their experts are “Germany lower interest rate” and “Pound lower”.

What is interesting about this approach is that they only look at news headlines such as:

```
1993-09-24 09:04:18
"DUTCH MONEY MARKET RATES LITTLE CHANGED"
```

In this way the risk of extracting the wrong rules from the article itself is eliminated and the focus is on the most important information. This makes it an interesting option to test in our system as well.

Their best prediction accuracy tested for 60 time frames for the exchange rate between the US Dollar and the German Mark is 51%. The chance this accuracy is gained by random guessing is less than 0.4%.

2.4 A complete system using a Bayesian algorithm

The initial project we focused on and which tries to predict trends in the stock market using language models is called Ænalyst and is described in [Lavrenko *et al.*, 2000]. In this system time-series are redescribed in high-level features called trends. These trends are then aligned to time-stamped news articles and with the help of language models the system learns which news articles lead to which trends. The final goal of Ænalyst is to select from a feed of news articles, the

ones which are likely to be at the beginning of an interesting trend and present them to the user, accompanied by this trend.

We will shortly describe each step in their process with the relevant information for our project.

Finding input data The input of Ænalyst consists of the history of news articles of 127 companies which were collected from biz.yahoo.com, including intraday stock prices of these companies in the same period.

Identifying and labeling trends This is done by using *Piecewise Linear fitting*, after which trends are labeled according to their slope.

Aligning articles and trends To see which news articles are relevant for a specific company, which is explicitly mentioned by Lavrenko *et al.* as an important step, Ænalyst uses the categorization of articles per company made by Yahoo. For each trend a window is used to see which articles are associated with the trend. This is the case if an article has a time stamp several hours or less before the beginning of the trend.

Building language model For this step word appearances are counted and used together with a Bayesian algorithm to result in the trend which most likely will be caused by the news article. Interesting is the use of a linear backoff, to account for new words in news articles which are not already in the language model.

Evaluation In the final test, stocks are being bought if the systems suggests a positive trend and sold short[3] if the system suggests a negative one. The stock is then held for a maximum of one hour or a price difference of 1% in the users' advantage. After this period the stock is sold or bought back. This led to a positive result after which was concluded that language models provide a good framework for aligning news stories with forthcoming trends.

Next to this evaluation scenario Ænalyst also uses standard evaluation methods but in these tests temporal ordering is ignored. Hereby they not only fail to test the predictiveness of the system but more importantly, when we look at our own data set we see that there are always multiple news articles about the same subject (causing the same trend). If we train our model with 90% of the data set, like they do, chances are very high that when the influence of an article in the test set must be predicted, the articles around that time (causing the same trend) are in the training set. Because these articles contain almost the same features, the chance that the result of the test article is successfully "predicted" is high while in fact this is not prediction.

For future work Lavrenko *et al.* suggests to have richer document features. Currently they take words as features, however these features could be augmented with associations among trends or pairs of related words that are significant among many documents. They also suggest relations between objects within a document that could add interesting knowledge.

[3]To sell short: Selling a stock which is not owned in the hope that the price will go down.

2.5 Other related work

In [Fawcett and Provost, 1999] the prediction of the influence of news articles on the stock price is seen as an *Activity Monitoring* task, which involves monitoring a stream of data and issuing alarms that signal positive activity in the stream. For this task they see the prediction of an occurrence of a 10% change in the company's stock price as an activity and the news article on which this prediction is based as alarm. As indicators in the articles they use single words (unigrams) and sequences of two words (bigrams). They report a performance better than random.

Lastly in [Ahmad *et al.*, 2002] words which represent good or bad news, such as *rise*, *bear*, *bull*, *strong* and *vigour* are counted and compared with the course of the FTSE 100[4]. Their conclusion was that that there *might* be a correlation.

2.6 Summary

As general design of systems systems trying to predict the stock price using news articles, we can see two important choices; what to take as features and how to couple them with the changes in stock prices. As suggestions for features in documents we saw words by experts, frequent collocated phrases, unigrams and bigrams. As suggestions for coupling them to changes in stock prices we saw a Bayesian approach using trends and probabilistic rules. Our method of extracting features is given in chapter 5 and 6 and the coupling to changes in stock price is described in chapter 7.

A way in which our system will improve on the work described in this chapter is by taking more extensive features which are not given by experts, in this way the use of our system becomes less domain dependent and we are able to model important sentence structures such as negation and favorability. Another way in which we improve on existing methods using automatic feature extraction (Fawcett and Provost and Lavrenko *et al.*) is by giving extensive results using confusion matrices in chapter 8 and appendix E instead of reporting the result on the stock market, testing in the training period or simply reporting that the system works better than random. In this way we can do a more objective statement about the possibility to predict the influence of news articles on the stock price using automatically extracted features.

Finally we made in this chapter the suggestion, based on the work by Peramunetilleke and Wong, to only look at the headlines of news articles and hereby focussing on the most important aspect of the article.

[4]The Financial Times/Stock Exchange 100 index, an index of top companies.

Chapter 3

Approach

In this chapter we will globally discuss how our system is build up on the basis of figure 3.1. The different parts are explained in more detail in the rest of our thesis.

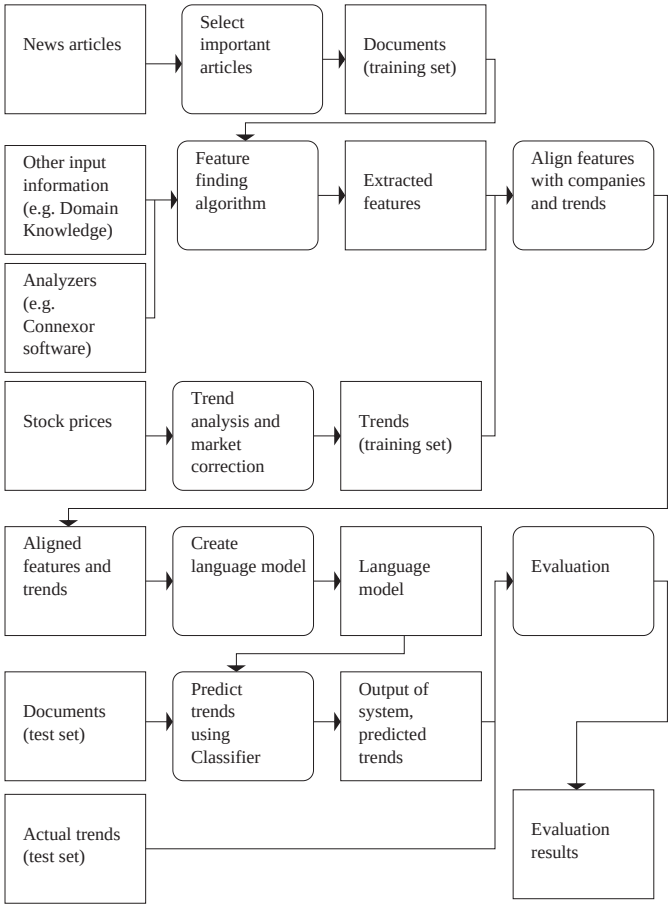

Figure 3.1: The whole system in sequence.

First we will, based on our observations of the market which we will discuss in chapter 4, extract news articles from our news source [Dow, 2003]. This can be seen in figure 3.1 as *select important articles*. These documents are then stored in a database.

To say something about our documents we have to extract certain features from them using a *feature finding algorithm*. The basic features of documents are words or unigrams. We however want to take a more linguistic approach to our problem and will, after discussing word based extensions in chapter 5, discuss a more linguistic template for features in chapter 6.

To be able to say something useful about single stock prices they must first be coupled to trends during *trend analysis*. *Market correction* is a way of compensating for changes in the market. Both are described in chapter 4.4.

We then *align features with companies and trends*. Because a good indexation of news articles per company already exists in the data we get from our news provider it is not necessary for us to develop a separate system to classify news per company. Based on observations in chapter 4 we will align news articles with the first trend visible after the article is published.

Because we now have documents with features coupled to classes (trends) we can formulate our problem as a text classification problem (also known as *text categorization*). [Sebastiani, 2002] gives the following definition of text classification:

DEFINITION 1 *The task of* ***text classification*** *is to approximate the unknown* ***target function*** $\breve{\Phi} : D \times C \rightarrow \{True, False\}$*, that describes how documents ought to be classified, by means of a function* $\Phi : D \times C \rightarrow \{True, False\}$ *called the* ***classifier*** *(also known as* ***rule****,* ***hypothesis*** *or* ***model****) such that* $\breve{\Phi}$ *and* Φ *"coincide as much as possible". The amount of coincidence is the* ***effectiveness*** *of the classifier.*

This definition also takes into account the classification of a document into multiple categories. Our **target function** however is the reaction of the stock price given a document D where C is the set of pre defined **categories**, *surge*, *plunge* and *not relevant*, where there is only one category possible. Our target function can therefore also be written as $\breve{\Phi} : D \rightarrow C$.

Our way of training in the *create language model* step is an example of supervised learning because the learning process is supervised by the knowledge of the categories and the training instances belonging to them[1].

The classification is done in our system in figure 3.1 when we *predict trends using classifier* as described in chapter 7. In this chapter we will compare different classifiers to discriminate between classes based on their features. The *evaluation* of the **effectiveness** of the classifier will be given in chapter 8.

[1] An example of unsupervised learning is described by [Ahmad *et al.*, 2002] where classification is done by self organizing feature maps (SOFM) which is a interesting algorithm if we want to see how documents are distributed over a two dimensional map and identify clusters. We will however classify documents into known categories and therefore focus on algorithms which exploit this fact.

Chapter 4

Input from the stock market

4.1 Introduction

To get an overview of the different aspects of the financial side of modeling the reaction of traders to news articles we conducted a literature study and several interviews. In this chapter we will give the results of this study with the corollaries for our project, which we will use as hypotheses in the rest of this thesis.

4.2 The influence of news articles on stock prices

Nowadays new statistical techniques are still being used to predict the stock market; from evolution strategies by [Korczak *et al.*, 2001] to Adaptive Belief Systems by [Hommes, 2002].

But how well does statistical analysis of previous stock prices predict the course of a stock price? To determine whether we should take into account the analysis of previous trends in stock prices for our system, we looked at the *semi-strong market efficiency* as described in [Breatley and Meyers, 2000]. This concept implies that the stock price incorporates all information which is enclosed in previous prices and previously published information, so when new articles arrive, the stock market will immediately react to this information.

Furthermore, Breatley and Meyers show that stock prices follow a so called, *random walk*: changes in price take place independently from past prices. A reason for this is that if there are patterns visible in the prices, people will follow these patterns immediately, thereby making them useless to rely upon, as seen in figure 4.1 where as soon as the pattern of the upswing (an upward turn in a security's price after a period of falling prices) is recognized the price will immediately rise to the highest expected price of the upswing. News articles do not have this problem because they report a new state of a company which cannot be known on forehand. This was also confirmed during our interviews with traders. Therefore the following hypothesis will be used for our project:

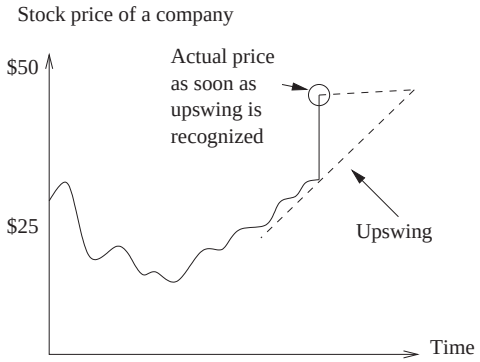

Figure 4.1: What if statistic trends are immediately recognized.

HYPOTHESIS 1 *News article analysis has significant advantage above trend analysis and it is not necessary to include the latter in our system.*

To quote Hommes: "Theoretical analysis of stylized evolutionary adaptive market systems [...] and its empirical and experimental testing may contribute in providing insight into the important question of whether asset prices in real markets are driven only by news about economic fundamentals, or whether asset prices are, at least in part, driven by *market psychology*.". This thesis is based on assuming the former and leaving the prove of the latter to mathematicians and psychologists.

But does this news about these economic fundamentals actually has influence on the stock price?

The answer to this question is not only made clear from the *efficient market hypothesis*, which says that the stock price will react immediately on new information. It is also concluded in [Patell and Wolfson, 1984] that the initial price reaction is evident in the first pair of price changes following the release of a new news article on the "Dow Jones News Service".

[Chan *et al.*, 2001] also confirm the reaction on news articles. They however also note that the price stabilizes relatively fast. Interesting is that the volume does not stabilizes as fast which could be due to the fact that traders after some time are more assured of their information and a small price difference will pay off.

Looking at newspapers, Chan *et al.* shows that economic news always has a positive or negative effect on the number of traded stock, on the day itself, on the day after and even on the day before. This last fact they attribute to the pre-announcement on the day before the newspaper came out.

Furthermore, the Ænalyst program by Lavrenko *et al.* was even able to predict the reaction of the stock market on news articles.

Finally Breatley and Meyers shows that the hypothesis that the stock price is always equal to the intrinsic value is almost impossible to test because traders often look at the current stock price compared to the price the day before. Because of this the influence of news articles will be even clearer.

We therefore also make the following hypothesis:

HYPOTHESIS 2 *News articles have an immediate influence on the stock market.*

This is also the reason that for the acquiring our news data, the best source for financial news is chosen, which also provides most other traders with real time objective articles, namely Dow Jones Newswires [Dow, 2003]. Although subjective articles may give a better idea of the sentiment around a company, they are often too late and are often not of immediate influence to the decisions of the traders.

Timing is also of essential value. According to interviews with traders the moment in which the reaction on a news new article is predictable is only immediately after the news article is published. Sometimes it could be possible that the effects of an article were already reflected in the stock price (also known as *in the market*), but mostly this only becomes clear after a while. Initially the price will rise if the article is positive and drops if it is negative. This goes well together with the fact that traders have only limited processing capacity, placing emphasis on recent events. Because of this, traders initially overreact to news articles as reported by [Raghubir and Das, 1999] and Breatley and Meyers[1]

[Patell and Wolfson, 1984] further reported that the simpler the trading rule, for example "If the dividend grows, I buy more stocks", the faster the market adjusts according to the market-efficiency hypothesis. So if we work with very simple trading rules, immediate reactions should be expected. The main influence of the "Dow Jones News Service" articles mentioned by [Patell and Wolfson, 1984] stops 60 till 90 minutes after they appear, where at the beginning of the next day there are also some changes visible. The maximum rise or drop of the stock price reported by them is within a time period of five minutes.

Research by [Brown and Warner, 1980] also supports the importance of time suggesting that taking stockprices during the day (also called intraday data) instead of closing prices will make the recognition of abnormal returns better, where the easiest abnormal returns to detect are with small companies. They even say that having the exact time of the expected abnormal return, due to the arrival of an article is more important than the method of correcting for the market changes, described later. We therefore make a third hypothesis:

HYPOTHESIS 3 *The predictable influence of news articles is only directly after it arrives, we must therefore react as fast as possible on incoming news articles and alignment of news articles with reactions is very important.*

Important in the way we are modeling our information is that we work with snapshots of continuous data in stock prices as described in [Dunham, 2002], namely closing prices. We must therefore deal with the assumption that the effect of a news article lasts long enough to be captured in such a snapshot. Because we have no other data available we simply make this assumption and come back on this subject in the evaluation of the results in chapter 8.

[1] Interesting in this case is that Breatley and Meyers reported that traders in the long term underreact to new news articles. "It looks as if traders observe the hot new issues, get carried away by the apparent profits to be made, and then spend the next few years regretting their enthusiasm"

4.3 The domain of news articles

In interviews with traders it became clear that we have to focus on a specific branch. Both, to be able to focus on a branch in which news articles have great influence, and to be able to have more specific domain knowledge.

A suitable branch would be pharmaceutical companies, this because the value of these companies is mainly determined by patents and new information about the approvement or expiration of a patent has much influence on their stock price. The chance whether a patent is approved is told to us by traders to be very unpredictable and therefore the chance that information about whether the patent will be approved is already rightly suspected by other traders is minimal. If the patent is approved (or not) this information is stock price sensitive and will be announced to all traders at the same time, having a large influence on the stock price. Another reason to chose the pharmaceutical domain is because in this domain are, contrary to for example the IT-industry, not too many complex relations between companies and less unpredictable reactions of the traders.

In interviews we did with traders as well as professors it was made clear that most interesting companies in the pharmaceutical domain release their information in English and fastest disposal of news articles would be in the English as well. We therefore conclude that for the news articles we must focus on the English language.

4.4 Preparation of the stock price

After deciding to take the stock price as reaction on news articles two problems arise.

Firstly we must take into account that a stock price can change in value each time a transaction is being made. If the first transaction after the appearance of a news article is for example made by two agencies who did not yet read the article, the immediate effect of the article on the stock price would be zero. We must therefore generalize over these values by doing trend analysis for these prices as described in [Nagy, 2001].

Secondly, from Breatley and Meyers and the interviews with traders also follows that it is important to compensate effects for the market to isolate the effect from a news article on the stock price. Brown and Warner describe the following three methods to do so:

Mean adjusted returns in which returns are compared with the returns from the same company on other days.

Market adjusted returns which means comparing the stock prices of a company to the prices of other companies at the same moment.

Market and Risk adjusted returns which takes also other high risk assets into account.

The best method for estimating abnormal returns according to Brown and Warner and therefore the most appropriate for our research is *Market adjusted*

returns with an *Equally Weighted Index*[2]. A method to calculate the abnormal return for one hour, using this method, where the return of the stock in that hour is $\tilde{r}$ and the return of the market is $\tilde{r}_m$ is described by Breatley and Meyers as follows:

$$r_{abnormal} = r_{actual} - r_{expected} \quad (4.1)$$
$$= \tilde{r} - (\alpha + \beta\tilde{r}_m) \quad (4.2)$$

In this equation α is how much the stock price changed when the market did not change and β how much extra the stock price changed for every percent change in the market index. Breatley and Meyers mentioned that it is important to choose the α and β values when the market behaved normally. To illustrate the algorithm, this easy example:

EXAMPLE 1 *Suppose β is 1 for ABC Company, so the stock price of this company is expected to change 1% for each percent change in the market. Furthermore we know that α for this company is equal to 0 so the stock price of this company normally did not change when the market did not. Now suppose the average market return to a stock was 10% for some hour, meaning stocks overall were 10% higher at the end of the hour than at the beginning. And suppose that the stock of ABC Company had risen 12% in that period. Then the* ***abnormal return*** *or* ***market adjusted return*** *of ABC Company's stock was 2%.*

We implemented this as a feature in our system with α equal 0 and β equal to 1 because we assume that each change in stock price is abnormal and all medical companies are equally effected by external factors. We return to this assumption in the evaluation chapter (chapter 8).

4.5 Domain knowledge

Out of the literature and our interviews, we also extracted some domain knowledge such as specific words which could be visible in news articles and patterns in the reaction of traders to certain events. These different words and patterns seen in reactions on news are listed here and will be used as domain knowledge for our system.

We will first list some general rules of thumb, which are observed in previous research after which we will list some special features for our domain and explain why they are important.

4.5.1 Rules of thumb

First [Patell and Wolfson, 1984] report that earning reports have greater effect on the market than dividend reports, taken into account that they can contain more information. In [Agrawal and Mandelker, 1992] it is shown that in merger proposals the buying companies experience share-price drops whereas acquired companies experience increased share prices. Furthermore traders report that the effect of new products is smaller in a *bear market*[3] than in a *bull market*[4].

[2]Equally Weighted Index: Counterpart of the *value weighted index*; A stock index in which each stock affects the index equally instead of in proportion to its market value.

[3]Bear market: A market in which prices are declining.

[4]Bull market: A market in which prices are rising.

Also in interviews with traders it was concluded that in general analyst reports do not have so much influence, what *is* interesting are reports of bond status changes. A downgrade[5] in a bear market can for example have a clear negative effect. Another way in which analyst reports have influence is the creation of a certain *atmosphere* around a company. If, in the course of time this atmosphere becomes negative, this will result in heavier reactions on negative news articles.

Another moment when reactions on news articles are heavier mentioned by [Brehm *et al.*, 2002], is when the prices rice or drop very quickly. This could be due to the uncertainty in these situations in which people tend to depend on other sources such as news articles.

4.5.2 Pharmaceutical trading knowledge

For the pharmaceutical sector there are several special concepts to look at:

Patent application. Most of the pharmaceutical industry is about patents. In the application for a patent two factors play a special role:

1. Potential of discovery.
2. The amount of people suffering from the disease the medicine cures.

Phase of the patent in the approval process. It would be possible to look from which which phase to which other phase a patent goes. This is reported in news articles. Depending on which phase the patent approval process will go in, a news article will have a different effect. Some companies run completely on the discovery of a specific medicine. Important in these transition of states is the term *fast track approval process*.

Analyst reports. In these reports one can find the stages of the approval process the different medicines from a company are in. The reports themselves have little influence, one can look if the stages from the analyst reports could be used by our system.

Patent expiration. When a patent expires for a medicine, there are a number of companies which come to the market with a *generic* version of this product and go for the lowest price[6].

Profit. Much profit leads also to commitments, for example to export cheaper AIDS-medicines to Africa.

4.6 Summary

In this chapter we discussed literature and interviews about the stock market leading to the following three hypotheses:

1. News article analysis has significant advantage above trend analysis and it is not necessary to include the latter in our system.

[5] Downgrade: The lowering of a bond rating by a rating service.

[6] Note that the patent for Viagra is not yet expired and therefore SPAM mail talking about a generic version hereof is spreading false lies.

2. News articles have an immediate influence on the stock market.

3. The predictable influence of news articles is only immediately after it arrives, we must therefore react as fast as possible on incoming news articles and alignment of news articles with reactions is very important.

Based on these hypothesis we can expect our system, if it finds the right features, to do well in predicting the reaction of news articles. Furthermore this chapter suggests to select our news articles in the English language from pharmaceutical companies, and suggest to prepare our stock prices by doing trend analysis and compensating for the developments in the market. These two components are at the basis of our system as seen in figure 4.2. The domain knowledge which we discussed is also seen in this figure and added to our system as described in 6.3. We will test our system both with and without compensation for developments in the market.

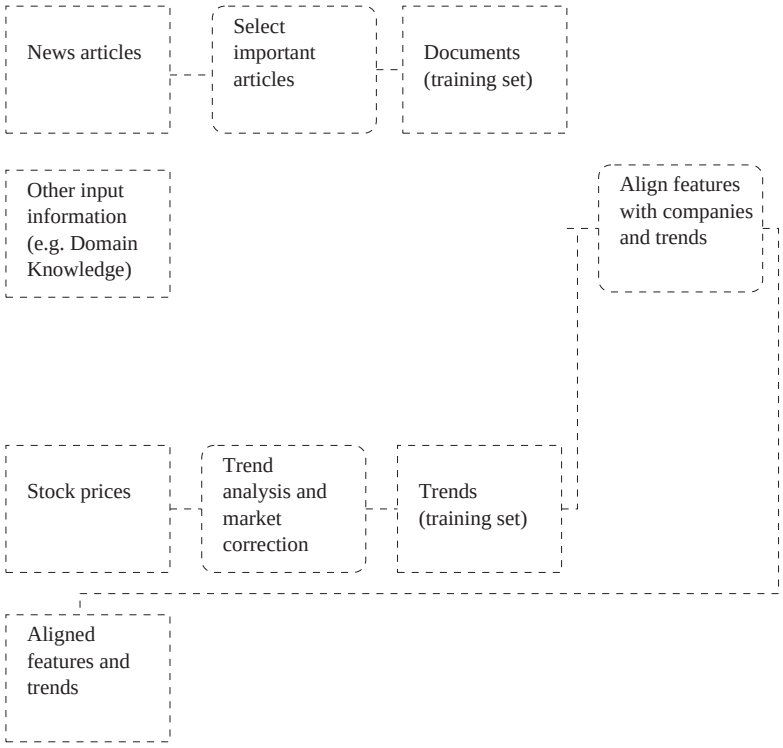

Figure 4.2: Contribution of the market chapter to our system.

Chapter 5

Word based features

5.1 Introduction

As we described in chapter 2, one of the most important design choices for a system like this, is how to choose the features. The most standard method is to see each word (unigram) in a document as a feature.

In this chapter we look how to improve on unigrams and look at combination of two of them (bigrams). This chapter is a sort of introduction to the next chapter in which we further improve the features.

5.2 Unigram based

Several extensions to unigram modeling have been proposed, such as the use of a *stop list* by [Jurafsky and Martin, 2000] and [Salton and McGill, 1983]. The motivation for this extension is that high frequency, closed class terms are seen as carrying little semantic weight and are thus unlikely to help with retrieval and eliminating them can save considerable space in an *inverted index*. This is a method of indexing in which for each word, all documents are mentioned in which it appears so it is possible to do very fast search queries on these words. However for our system, this removal of stop words could remove too many important words such as *above* or *interesting*. Plus, words which are not discriminating between trends will have no negative effect on classification because that is what they implicate and therefore it will not matter if they are included or not.

Another often used technique is the requirement of a *minimum appearance* of words or features before using them. A reason to require this is because a word which appears only two times and by a chance in the same category would wrongly be taken for a good feature for this category. Because it could be possible that a news article sometimes does not cause the trend which follows it, there is fair chance such features exist in our system. We will therefore require such a minimum appearance as we will see in chapter 8.1.3.

Stemming algorithms such as the Porter algorithm as described by [Salton and McGill, 1983] and [Jurafsky and Martin, 2000] can improve recall by con-

verting words to word classes using their stem, with rules such as:

$$ATIONAL \rightarrow ATE(e.g. relational \rightarrow relate) \tag{5.1}$$

Another advantage is that for words which appear not often but still are specific to a certain category their stem can still fulfill the property of appearing a certain number of times.

Some mappings of this algorithm are however too rigorous, for example *policy* and *police* are mapped to the same stem and other mappings are too soft such as *noise* and *noisy* which are mapped to different stems. [Krovetz, 1993] gives the results of a number of tests of whether the Porter algorithm actually improved IR performance but overall he found only some small improvement, mainly with smaller documents. And there is a better alternative; using WordNet [Miller *et al.*, 1990] for the *conversion from a word to its lemma* (uninflected form).

WordNet is a lexical reference system in which English open class words (nouns,verb, adjectives and adverbs) are organized into synonym sets, each representing one underlying lexical concept. Different relations link the synonym sets. By using WordNet to convert words to their lemma, the reduction is adequate and an objective mixture is chosen by on one hand not leaving out good, but little appearing, discriminators but on the other hand not oversimplifying. This is for us enough reason to include this technique in our system.

Another way to use WordNet and often applied in Information Retrieval is the use of *thesaurus classes* such as in [Basu *et al.*, 2001]. It has the same advantage as reducing a word to its lemma; to make not so often occurring discriminating features more visible, but then in a stronger form. The problem could be that it will reduce precision because words occurring in the same thesaurus classes will not exactly have the same meaning. We will discuss generalization for our features in chapter 6.5.2.

An alternative for WordNet is The Integral Dictionary (TID) which is described in [Dutoit and Nugues, 2002] and is of similar size. It is organized around ontological concepts where each concept is annotated by a few words that describe its content. When a concept is entirely lexicalized, these words are reduced to one. Concepts are classified into categories. The main ones are classes and the themes. Classes form a hierarchy and are annotated with their part-of-speech. Themes are concepts that can predicate the classes.

The crucial difference between TID and WordNet is the organization of the word and concept network. In the WordNet model, concepts are most of the time lexicalized under the form of synonym sets (synsets). They are thus tied to the words of a specific language. In TID, themes and classes do not depend on the words of a specific language and it is possible to create a concept without any words. This is useful, for example, to build a node in the graph and share a semantic feature that is not entirely lexicalized.

The Integral Dictionary contains approximately 16,000 themes and 25,000 classes, the equivalent of 12,000 WordNet synsets (with more than one term in the set), and, for French, 190,000 words. WordNet contains approximately 95.600 different word forms organized into 71.100 synsets. Because TID is not yet as well developed for English as it is for French it is not yet suited for our purpose to extract English lemmas or thesaurus classes.

5.3 Bigram based

If we want to extend unigram features and thereby for example be able to differentiate between *good results* and *good weather* we can use n-grams instead of or as addition to unigrams. The fact that n-grams have a positive effect categorization tasks has among others been shown in [Fürnkranz, 1998] and [Tan *et al.*, 2002]. Fürnkranz shows that results improved by taking bigrams or trigrams but that sequences with a length greater than 3 no longer improved them (and made them worse in some cases).

Tan *et al.* see bigrams as something that reinforces the unigram method instead of something that replaces it. This is made clear by the example bigram "artificial+intelligence" which was one of the bigrams they found very good at classifying the computer science domain. However the word "artificial" alone was also a good feature, which would be dropped if they only looked at bigrams.

[Jurafsky and Martin, 2000] notice that current n-gram based systems are based on word form which is the inflected form as it appears in the corpus. This is however for many domains, including ours, not a good simplification because for example *loss* and *losses* will be treated as different words. Jurafsky and Martin therefore suggest, as we did in chapter 5.2, to base the n-grams on their lemma.

If we want to deal with a minimum number of appearances of the found n-grams an important implementation optimization can be made using the *Sub-sequence property* as given by Fürnkranz:

Sub-sequence property The number of occurrences of a sequence of m words in a document collection is bounded from above by the number of occurrences of each of its sub-sequences.

In this way we can, while extracting n-grams, already not take into account n-grams from which their component words appear less than the minimum requirement of appearance of the n-gram itself.

But even if we make use of this property, we still cannot take all bigrams because we would arrive at $|words|^2$ possible bigrams, which is for us about 1.6 million. Therefore our suggestion is to take only bigrams into account which include relevant words for the domain, such as *patent* or *company* which we will define in our domain knowledge as described in chapter 6.3.1.

The concept of n-grams comes from the language technology, where it is used to predict a word based on its preceding words. However, in our research we do not concentrate on predicting the course of a document but the course of the stock price, using bigrams as features. This is the reason why we do not require bigrams to be immediately next to each other but allowing a certain distance between them in as sentence. Further information about n-grams and using them to predict words in a document are given are given in [Jurafsky and Martin, 2000] and [Ordelman, 2003].

5.4 Summary

In this chapter we shortly focussed on the possibilities of improving words as features for documents as introduction to our next chapter. We argued why we will not use stemming but use WordNet for generalizing words to their lemma

or synset, which we will also take into account with our more extensive features. We argued why we will not use a stop list and how we could efficiently extract n-grams as features. We implemented this as seen the update of our system in figure 5.1, the results of this thesis will however focus on the next chapter in which will discuss our final features.

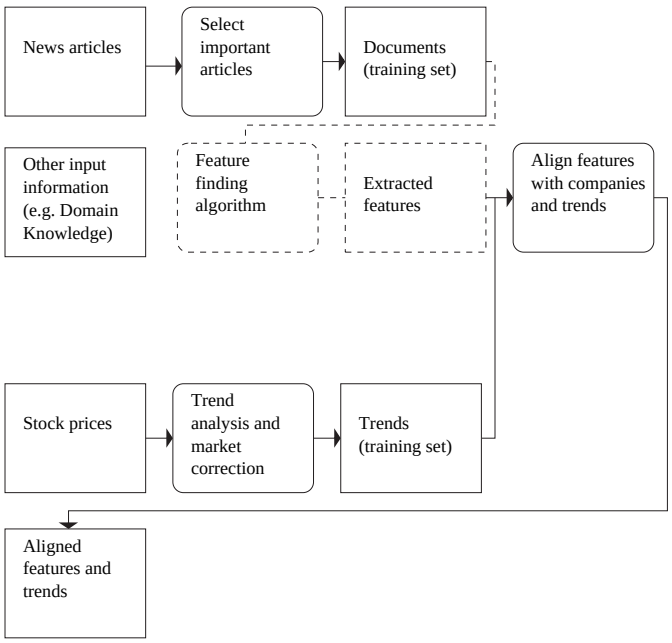

Figure 5.1: Contribution of word based improvements to our system.

Chapter 6

Template based features

6.1 Introduction

To have more extensive features than the words and bigrams mentioned in chapter 5, we can also construct our own features out of the sentences using more knowledge about the relation between the words.

By using this knowledge we can take into account negation and the relation between unigrams. In this way, we can differentiate between the sentences "the results were good" and "the results were *not* good" or between the sentences "*GlaxoSmithKline* acquired *Corixa*" and "*Corixa* acquired *GlaxoSmithKline*". Note that this behavior is much more important in our domain of predicting stock prices than for example in the domains of religion or coffee as we will discuss in chapter 7.4.

A possibility for our system to construct these features is the application of semantics. Semantics is concerned with the *meaning* of words, expressions and sentences and is among others described by [Reidsma, 2001]. In his thesis he discusses *knowledge graphs*, a way to represent knowledge about the world and about a sentence. If we could automatically construct these graphs out of a sentence we would have a so called *sentence graph* with which it would be possible to reason about the semantics of the sentence.

This technique is however not yet attractive because it lacks a vocabulary containing words with their meaning which is large enough to construct these sentence graphs. What we however could use is a less specific system, such as syntactical graphs, and combine them into a sentence graph, as mentioned by [Willems, 1993] and Reidsma. This technique would for example identify in the sentence "the man breaks the glass with the stone" that the subject is the man, the object is the glass and it happens with a stone. Problems of these syntactical graphs are with sentences such as "Peter eats vegetables with sugar" or "Peter eats vegetables with a fork", where the meaning of "eat with" is ambiguous if we do not have knowledge about *sugar* or *a fork*.

Considered that semantic analysis is not yet attractive, syntactic analysis is a viable alternative for us, which is the reason why we will discuss possibilities for syntactic analysis of texts in chapter 6.2. We however do not need the whole syntactic graph of a sentence but only a representation which we can use as features. We will therefore discuss the features we extract, using this analysis,

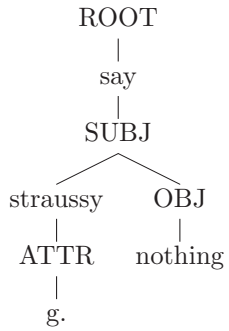

Figure 6.1: Syntactic analysis of "G. Straussy said nothing".

in chapter 6.4 and 6.5.

Another point on which we can improve on the bigrams as described in chapter 5.3 is by using more domain knowledge to reason about the components of our features. In this way we are able to reason about very general concepts such as "the company is warned by *something which has formal power*". Where the *something which has formal power* can be a court or a trade organization. We will discuss how to use domain knowledge in chapter 6.3 and how we used it in chapter 6.5.

6.2 Syntactic sentence analysis

6.2.1 Named entity recognition

To learn features specific to our domain we need to supply the system with knowledge of companies, medical terms and stock terms. At first the most important task is to recognize names such as *GlaxoSmithKline*, but also to see that *GSK* is the ticker symbol for *GlaxoSmithKline* and that when we speak of *GSK* or *GlaxoSmithKline* we mean the same entity. This process is done during the named entity recognition phase.

In the sentence "The head of charles river laboratories international said nothing" we first tag "charles river laboratories international" as the company which has the symbol *CRL*. This is done using our domain knowledge. The result of this step is a set of documents annotated with information about which words are named entities by our knowledge and which words are, according to our knowledge, used for the same entity.

However the biggest part of the named entity recognition is done by our syntactic analyzer which for example recognizes in the sentence "G. Straussy said something" the structure in figure 6.1, where it is clear that "Straussy" is the subject and "G." is an attributive nominal of it (see appendix B for an explanation of the tags). This is the reason that except for the named entities we explicitly want to use in the rest of our system we do not need to tag the named entities.

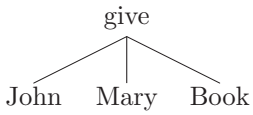

Figure 6.2: Example of a semantic structure of the sentence "John gives a book to Mary."

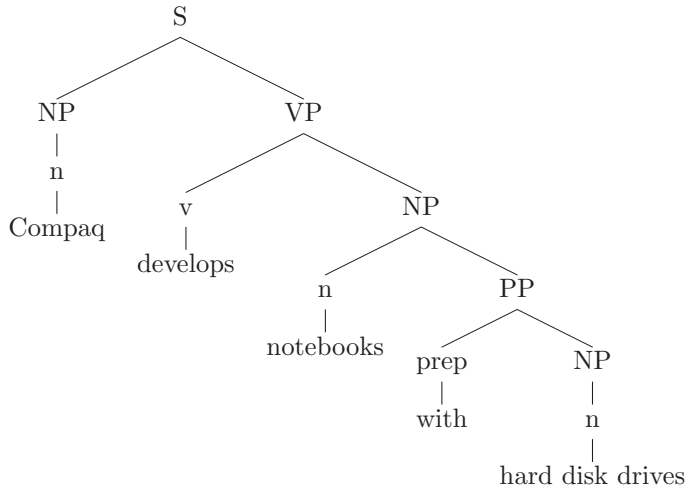

Figure 6.3: ParseTalk grammar.

6.2.2 Syntactic structure

To extract out of the documents the syntactical structure needed for feature extraction there exist several methods.

[Lie, 1998] uses its own Sumatra top-down parser to get a syntactic structure out of a document. Next semantic relations are discovered, with as special relations the *ISA*, for something which *is* something, and *HASA* ,for something which *has* something. The *ISA* hierarchy is among others used by his anaphora resolution algorithm. For all other relations the stem form of the main verb is used of which we can see an example in figure 6.2. He describes two differences between his syntactic parser and semantic analyzer.

1. The output of the semantic analyzer consists of a relation instead of a parse tree or tagged sentence parts.

2. The semantic analyzer can interpret the meaning of sentence constructions.

Another system is ParseTalk, of which we can see two examples in figure 6.3 and 6.4 and is used by [Hahn and Markó, 2002] as an architecture for text knowledge extraction in the IT domain. However this is more a method than a system which can be used in our project. Two systems that *are* available to use in our project are the Connexor Functional Dependency Grammar (FDG) and Link Grammar.

Both systems create syntactical dependency graphs from English sentences of which demos are available at their websites ([Con, 2003] and [Lin, 2003a]).

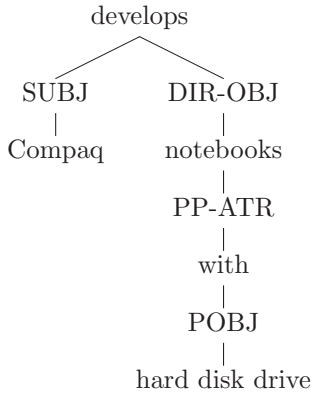

Figure 6.4: ParseTalk dependency grammar.

The difference between dependency grammars and constituent-based grammars as seen in figure 6.3 is that the former gives relations between pairs of words in the document whereas the idea of the latter is to adjacent items into larger categories (constituents). The advantage of dependency sets is that they lie closer to knowledge representations.

The Connexor FDG is a full syntactic parser that produces both morphological information for word-form tokens and functional dependencies representing relational information in sentences, e.g. it shows simple and complex entities in sentences and it describes relations between these entities. This comes down to:

- Objects and ontological facts (names, organizations and places), their
- Actions ('who did what to whom') and the
- Circumstances (where, when, why, how...)

The different tags it can produce are described in appendix B.

[Mollá and Hutchinson, 2002] compare the Connexor dependency parser to Link Grammar [Lin, 2003b] for implementation in ExtrAns, an answer extraction system that operates over Unix manual pages. The dependency structures are hereby converted to logical forms and answer extraction is performed by finding those sentences whose logical forms form a superset of the logical form of the predicates of the question. Mollá and Hutchinson also gives the main differences between the two analyzers:

Direction of dependency In link grammar there is no explicit direction in the dependencies.

Clausal heads In the Connexor FDG, the head is the verb, in Link Grammar it is the subject.

Structure The output graph of the Connexor FDG is always a tree structure whereas the graph of a link grammar is not always.

Conjunctions In Link Grammar for each conjunction a different parse is generated, in the Connexor system they are coordinated using the *cc* tag.

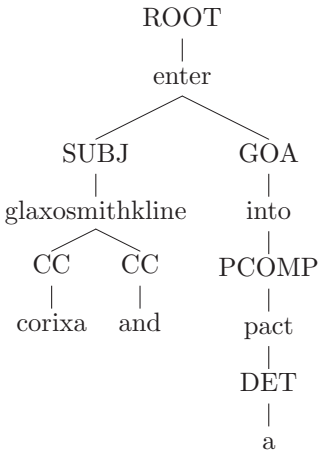

Figure 6.5: FDG tree of the sentence "Corixa and GlaxoSmithKline enter into a pact."

Dependency types Both have different dependency types, Link Grammar about 90, Connexor 36.

Non dependency information Connexor returns more syntactic background information, link grammar returns only suffixes on some words indicating in which syntactic class they belong.

An example analysis of the sentence "Corixa and GlaxoSmithKline enter into a pact" can be found in figure 6.5 for the Connexor FDG and for Link Grammar here:

```
                                    +---Js--+
   +------------Sp------------+--MVp-+  +-Ds-+
   |                          |      |  |    |
Corixa and GlaxoSmithKline enter.v into a pact.n

                                    +---Js--+
                  +-----Sp----+--MVp-+  +-Ds-+
                  |           |      |  |    |
Corixa and GlaxoSmithKline enter.v into a pact.n
```

Because of the robustness and clear structure of the Connexor output we will use their system for our project.

[Nicolas *et al.*, 2003] describe a system which uses the combination of the Connexor system and WordNet to reorder query results and rank the more relevant ones higher. Based on similarity in syntactic graphs of the Connexor system and using the distance between words based on relations in WordNet, sentences in the returned documents by the search engine and the query are compared and documents with a higher similarity are ranked better. They report positive results.

[Khoo *et al.*, 1999] also use the Connexor FDG to extract cause and effects from a medical database to see what influences medicines have. They report

that the usage of a dependency grammar leads to a smaller number of, manually constructed, extraction patterns needed when compared to part-of-speech tagging. They report an accuracy about .54 for the cause and 0.58 for the effect. One of their main suggestions for feature work is the automatic construction of the extraction patterns.

6.2.3 Reflection

In this chapter we described how our named entity recognition works and how it can be done simple because of our FDG system. Furthermore we looked at different syntactical analyzers of which we choose the Connexor FDG because:

- It generates dependency trees which lie closer to knowledge representations than constituent trees.
- The dependency trees it generates are robust and have a clear structure.
- It is available and possible to use it in our system.

6.3 Domain knowledge

6.3.1 Representation

Another important aspect of our system is how to represent the knowledge extracted from expert sources about our domain and about the world.

The representation hereof is called an ontology [Chandrasekaran *et al.*, 1999]. According to Chandrasekaran *et al.* there is general agreement about the following basis of knowledge:

- There are *objects* in the world.
- Objects have *properties* or *attributes* that can take *values*.
- Objects can have *parts*
- Objects exist in various *relations* with each other.
- Properties and relations can change over *time*
- There are *events* that occur at different *time instants*.
- There are *processes* in which objects participate and occur over time.
- The world and it's objects can be in different *states*.
- Events can *cause* other events or states as *effects*.

This is the ground for an ontology of a system in which we can represent our information. OpenCyc [Ope, 2003b] already has this basic knowledge or *upper ontology* and also has a common sense reasoning engine. This is the main reason why why we chose to implement our domain knowledge in it. The hierarchy of OpenCyc can be seen in figure 6.6.

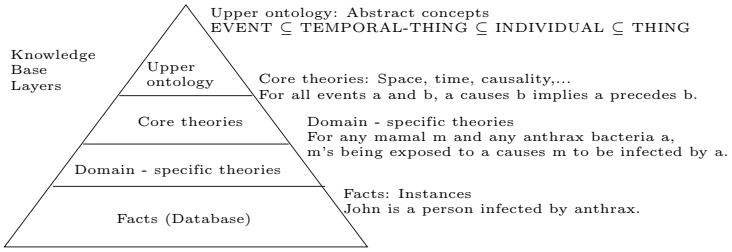

Figure 6.6: The OpenCyc hierarchy.

In OpenCyc we can for example represent that a court has formal power. This can be done in so called *assertion* as follows:

$$\#\$hasFormalPower\ \#\$Court - Judicial \tag{6.1}$$

Following this assertion we can find a feature in which *something that has formal power* warns a company. This feature can have a positive or negative effect on the stock market.

In OpenCyc one can have microtheories which contain a set of assertions. Each microtheory bundles assertions based on a shared set of assumptions on which the truth of the assertions depend, a shared topic (medicines, diseases, the stock market) or a shared source (e.g. articles from Dow Jones Newswires). All information within a microtheory must be mutually consistent and two microtheories can be related such that one of them inherits the assertions in the other. In OpenCyc we can for example make a microtheory which represents our knowledge about companies, medicines and the stock market.

An important distinction must be made between a concept and an entity. As an example take the concept *pharmaceutical company* which we will represent in OpenCyc as *#$PharmaceuticalProducingCommercialOrganization*, this concept will not change over time and as long as something is a pharmaceutical company it can take over other companies. However *GlaxoSmithKline* is an entity which will hopefully exist forever but can change over time because it might for example go bankrupt. An example of the *pharmaceutical company* concept in our OpenCyc hierarchy is given in figure 6.7.

It is also important where to place assertions in the hierarchy of known knowledge (the microtheory hierarchy). If accuracy (or completeness) is more important than efficiency, it is better to place the assertions higher in the microtheory hierarchy. If efficiency is more important it is better to place them lower and sacrifice completeness. For example in figure 6.7, when completeness is concerned we can say that each organization can have its headquarters in a country. When this fact is only important for commercial organizations and we want to have a more efficient system, we can make this assertion only for *commercial* organizations.

A general rule is that you can only extract from the knowledge database what we put in. For example we could for organizations make a difference between governmental or commercial organizations so our features could be more specific.

As we discussed in chapter 6.2.2 we will use the Connexor system to analyze our sentences. OpenCyc however also can include natural language information.

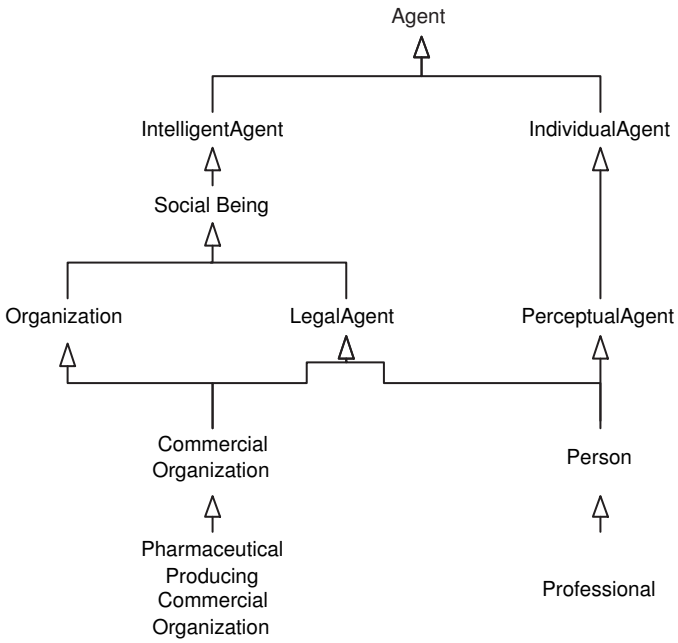

Figure 6.7: The pharmaceutical company concept in our OpenCyc hierarchy.

The more complex way of adding lexical terms and make the connection between words and concepts is the *TheWord*. *TheWord* is a Natural Language(NL) addition to a word which can be used to couple between words and meaning in OpenCyc. OpenCyc NL constants are terms which support OpenCyc's natural-language processing capabilities. These terms do not express common sense knowledge; instead, they encode information about English words and phrases. For example, any OpenCyc term that ends in '-TheWord' is a lexical (as opposed to common-sense) term. More NL tools will become available in future versions of OpenCyc, so it is expected that in the future, it will be easier to work with the TheWord-extension. For a start we will however just use the *nameString* method to indicate the different word forms in which a concept can be written.

Furthermore it is especially pointed out in the OpenCyc tutorial [Ope, 2003a], that the adding of vocabulary, for example by adding terms, is recommended over keeping the vocabulary as simple as possible and adding many assertions or complex expressions. It is more important what you say than trying to generalize so much it becomes not understandable. The CEO of Cycorp, Doug Lenat, formulated this as the *Physics Envy* where people in knowledge representation are wishing for the Maxwell's Equations of knowledge representation, "Four simple little things which can express how to represent knowledge and put it on a T-shirt. It's just not like that." [1]

Another interesting predicate to use is *#$completeExtentKnown*, it means that something is known to its whole extend, so if we have the assertion *#$completeExtentKnown #$border* and we do not know anything about a border be-

[1] The whole (quite strong) mail can be found at http://www-ksl.stanford.edu/email-archives/srkb.messages/484.html

tween France and China, it is not there. Another way of talking about completeness of information is the *#$completeCollectionExtend* predicate, where we can iterate over the whole collection. For example is we have the assertion *#$completeCollectionExtend monthOfYearType January February March* etc., we have defined all months and the reasoning engine "knows" that there are no others.

In OpenCyc many knowledge is already expressed, especially medical knowledge and knowledge about companies, for example the predicate *#$makingAnAgreement*. Therefore we should look carefully which predicates and concepts are already defined, which of them we want to use and to what extend we should use them.

To construct an ontology in OpenCyc it is advised in the tutorial to keep arity in relations as low as possible, preferably one or two. In this way more information can be extracted. For example if we define that Alexander Graham Bell invented the telephone in 1876 in one relation, it would be much more difficult to ask when the telephone was invented, than when we had two assertions; one about Bell inventing the telephone and the other about the invention date of the telephone. To make rules as general as possible there exists the predicate *#$exceptFor* to make exceptions to a rule. This can for example be used to define German grammar rules. Another interesting option is the *#$endsAfterStartingOf* predicate by which we can state that one patent phase occurs after another.

Another way to represent world knowledge is to use a Prolog based solution called Xi, as described in [Gaizauskas and Humphreys, 1996] and among others used in [Pastra *et al.*, 2002]. In this language one can for example define the following ontology as given by Gaizauskas and Humphreys:

```
top(X) ==> object(X) v event(X) v property(X).
object(X) ==> vehicle(X) v country(X) v person(X)
        v body(X) v floor(x).
    vehicle(X) ==> (car(X) v lorry(X) v motorcycle(X)) &
        (commercial(X) v private(X)).
    car(X) ==> (rover(X) v toyota(X) v renault(X)) &
        (twodoor(X) v fourdoor(X)).
event(X) ==> drive(X) v own(X) v lay(X).
property(X) ==>
    single_valued_prop(X) v multi_valued_prop(X).
        made_in <-- single_valued_prop(X).
        colour_of <-- single_valued_prop(X).
        lsubj <-- multi_valued_prop(X).
        on <-- multi_valued_prop(X).
```

Semantic rules can be made which automatically extract facts such as *body*(e1),*lay*(e2), *floor*(e3) and binary predicates for these properties such as *lsubj*(e1,e2), *on*(e2,e3). Advantages of this approach to representing knowledge are the ease of reasoning, the clearness of the ontology and compatibility with semantic rules. The disadvantage is that we need to construct our own ontology completely ourselves and because everyone does so, research in this area in the future will not lead to a world-encompassing ontology. The latter is the reason why we did not use this knowledge representation technique.

6.3.2 Creation

There exist some techniques to generate an ontology. [Hahn and Markó, 2002] for example describe the learning of grammar and domain knowledge at the same time. Here domain knowledge provides a base for judging in which category a new concept should fall. Grammatical knowledge on the other hand contains a hierarchy of lexical classes which makes increasingly restrictive grammatical constraints available for linking an unknown word with its corresponding word class each time an the word is mentioned again. As data set they take IT magazines where they learn for example *grammatically* that an item is a noun and *conceptually* that it is a "printer". They conclude that in the future learning should encompass several linguistic dimensions simultaneously within a unified approach. Their accuracy is 43% under optimal conditions for the ontology task and drops to 23% under normal conditions.

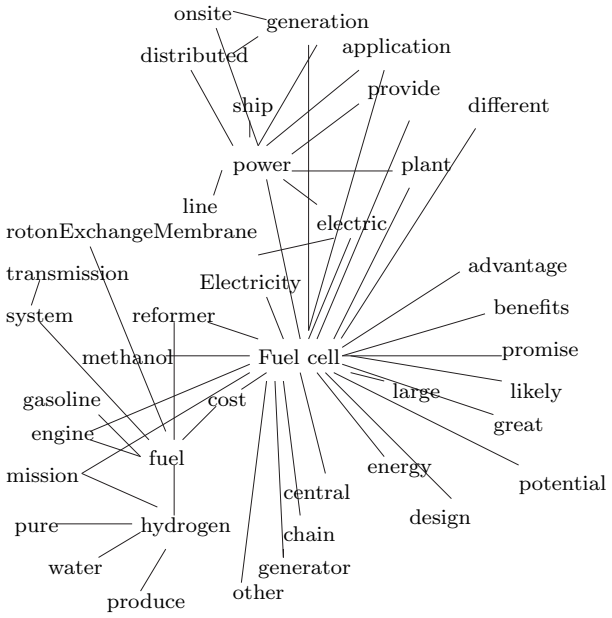

Figure 6.8: Example of relations found with Quorum.

Another way of finding relations between words is with the Quorum algorithm as described in [McGreevy, 1995]. Here relations between important words (or words appearing very often in the document) and other words in the documents are analyzed. The relations are analyzed within a context window, of which the proposed length is the average sentence length of the documents processed, and the value of the relation is the bigger the closer the words are in the window. In further processing the values of all the relations in which a word appears are summed for every word and this sum is used for estimating the significance of the words. An example of strong relations found with this algorithm in [Keijola, 2003] can be seen in figure 6.8.

We however will for this thesis only use expert knowledge as described in chapter 6.3 to fill our ontology because of the specific requirements of the ontol-

ogy in relation to stock market changes. We therefore constructed an ontology in OpenCyc from this domain knowledge and knowledge sources such as the Inpharm website [Inp, 2003], in which important concepts were as much as possible placed within the existing OpenCyc hierarchy. The complete list of concepts we added can be found in appendix C. For the rules of thumb we only took some of the important concepts, the rules itself are not used by our system.

6.3.3 Reflection

In this chapter we discussed OpenCyc as knowledge representation and reasoning system already having an implemented *upper ontology* and discussed some important design recommendations we can use to implement our ontology of the stock market in it. As an interesting aspect we discussed some simple techniques for creating an ontology automatically but concluded that because of our specific requirements of our ontology we chose only to fill it with expert knowledge.

The main reason for choosing OpenCyc above a Prolog based solution called Xi is because in Xi the whole ontology must defined by ourselves, so research in this area will in the future not lead to a world-encompassing ontology. However the question remains if OpenCyc will be the basis of a future ontology. We expect that if a standard upper ontology is formed and ontology editors like Protégé [Pro, 2003] will be able to import this standard work research in this area will be one of the most important in Natural Language Processing. Because, although ontology creation is not yet commonly seen as an NLP-task, we need it for reasoning about documents and it needs us to help automatically constructing it. On the website of Protégé many manually created ontologies can already be found.

6.4 Templates

6.4.1 Definition and use

In this chapter we will describe a way of extracting features using a template. According to [Wilks and Catizone, 2000] a template has the following definition:

Template A linguistic pattern, usually a set of attribute value pairs, with the values being text strings, created by experts to capture the structure of the facts sought in a given domain, and which Information Extraction (IE) systems apply to text corpora with the aid of extraction rules that seek those filters in the corpus, given a set of syntactic, semantic and pragmatic constraints.

Templates are at the basis for the five types of information extraction discussed during the last Message Understanding Conference (MUC-7) by [Marsh and Perzanowski, 1999] namely:

Named Entity recognition (NE) Find persons, organizations, currencies etc. (also in other languages).

Template Element construction (TE) Extract basic information related to organization, person, and artifact entities, drawing evidence from anywhere in the document.

Template Relation construction (TR) Extract relational information between TE elements.

Scenario Template construction (ST) Extract pre-specified event information and relate the event information to particular organization, person, or artifact entities involved in the event. (fits TE and TR results into specified event scenarios.)

Coreference resolution (CO) Capture information on coreferring expressions: all mentions of a given entity, including those tagged in NE and TE tasks.

This is the most common definition and use in the field of natural language processing. However our template must be used to extract generic features and can not be too domain specific. Therefore our template is much less structured than manually created ones. It *can* still be used to extract information, although that is not our main concern in this thesis. In this chapter we will give several example forms of templates which will lead to our own structure.

6.4.2 Forms of templates

In general

In the most general form, as described the precious section, the fields of a template are constructed by humans. This is the reason that these fields are from a human point of view descriptive. For example, if *SEM-MEDICINENAME* is a noun indicating a medicine, we could construct a template element *TE-MEDICINE* which looks as follows:

TE-MEDICINE Contains information about a medicine. It consists of the following fields:

- ID of SEM-MEDICINENAME
- Chemical formula: $C_8H_9NO_2$

The problem is that the coupling to the syntactic analysis of sentences is less evident. As a matter of fact the filling of these templates is the assignment in the Message Understanding Conferences.

Yangarber

[Yangarber *et al.*, 2000b] and [Yangarber *et al.*, 2000b] do not describe a new sort of templates but describe a new method to discover patterns to fill these templates. These patterns are focussed on the syntactic analysis of sentences.

Each pattern is a tuple of the major roles of the sentence; the head of the subject, the verb group and the object and object complement. Two manually constructed examples which are important in the domain of management successions are given in table 6.1.

Based on these manual "skeleton" patterns, new patterns are found. This is based on the idea that documents which contain the manual patterns are relevant and contain new relevant patterns. Depending on number of times new patterns appear in relevant documents, these patterns are considered relevant as well and the procedure starts over again by searching for relevant documents.

Subject	Verb	Direct Object
C-Person	C-Resign	×
C-Company	C-Appoint	C-Person

Table 6.1: Two skeleton patterns for management successions.

If enough patterns are found, these patterns are manually converted to extraction patterns to fill the templates, for example the template for management successions as given in appendix A.

Collier

[Collier, 1996] describes ideas for automatically creating templates for information extracting. He reasons from the perspective that there are three fundamental types of elements of a template: the objects which interact, the relationships representing the interaction and the features that are specific to the objects and relationships. He argues that a fundamental object for a certain category should appear in every relevant document, so entities which occur in every relevant document are considered to be objects. As a a plausible approach for constructing a template he considers the verb/subject/object interaction in sentences, so the following two lines are considered as simple templates:

```
<PERSON> retire <ORGANIZATION>

<ORGANIZATION> employ <PERSON>
```

Furthermore relationships and objects will be in all relevant documents so if sufficient generalization of verbs into higher order categories can be carried out, then the frequency of occurrence of these higher order categories will provide evidence concerning the significant relationships that occur within relevant documents. As candidates for generalization he proposes WordNet and the Longman Activator[2] and observes that for some domains extremely useful relationships exist in these sources, although it is not expected to be sufficient for each domain. Features of relationships, such as a date of a retirement, are acquired after the relationships are extracted by looking at the context of the relations mainly at sentence level.

Roux

The idea of [Roux *et al.*, 2000] was not about automatic template generation but to automatically create (simple) medical ontologies out of documents using syntactic parsing.

For example, the sentence "Antp protein represses the BicD gene.", is syntactically parsed to the form as seen in figure 6.9. The syntactic information is then stored in a so called "conceptual graph", as seen in figure 6.10, where the verb is the key element and the nouns are connected to the verb with their syntactic relation. The idea is to combine these graphs into a network of graphs to extract and learn knowledge from them.

Although Roux *et al.* never mention templates, they do give a simple syntactically extractable form for features, useful for representation of information.

[2] A dictionary of synonyms which is not publicly available online

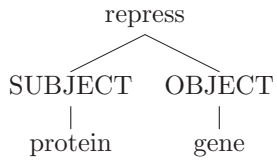

Figure 6.9: Syntactic analysis of the sentence "Antp protein represses the BicD gene."

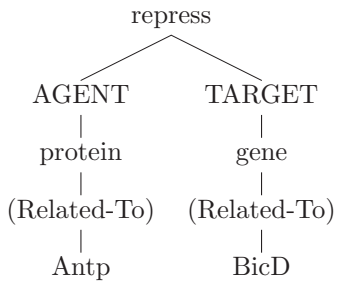

Figure 6.10: Conceptual graph of the sentence "Antp protein represses the BicD gene."

6.4.3 Reflection

In this chapter we saw uses of templates and discussed how we want to use them. We think that our template should be as general as possible. In this way we can find many general features in a document using a simple general way of extraction. The importance of certain features for a category can now simply be learned while creating the model. And, if we want to use these features in a future stage for ontology creation, we can combine them to a whole[3].

Furthermore we want to keep the template tight to the syntax of the text because we want to generate them automatically

We will therefore base our template form on the work of [Roux *et al.*, 2000] and [Collier, 1996] and use the following structure of a template where an agent has a certain relation to a target:

```
<AGENT> <RELATION> <TARGET>
```

We could for example from the sentence "Aventis welcomes positive study results for Arava drug" fill the following template:

```
AGENT:aventis RELATION:welcome TARGET:result
```

Because we want our template to be as generic as possible, we do not take features of the relations or objects as mentioned by Collier into account. We shall however add some useful information to the template in chapter 6.5

Important to note about our template form is that it is very generic. In fact, it can be used to extract more specific templates as suggested by Collier, but this is not our purpose. Our purpose is to use this generic template as method

[3]This argument is similar to the recommendation of OpenCyc to keep the arity small in chapter 6.3.1 in which we will give an example hereof.

for feature extraction and let the classification method decide whether a certain filling of our template has a positive or negative effect on the stock market. A nice property is of course, that this form can lead to more information about the different categories on which we classify but this is not our starting point.

[Wilks and Catizone, 2000] give an overview of the whole domain of template construction, and specifically see the algorithm of Collier as "yet another" application of technique of [Luhn, 1957]; to locate in a corpus statistically significant words and use those words in which they occur as key sentences. Yangarber *et al.* however also considers tuples contained in relevant documents to be relevant. In fact, it is almost impossible to create new unsupervised information out of documents without using these kind of statistics. But we think there is no problem in using these statistics because of the large amounts data available in almost any domain. The question is however how to use statistics to acquire knowledge.

6.5 A template responsible for changes in stock price

In the last section we discussed the basic form of our template. In this chapter we will describe how to fill this template and how to adapt it to capture features which are important for influencing the stock price. In this way we hope to get a template which extracts features which are on one hand *generic* enough to appear enough times in our articles to say something meaningful but *specific* enough to be responsible for a rise or drop in stock price.

We will do this with examples taken directly from Dow Jones Newswires and are in this way representative of sentences given in news articles.

6.5.1 How to fill the template?

The first problem which arises is that our template can be filled by each sentence. This leads to enormous amounts of possible fillings which is not only very slow but also lead to many unimportant features. We solve this problem by only triggering our template algorithm if a company is mentioned in the sentence. This technique was previously used by [Seo *et al.*, 2002] to extract important parts of the document as we described in chapter 2 and is based on the fact that a company is almost always mentioned explicitly in stock market articles.

To fill the template for the remaining sentences each sentence is first converted to parsable sentence for our FDG, for example by stripping everything before the colon in the sentence: "DJ MARKET TALK/EU: Corixa and GlaxoSmithKline enter into pact"[4]

As we described in chapter 6.3.1 the template consists of a relation between an agent and a target. Our basic method is to fill them with respectively the *verb* as relation as proposed by [Collier, 1996], the *subject* as agent and the *object* as target. In a simple example as "The manager rewarded the employee", the agent will be manager, the target is employee and the relation is rewarded.

[4]One other not so obvious rule is that all mentions of "oks " are replaced by "approves ", because "to ok" in different forms is a common verb in the stock market and not understood by the FDG

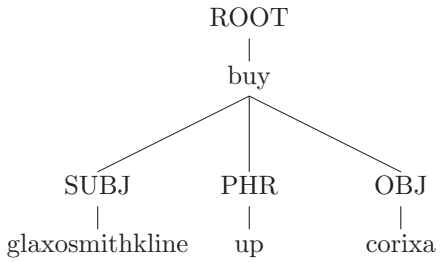

Figure 6.11: FDG tree of the sentence "GlaxoSmithKline buys up Corixa."

Our algorithm extract this information out of the FDG tree, using three steps:

1. Look for a company; for example in figure 6.11 "GlaxoSmithKline".
2. Trace back to the verb; here "buy" (the "up" is later added as described in appendix D)
3. Get the subject and object; here "GlaxoSmithKline" and "Corixa".

However this will not always give the features we want. The precise algorithm in therefore described in appendix D. We will here discuss the most important exceptions to the rule.

For example if no object is found like in the sentence "DJ MARKET TALK: J&J Alzheimer Study Goes Well." Here we have the alternative of taking the *manner* (*well*) as target instead of the object and the final filling will be:

```
Agent:study Relation:goes Target:well
```

Another example is, if there is no subject found for the verb, for example in the sentence "Not even Roche can stop GSK.". In this case we take the verb chain ("not even Roche") and search in it for a subject, here "Roche". The final filling is:

```
Agent:roche Relation:stop Target:gsk
```

Finally, an often appearing construction is when a company says something, for example in the sentence: "CRL said the growth dropped" of which the FDG tree is drawn in figure 6.12. The values according to the tree are:

```
Agent:CRL Relation:say Target:drop
```

This is true according to our subject/object/verb-idea, but we are more interested in what CRL says. We therefore first look at a potential target to see if it is the *main verb in an active verb chain*. If this is the case, like here, we search this verb chain for a construction which matches our template. In this case the extracted values are:

```
Agent:growth Relation:drop Target:significantly
```

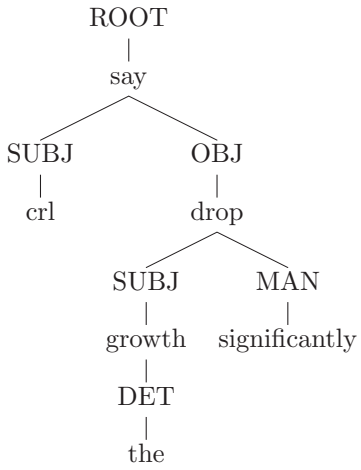

Figure 6.12: FDG tree of the sentence "CRL said the growth dropped significantly."

6.5.2 How to generalize the template?

An important step is to generalize our features to make them generic enough so they appear often enough to be useful. Especially because we work with a general extraction template instead of words or bigrams, there are many possibilities of filling it and it is essential to generalize to reduce this number of possibilities. We have two ways of generalization: word based and context based.

Word based

The word based generalization can be explained using our example in figure 6.11. Here one feature can be extracted: a *buy up* with as agent *GlaxoSmithKline* and as target *Corixa*. But what if instead of *buy up* the sentence was "GlaxoSmithKline buys *out* Corixa" in which *buy up* has approximately the same meaning as *buy out*?

In this case we will use WordNet to create more general features as suggested by [Collier, 1996] by using synsets (see chapter 5.2, chosen using the grammatical information from our FDG[5]) instead of words as part of our features. To deal with the observation of Collier that not for each domain sufficient possibilities of generalization exists using synsets we will combine these synsets with our domain knowledge. Further more we will convert different numbers to a *number tag* as proposed by [Yangarber *et al.*, 2000a].

We will in our experiments use our domain knowledge in a very simple way, based on just one generalizing step and not yet reason about *something which has formal power* as mentioned in the approach of this chapter because we first want to extract reasonable templates with our current system

A concrete example how the feature from figure 6.11 would be stored in the database is seen in the following example[6]

[5]WordNet synsets are different for the different forms it appears in, for example *noun* or *verb*. We can use the information from the FDG to choose the right synset.

[6]Sometimes a wrong WordNet synset is chosen, when it is not the most common group for

Template	Plunge	No Recommendation	Surge
T_1	50	0	50
T_2	25	0	70

Table 6.2: What if negation is not well implemented.

```
Agent:glaxosmithkline

Relation:"take over, buy out, buy up "(WordNet group)

Target:corixa
```

Context based

Next to the generalization of words, it is important to have our features to be generic for the different companies we look at. We will therefore replace a company appearing in a template with a marker. This marker identifies if this company is the company at which we are looking, considering for example the stock price, or another company. In this way we can abstract features from company names while still taking in to account if something is said about the company we are looking at or about another company. We could for example have a feature which says "*This company* buys up *other company*" which could have a negative effect on the stock price. Another feature could then be "*Other company* buys up *this company*" and have a positive effect. These two features could be extracted from the same article (and sentence) which is likely considered relevant for both the two companies. The feature extracted from the FDG tree in figure 6.11, if the sentence appeared in an article about GlaxoSmithKline, now becomes:

```
Agent:ThisCompany

Relation:"take over, buy out, buy up "(WordNet group)

Target:OtherCompany
```

6.5.3 How to deal with negation and favorability?

If negation is not well handled in our algorithm chances are that there will be many templates which will have much influence but are not well dividing between trends because they contain both high values for *Surge* as for *Plunge* as can be seen in figure 6.2. Our FDG parser however can cope with this negation as we see in an analysis of the sentence "Not even Roche can stop GSK." in figure 6.13 by the use of a negation tag (NEG).

We can now implement a function to look for negation tags which depend on the relation (verb[7]) which rates a sentence if it is true or false.

Another construction which can alter the meaning of a sentence are positive and negative triggers such as *successful* or *disastrous*. For example in the

a word. Research must be done to automatically decide which synset to take.

[7]These words must not be main verbs in an active verb chain (unless it is a verb chain) otherwise the negation is not of the verb we are looking at.

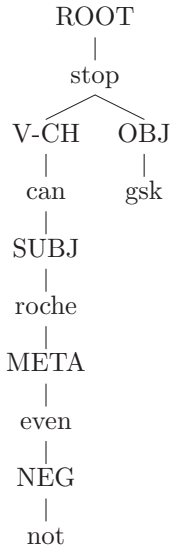

Figure 6.13: FDG tree of the sentence "Not even Roche can stop GSK."

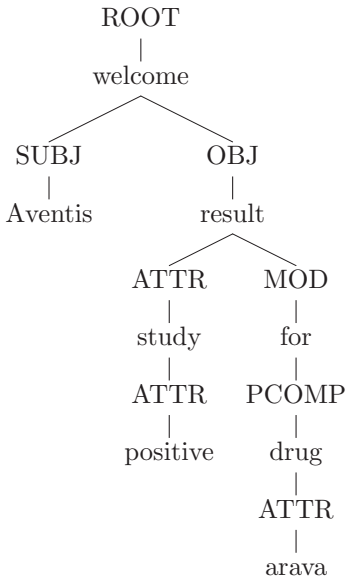

Figure 6.14: FDG tree of the sentence "Aventis welcomes positive study results for Arava drug."

sentence: "Aventis welcomes positive study results for Arava drug". Here we use our domain knowledge to reason about positive and negative words such as *positive* and look if such a word depends on the target. In the example, as seen in figure 6.14, the word positive depends on the target "result".

To represent these constructions we extend our template in the following way:

```
<AGENT> <RELATION,VALUE> <TARGET,VALUE>
```

where the value of the relation depicts any negations and the value of the target depicts positive or negative triggers. The template filling for the Aventis sentence now becomes:

```
Agent:aventis Relation:welcome,10 Target:result,10
```

and for the Roche sentence:

```
Agent:roche Relation:stop,-10 Target:gsk,0
```

And finally when we consider abstraction from the companies and consider the second sentence to be in an article about Roche we get the following features:

```
Agent:This company Relation:welcome,10 Target:result,10
```

```
Agent:This company Relation:stop,-10 Target:Other company,0
```

6.5.4 Other problems

Sometimes the Connexor FDG does not find complete parse trees. [Mollá and Hutchinson, 2002] solved this problem by taking a bottom up approach and by introducing dummy subjects. For example consider the case that the FDG tree of the part of the sentence "ate cakes" is disconnected from the FDG tree of rest of the sentence. In this case a dummy subject is created for "ate cakes". In the end those dummy subjects are matched against subjects in the other trees to extract logical forms. Most disconnected trees are however due to syntactic ambiguities in the sentence. We therefore propose an algorithm in which our program can decide on the best possible solution by adding syntactic information to words. This solution will be implemented in the next version of the Connexor FDG.

6.6 Summary

In this chapter we looked at a more extensive approach for extracting features. A template with the form

```
<AGENT> <RELATION,VALUE> <TARGET,VALUE>
```

was defined with an algorithm to extract it out of sentences. This template

- Can deal with negation, using the information provided by the Connexor FDG.
- Can deal with favorability using the combination of positive and negative triggers with our dependency trees.

- Abstracts from companies using background knowledge.
- Uses WordNet and domain knowledge for generalization.

The system as a result of the use of these features can be seen in figure 6.15.

Because our template filling is unsupervised and extracts relatively general features it has the advantage over templates defined by experts, such as in [Khoo *et al.*, 1999], or by users, as in [Ciravegna and Petrelli, 2001], that it can be used in a much broader domain. It is especially worthwhile to look at the use of our template in other areas of favorability analysis for example the sentiment towards a certain product as described by [Nasukawa and Yi, 2003].

Of the many talks with experts one thing was clear. There are so many exceptions in the textual construction of news articles, if we consider for example a template which is able to extract take overs, that if we take every exception into account we probably would have built a system which completely understands every document. This is surely too much to expect from this project. But at least we can make a small step into that direction.

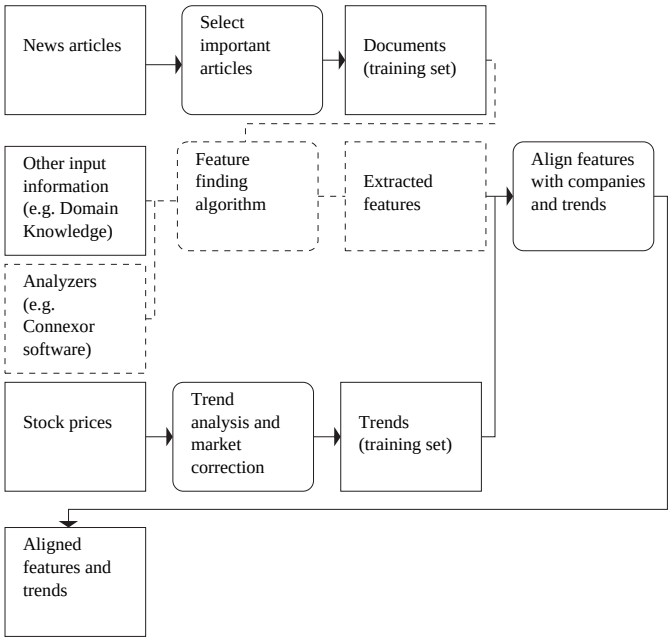

Figure 6.15: Contribution of template based features to our system.

Chapter 7

Constructing a model for text classification

7.1 Introduction

The other important design choice we described in chapter 2, next to to the choice of features, is the classification method. Especially since we argued in the last chapter that our system determines the important features based on this method. This means that our feature extraction algorithm simply extracts all features with the conditions defined in chapter 6.5 and our classification takes care of giving the most influence to the most discriminating features.

In this chapter we will compare different methods for text classification and make clear which design choices where made by each measure. We will compare these measures based on unigram modeling but we will make clear how the algorithm we chose to implement can be used to classify on other features. Finally we choose one method for our system.

What is often seen in articles about the analysis of documents is that they introduce new classification measures, like in [Cho, 1999] or [Xu *et al.*, 2002], which are often not compared to existing ones and at least not proven in practice. We think that improvements in the field of new classification measures for documents will not lead to significant better results, and that most improvement can be gained by improving the quality of the input features of which we have given examples in chapter 5 and 6. This improvement of features can of course not be done without choosing a good existing text classification method and showing that it can be used for more extensive features.

This is what we will do in this chapter.

7.2 TFIDF

The document frequency $DF(w_i)$ of a term w_i is defined by the number of documents this term occurs in. A term with a low document frequency is more specific than a term with high document frequency. [Sparck-Jones, 1972] suggested that therefore a system for information retrieval should treat matches on non-frequent terms as more valuable than ones on frequent terms. This

eventually led to the *inverse document frequency* or *IDF*[1] as a measure of the importance of a term as given in equation 7.1.

$$IDF(w_i) = log(\frac{|D|}{DF(w_i)}) \tag{7.1}$$

In equation (7.1) $|D|$ stands for the total number of documents. In the Smart retrieval system [Salton and McGill, 1983] it was suggested to combine the inverse document frequency with the term frequency $TF(w_i, d)$ of a term w_i in a document d, leading to the so called *TFIDF*. Where the weight $d^{(i)}$ of word w_i in document d is calculated as follows:

$$d^{(i)} = TF(w_i, d) \cdot IDF(w_i) \tag{7.2}$$

In this way we can for each document construct a vector $\vec{d}$, which we can see in equation 7.3, where n is the total number of features in all training documents.

$$\vec{d} = (d^{(0)}, d^{(1)}, ..., d^{(n)}) \tag{7.3}$$

To classify new documents, document vectors can be compared to a prototype vector for each category. This vector is constructed by taking for each term positive and negative examples. In this thesis we will call classifiers which use these prototype vectors, *prototype vector based* classifiers. Because in this thesis we will classify on different types of trends (*surge*, *plunge* and *not relevant*) we can from the set of training documents for each trend construct a category vector. A category vector for trend t can then be constructed as seen in (7.4).

$$\vec{c_t} = \beta(\frac{1}{|C_t|} \sum_{\vec{d} \in C_t} \frac{\vec{d}}{||\vec{d}||}) - \gamma(\frac{1}{|D| - |C_t|} \sum_{\vec{d} \in D - C_t} \frac{\vec{d}}{||\vec{d}||}) \tag{7.4}$$

β and γ are parameters that adjust the relative impact of positive and negative training examples and are suggested by [Buckley *et al.*, 1994] to be set at 16 and 4 respectively. D is the set of al training documents and C_t is the set of training documents assigned to category t, of which $|C_t|$ is the amount. Finally, $||\vec{d}||$ is the *Euclidean length*[2] of the vector $\vec{d}$.

The resulting set of prototype vectors, one for each category, represents the learned model. This model can be used to classify a new document d' by using a distance measure such as the cosines measure and assign the document to the category with which its document vector with the lowest distance or, if we use the cosines measure, has the highest cosine. This is formalized in (7.5).

$$H_{TFIDF}(d') = \arg\max_{t \in trends} \cos(\vec{c_t}, \vec{d'}) \tag{7.5}$$

In the TFIDF algorithm there are several design choices:

The choice for β and γ Normally 16 and 4.

[1] In [Papineni, 2001] it is showed that the IDF is the optimal weight associated with a word-feature in an information retrieval setting where we treat each document as the query that retrieves itself. That is, IDF is optimal for document *self retrieval*.

[2] Euclidean length of vector v with length N is $||\vec{v}|| = \sqrt{\sum_{i=1}^{N} v_i^2}$

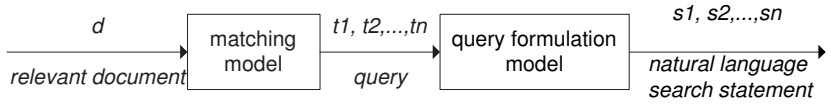

Figure 7.1: Model of matching and query formulation.

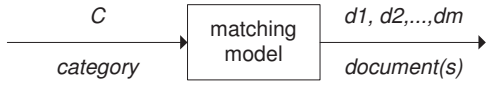

Figure 7.2: Adapted model of matching.

The word weighting method in a document vector For example the term frequency or the term frequency times the inverse document frequency.

The normalization of the document vector For example document normalization using Euclidian vector length or no normalization at all.

The similarity measure between document and category vectors For example cosine similarity or the Euclidean distance.

In chapter 7.4 we will discuss an implementation of these choices by [Joachims, 1997] based on a probabilistic approach as seen in the next chapter.

7.3 Naive Bayes

Another approach of calculating the best category match for a document comes from the information theory and is based on a probabilistic model of a document. We can explain this approach on the basis of figure 7.1 which we adapted from [Hiemstra, 2001]. From this viewpoint, a relevant document d gets corrupted into a query $t_1, ..., t_n$ by sending it through a noisy channel, and the query gets again corrupted into a request by a second noisy channel. A natural language information retrieval system which has to find a document which is relevant for the request can be thought of as a decoding function. $f : s_1, ..., s_n \rightarrow d$, that tries to reproduce the message that was originally sent based on the query terms.

We can adapt this model to our problem by arguing that a certain category C of news articles leading to a certain trend gets corrupted producing one or several documents as seen in figure 7.2. The retrieval system can now be thought of as a decoding function which given a set of documents produces the category of articles which produced this document. This leads for one document to equation (7.6) and for a set of stories $d'_1...d'_m$ to equation (7.7) where $H_{BAYES}(d')$ and $H_{LAVRENKO}(d'_1...d'_m)$ are the functions which define to which category a document d' or documents $d'_1...d'_m$ are assigned given their vector representation.

$$H_{BAYES}(d') = \underset{t \in trends}{\arg\max} Pr(C_t|d') \tag{7.6}$$

$$H_{LAVRENKO}(d'_1...d'_m) = \arg\max_{t \in trends} Pr(C_t|\{d'_1...d'_m\}) \quad (7.7)$$

$$= \arg\max_{t \in trends} \frac{Pr(\{d'_1...d'_m\}|C_t)Pr(C_t)}{Pr(\{d'_1...d'_m\})} \quad (7.8)$$

$$= \arg\max_{t \in trends} Pr(\{d'_1...d'_m\}|C_t)Pr(C_t) \quad (7.9)$$

Equation (7.7) comes from [Lavrenko *et al.*, 2000]. They assume a uniform prior for each trend, so $Pr(C_t)$ can be removed. Furthermore they estimate the prior $Pr(\{d'_1...d'_m\})$ as the probability that the set of documents was generated by random words of *General English* (represented by GE). If it is also assumed that the different documents are random samples from a common distribution and not dependent on each other this leads to equation (7.10).

$$H_{LAVRENKO}(d'_1...d'_m) = \arg\max_{t \in trends} \prod_{j=1}^{m} \frac{Pr(d_j|C_t)}{Pr(d_j|GE)} \quad (7.10)$$

The other approach which does not use the uniform prior or General English comes from [Joachims, 1997]. What both approaches have in common is that they assume that the occurrence of a word in a document is only dependent on the category the document is in but that it occurs independently of the other words in the document, which means that they look at a document as a *bag of words* where the order is not considered. Because of this determining $Pr(d'|C_t)$ comes down to determining $\prod_{w \in F} Pr(w|C_t)$[3]. This leads to the following calculation formulas:

$$H_{BAYES}(d') = \arg\max_{t \in trends} \frac{Pr(C_t) \cdot \prod_{w \in F} Pr(w|C_t)}{\sum_{u \in trends} Pr(C'_u) \cdot \prod_{w' \in F} Pr(w'|C'_u)} \quad (7.11)$$

$$= \arg\max_{t \in trends} Pr(C_t) \cdot \prod_{w \in F} Pr(w|C_j) \quad (7.12)$$

$$H_{LAVRENKO}(d'_1...d'_m) = \arg\max_{t \in trends} \prod_{j=1}^{m} \prod_{i=1}^{|d'_j|} \frac{P(w_i|C_t)}{Pr(w_i|GE)} \quad (7.13)$$

Because this *bag of words* assumption is not verified in practice and if we remove the *argmax* part in equation (7.11) we get an application of Bayes' rule[4] these classifiers are called *naive Bayes*.

The last step we still need to take is to estimate $Pr(w_i|C_j)$. Next to the use of a uniform prior of $Pr(C'_t)$ by Lavrenko *et al.* and the use of General English as prior for $Pr(\{d'_1...d'_m\})$ the third difference between the two approaches lie in their solution for the zero-frequency problem; what happens if a feature does not occur in every category vector and $P(w|C_t)$ and therefore also $P(d'|C_t)$ would be zero? This problem can be solved using *smoothing*. Where by Lavrenko *et*

[3] We will further discuss this in chapter 7.5.

[4] Namely

$$Pr(c_t|d_j) = \frac{P(d_j|c_t)P(c_t)}{P(d_j)} \quad (7.14)$$

al. a so called linear backoff is used which can be seen in equation (7.18)[5], Joachims solves this problem by adding 1 to the count for every descriptor-category, divided by the number of words. This is worked out in equation (7.19) where $|F|$ is total number of words in d'. Other smoothing methods can be found in [Ordelman, 2003].

$$Pr_{ml}(w_i|C_t) = \frac{TF(w_i, C_t)}{\sum_{w' \in F} TF(w', C_j)} \tag{7.17}$$

$$\hat{Pr}_{LAVRENKO}(w_i|C_t) = \lambda_t Pr_{ml}(w_i|C_t) + (1 - \lambda_t) Pr(w_i|GE) \tag{7.18}$$

$$\hat{Pr}_{BAYES}(w_i|C_t) = \frac{1 + TF(w_i, C_t)}{|F| + \sum_{w' \in F} TF(w', C_t)} \tag{7.19}$$

7.4 PrTFIDF

In the PrTFIDF algorithm proposed by Joachims one of the new introduced ideas is, instead of making the word independence assumption, to introduce a new descriptor x for a document. This descriptor is assigned to a document with the probability $Pr(x|d)$, where X is the total set of descriptors for a document. Using the theorem of total probability[6] we can write

$$Pr(C_t|d) = \sum_{x \in X} Pr(C_t|x, d) \cdot Pr(x|d) \tag{7.20}$$

Using Bayes' rule[7] we can rewrite this to:

$$Pr(C_t|d) = \sum_{x \in X} \frac{Pr(d|C_t, x)}{Pr(d|x)} Pr(C_t|x) \cdot Pr(x|d) \tag{7.21}$$

Given that the descriptor x provides enough information about a document that the category it is in will not add any extra information, formally $Pr(d|C_t, x) = Pr(d|x)$, equation (7.21) can be rewritten as:

$$Pr(C_t|d) \approx \sum_{x \in X} Pr(C_t|x) \cdot Pr(x|d) \tag{7.22}$$

As classifier x Joachims proposes a bag of n words drawn randomly from document d. The underlying assumption for this is that documents are highly redundant considering the classification task and that any sequence of words is equally sufficient to determine the category of a document. As an example

[5]For completeness the English language, especially in the stock market, can also change over time. For this reason the general English has a backoff as well:

$$\hat{Pr}(w_i|GE) = \lambda_{GE} Pr_{ml}(w_i|GE) + (1 - \lambda_{GE})/N_{GE} \tag{7.15}$$

$$Pr_{ml}(w|GE) = \frac{Pr(w_i, GE)}{\sum_{w' \in GE} TF(w', GE)} \tag{7.16}$$

Where N_{GE} is the total number of tokens in general English.

[6]Theorem of total probability: Given n mutually exclusive events $A_1, ..., A_n$ which are a partition, $Pr(B) = \sum_{i=1}^{n} Pr(B|A_i)Pr(A_i)$, where B is an arbitrary event.

[7]Bayes' rule: $Pr(Y|X) = \frac{Pr(X|Y)Pr(Y)}{Pr(X)}$

consider documents about religion or coffee where any sequence of n words would probably be equally sufficient for classification[8]. If we use for n the total number of words in the document $Pr(C_t|d)$ equals $Pr(C_t|x)$. With decreasing n this simplifying assumption will be violated in practice, but it can be used as a starting point. For $n = 1$ we will arrive at a TFIDF classifier, where equation (7.22) can be written as:

$$Pr(C_t|d) \approx \sum_{w \in F} Pr(C_t|w) \cdot Pr(w|d) \tag{7.23}$$

where F is the set of features (words) found in trend t and $Pr(w|d)$ can be estimated as:

$$\hat{Pr}(w|d) = \frac{TF(w,d)}{|d|} \tag{7.24}$$

$Pr(C_t|w)$ can be rewritten using Bayes' rule to:

$$Pr(C_t|w) = \frac{Pr(w|C_t) \cdot Pr(C_t)}{\sum_{u \in trends} Pr(w|C_u) \cdot Pr(C_u)} \tag{7.25}$$

As in the naive Bayes implementation C_t is also estimated as the number of documents in C_t divided through the total amount of documents. The chance a category leads to a certain word, $Pr(w|C_j)$, can be estimated as the mean of the probabilities of the different documents generating this word in the respective category; $\frac{1}{|C_t|} \sum_{d \in C_t} Pr(w|d)$. This eventually leads to the following decision rule:

$$\begin{aligned} & H_{PrTFIDF}(d') \\ &= \arg\max_{t \in trends} \sum_{w \in F} \frac{Pr(w|C_t) \cdot Pr(C_t)}{\sum_{u \in trends} Pr(w|C_u) \cdot Pr(C_u)} \cdot Pr(w|d') && (7.26) \\ &= \arg\max_{t \in trends} \sum_{w \in F} Pr(w|C_t) \cdot Pr(C_t) \cdot Pr(w|d') && (7.27) \end{aligned}$$

The nice property of this rule is, that it can easily be converted to the shape of the TFIDF classifier with adapted IDF, DF, $d^{(i)}$ and $\vec{c_t}$. The proof for this is given in Joachims, we will only give the adapted formula's:

$$IDF'(w) = \sqrt{\frac{|D|}{DF'(w)}} \tag{7.28}$$

$$DF'(w) = \sum_{d \in D} \frac{TF(w,d)}{|d|} \tag{7.29}$$

$$H_{PrTFIDF}(d') = \arg\max_{t \in trends} \frac{\vec{c_t}}{1} \cdot \frac{\vec{d'}}{|d'|} \tag{7.30}$$

$$d^{(i)} = TF(w_i, d) \cdot IDF'(w_i) \tag{7.31}$$

[8]This is not the case for the description of articles specific for different trends but using the features described in chapter 6 and implemented in chapter 7.5 we can argue this holds for these news articles as well.

$$\vec{c_t} = \frac{|C_t|}{|D|} \cdot \frac{1}{|C_t|} \cdot \sum_{d \in C_t} \frac{\vec{d}}{|d|} \tag{7.32}$$

In equation (7.30) the inner product is taken and equation (7.32) is not simplified so we can compared it with the TFIDF algorithm in equation (7.4). When we compare PrTFIDF with TFIDF we can see the following design choices:

The choice for β and γ In PrTFIDF prior probabilities $Pr(C_t)$ are incorporated through β, which is $\frac{|C_t|}{|D|}$, furthermore $\vec{c_t}$ does not depend on negative examples, which means $\gamma = 0$.

The word weighting method in a document vector PrTFIDF uses $IDF'(w)$ for word weighting.

The normalization of the document vector PrTFIDF uses the number of words for document normalization.

The similarity measure between document and category vectors PrTFIDF uses the inner product for computing similarity.

We have implemented this algorithm in our system, but removed the a priori chance for a trend $Pr(C_t)$ here estimated by $\frac{|C_t|}{|D|}$. We do this because of three reasons:

1. As we saw in chapter 4 it is assumed that trends do not have an a priori chance.
2. Even if trends have an a priori chance, because most articles do not have a great influence, the best classification method would therefore be to always a priory classify an article as being *not relevant*, which is not interesting for traders.
3. Finally we cannot guarantee that our algorithm is trained during a representative period considering the market. For example if the training period was in a time where the price dropped most of the time because. Our system would be biased towards a negative trend and would not function when the financial climate is better.

7.5 Adaption to more extensive features

To estimate the importance of features which contain more information, such as our templates, we can still use the PrTFIDF algorithm, however we must make some adjustments. Most important we can still use equation (7.22), since these descriptors will provide more information than single word descriptors. In this way the assumption which was made by PrTFIDF, that the category a document is in will not add extra information about a document when the descriptor is known, is even more reasonable. We can even argue that the PrTFIDF algorithm works better with these features because of this. Now we can rewrite (7.23) as:

$$Pr(C_t|d) \approx \sum_{f \in F} Pr(C_t|f) \cdot Pr(f|d) \tag{7.33}$$

where f is a certain feature. $Pr(C_t|f)$ can be estimated as:

$$\hat{Pr}(f|d) = \frac{Freq(f,d)}{\sum_{f' \in F} Freq(f',d)} \tag{7.34}$$

Compared to equation (7.24), TF, which stood for the term frequency, is replaced by the frequency of a certain feature f and the number of words in a document, $|d|$, is replaced by the total frequency of all existing features in d.

The other equations are only dependent on the choice of the descriptor x via this estimation, especially $\vec{Pr}(f|C_j)$ which is equal to $\frac{1}{|C_t|}\sum_{d \in C_t} \vec{Pr}(f|d)$. This leads to the following decision rule:

$$H_{PrTFIDFFeature}(d') = \arg\max_{t \in trends} \sum_{f \in F} Pr(f|C_t) \cdot Pr(C_t) \cdot Pr(f|d') \tag{7.35}$$

which can be rewritten in TFIDF form. For completeness we list all formula's here:

$$IDF'(f) = \sqrt{\frac{|D|}{DF'(f)}} \tag{7.36}$$

$$DF'(f) = \sum_{d \in D} \frac{Freq(f,d)}{\sum_{f' \in F} Freq(f',d)} \tag{7.37}$$

$$H_{PrTFIDF}(d') = \arg\max_{t \in trends} \frac{\vec{c_t}}{1} \cdot \frac{\vec{d'}}{Freq(f',d)} \tag{7.38}$$

$$d^{(i)} = Freq(f_i,d) \cdot IDF'(f_i) \tag{7.39}$$

$$\vec{c_t} = \frac{|C_t|}{|D|} \cdot \frac{1}{|C_t|} \cdot \sum_{d \in C_t} \frac{\vec{d}}{Freq(f',d)} \tag{7.40}$$

7.6 Support Vector Machines

7.6.1 In general

The drawback of the prototype vector based PrTFIDF, and also of the Naive Bayes approach as proposed by Lavrenko *et al.* is that they classify linearly. An example where this can go wrong can be seen in figure 7.3. Suppose documents with a either low or high occurrence of both words "aap" and "bakker" tend to be indicators of a positive trend, while if only one of the two features occurs in a document it will probably indicate a negative trend. Then because of these properties $Pr(w|C_j)$ in equation (7.13) and (7.11) for both words "aap" and "bakker" will be equal for the positive and the negative category and the classifier will not be able to make a choice based on these words.

This contrary to for example a k-nearest neighbor classifier[9] (k-NN). In figure 7.4 we see how a k-NN classifier would rightly identify the different clusters in

[9]k-nearest neighbor classifier: This classifier labels an unknown object O with the label of the majority of the k nearest neighbors. A neighbor is nearest if it has the smallest distance in feature space. For $k = 1$, this is the label of its closest neighbor in the training set.

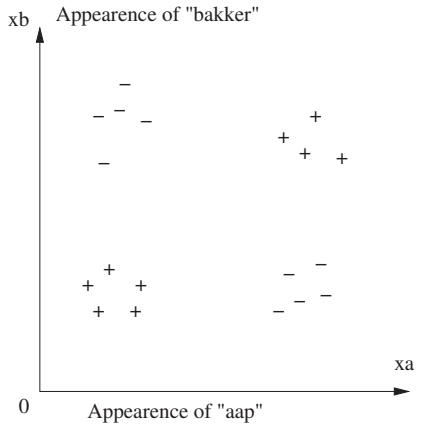

Figure 7.3: The aap and bakker example.

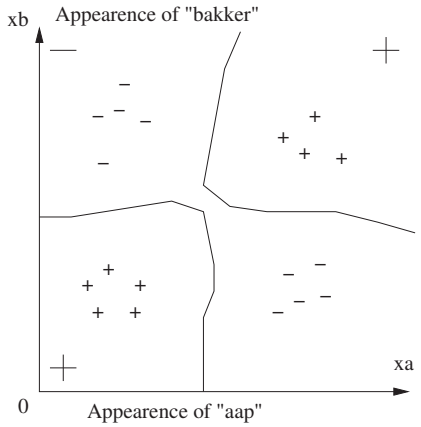

Figure 7.4: k-nearest neighbor classifier for the aap and bakker example.

the "aap" and "bakker" example. This good performance of k-NN is confirmed by [Yang, 1999] who compares fourteen text classification methods on their break even point[10]. The most important drawback from k-NN is however that because it is based on comparison with the training examples it is very inefficient at classification time.

In this chapter we will discuss Support Vector Machines (SVMs). A classifier which can solve the aap and bakker example but do not need comparison with all training examples. SVMs were developed by [Vapnik, 1982] and date back to 1965.

Fisher suggested in 1936 the first algorithm for pattern recognition. He considered two normal distributed populations of n dimensional vectors and showed that the optimal solution to separate them was a quadratic decision function with $\frac{n(n+3)}{2}$ free parameters. In case the number of observations is

[10]Break-even point: Point where precision and recall are tuned to being equal, for more on evaluation measures see chapter 8.

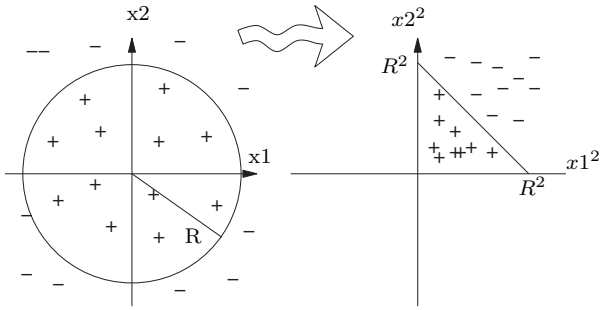

Figure 7.5: To another space.

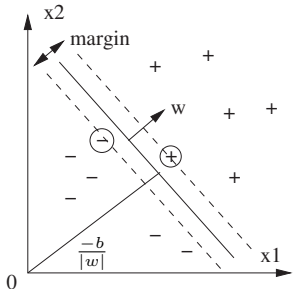

Figure 7.6: A linear separable problem.

too small, for example $10n^2$, estimating these $o(n^2)$ parameters is not reliable. Fisher therefore recommended to use a simplified linear discriminator.

A Support Vector Machine implements the following idea: it maps the input vectors into some high dimensional feature space Z through a non-linear mapping which is chosen a priori. In this high dimensional space a linear decision surface or hyperplane is constructed with special properties that ensure high generalization. An example of such a mapping can be seen in figure 7.5[11]. In this way we can effectively use Fischer's linear discriminator for non linear separable problems.

Now if we can map features into another space (*feature space*) to create a linear separable problem two problems arise. First, how can we find a separating hyperplane in this huge dimensional feature space that will generalize well. And, secondly, how to construct hyperplanes in this huge dimensional feature space.

As a solution to the first problem, SVMs find for a linear separable problem the hyperplane with the maximum Euclidian distance to the closest training examples which are called the *support vectors* and are circled in figure 7.6. The sum of these two distances is called the *margin* and by maximizing this margin our hyperplane can generalize well. If we pick the margin to be 1, we can write the margin in terms of w as $\frac{2}{||w||}$. The goal is now to to get an equation for the hyperplane by determining w and b. We can incorporate b into w which leads to

[11] In the same way we could transform the input vectors of figure 7.3 if we lay the origin in the middle of the four groups and add an extra attribute $xc = xa \times xb$.

a $\hat{w}$. This $\hat{w}$ can be calculated as a linear combination of the data points as[12]:

$$\hat{w} = \sum_{j=1}^{m} \alpha_j y_j \vec{x}_j \tag{7.41}$$

The input data for a SVM consists of m training data points of the form $\vec{x}$ with their classification y, which is either -1 or 1, this finding of the maximum of the margin eventually leads to the following problem:

Find $\max_a L(\vec{a})$ subject to $\sum_i \alpha_i y_i = 0$ and $C \geq \alpha_i \geq 0, \forall i$ where

$$L(\vec{a}) = \sum_{j=1}^{m} \alpha_j - \frac{1}{2} \sum_{j=1}^{m} \sum_{k=1}^{m} \alpha_j \alpha_k y_j y_k (\vec{x}_j \cdot \vec{x}_k) \tag{7.42}$$

Here $\vec{x}_j$,$\vec{x}_k$,y_j and y_k are known from the training set and C is a factor which allows trading of training error versus complexity. A higher C means that more complexity (higher variance) is allowed therefore allowing less errors on the training set (lower bias). If there are training errors we speak of a *soft margin* because the hyperplane will not completely separate the training examples. Equation (7.42) is known as a *quadratic programming problem*. This is a well studied form of optimization for which there exist good optimization algorithms. And, also important, it is known that problems of this kind have global optima.

Classification of an unknown vector $\vec{u}$ can for an SVM with k support vectors be done by calculating:

$$h(\vec{u}) = \text{sign}(\sum_{i=1}^{k} \alpha_i y_i (\vec{x}_i \cdot \vec{u}) + b) \tag{7.43}$$

Equation (7.42) and (7.43) lead to the solution for the second problem. This because in these expressions the data points only appear as dot products with other data points in the huge dimensional feature space, so in fact, all we need to know about the data is the set of m^2 scalar values for these dot products.

Here Cortes and Vapnik introduces kernel functions as in equation (7.44) which do exactly this; given two vectors these functions give the dot product in a certain feature space.

$$\Phi(\vec{x}_j) \cdot \Phi(\vec{x}_k) = K(\vec{x}_j, \vec{x}_k) \tag{7.44}$$

The interesting part is that these functions already exist. As an example we give the radial basis kernel:

$$K(\vec{x}_j, \vec{x}_k) = e^{\frac{-|\vec{x}_j - \vec{x}_k|^2}{2\sigma^2}} \tag{7.45}$$

With this kernel function we get a classifier based on the sum of Gaussian bumps centered on support vectors with σ as standard deviation of which we can see a visualization in figure 7.7 and which is quite as powerful as k-NN[13]

[12]Here α is the number of times the data point is classified incorrectly when using the perceptron algorithm.

[13]A difference between k-NN and a radial based kernel is that k-NN labels a new sample with the label of the (weighted) majority of the k nearest neighbors and generalize after it has seen a test example, which makes it slow at classification time. The radial base function generalizes before seeing a test example so will perform faster.

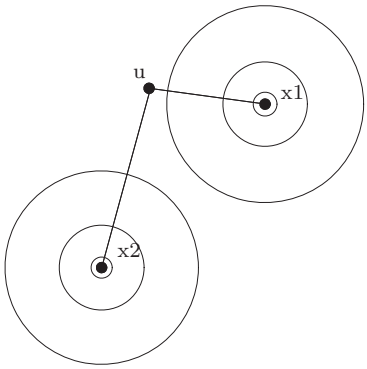

Figure 7.7: A visualization of a radial based kernel.

Next to this radial based kernel, we have a linear and a polynomial kernel as the most standard kernels. The choice hereof depends on the feature space but most of the time simply all kernels are tried with different parameters and the one which performs best is chosen.

A problem with such an approach could be similar to k-NN, namely the risk of having a high variance and be extremely sensitive to overfitting. Luckily SVMs only depend on support vectors, which we could reduce in the radial base form, for example by choosing a larger sigma.

7.6.2 For text classification with templates

[Joachims, 1998] reports several reasons why SVMs are well suitable for text classification. We introduce three of them here and reason why they also hold and maybe even better for more extensive features.

High dimensional input space When learning text classifiers we have to deal with many features. Joachims reported more than 10000, we report for our template method which we described in chapter 6 about 30000 features of which about 2500 appear at least three times. Because SVMs are able to generalize well with this many features, they are well suited for text classification.

Few irrelevant features To avoid high dimensional feature spaces one can often assume most features are irrelevant. However in text classification there are only few irrelevant features. Joachims shows that even if the features with lowest Information Gain are considered, a naive Bayes classifier with these features still performs much better than random. He concludes that it seems unlikely that all those features are completely redundant and therefore aggressive feature selection may result in a loss of information. For high level features this has not yet been tested, but the expectation is that because they extract the core of the sentences they will be even less irrelevant.

Document vectors are sparse For each document, the corresponding document vector, which we saw in equation (7.3), contains only few entries which are not zero. Here [Kivinen and Warmuth, 1995] is quoted who

Table 7.1: Micro averaged performance over all Reuters categories.

Classifier	Result
Bayes	72.0
Rocchio	79.9
C4.5	79.4
k-NN	82.3
SVM, polynomial kernel	86.0
SVM, radial based kernel	86.4

concludes that for mostly relevant features with sparse document vectors "additive" algorithms like perceptrons and SVMs are well suited[14].

Finally Joachims gives empirical evidence of the performance of SVMs compared to several other techniques on the Reuters corpus, of which the totals can be seen in table 7.1.

Important to note is that the weighting of features is still important for text classification, Joachims used TFIDF to acquire his results. Furthermore the weight of features for each document should be normalized.

7.7 Summary and application

In this chapter we gave an extensive overview of different text classification techniques and their relation to each other. Because of the advantages of Support Vector Machines for our domain, as discussed in 7.6.2, we will use them for our classification problem.

Now back to our problem; predicting *surge*, *plunge* or *not relevant* documents. Because a SVM can only predict between two categories we must use two SVMs; one to predict surges and another to predict plunges and use them together as one classifier. The training data we used to train both SVMs can be seen in figure 7.8.

The result of our target function Φ for the resulting classifier, given the results of the two SVMs, is listed in table 7.2.

	SVM_{surge} predicts true	SVM_{surge} predicts false
SVM_{plunge} predicts true	notRelevant	plunge
SVM_{plunge} predicts false	surge	notRelevant

Table 7.2: Prediction of the resulting classifier based on the two SVMs.

We expect our document space based on more extensive features to be better separable with a linear separator than a document space based on words. This because of the fact that our features will for example take into account negation and the difference between "making profit" and "profit expectation". In these examples the words "make" and "expectation" solely mean nothing but their appearance does influence the meaning of profit.

[14]This contrary to a data set with many irrelevant features and dense document vectors, where algorithms which do "multiplicative" updates to their weight vector (like the "Winnow" algorithm) perform better

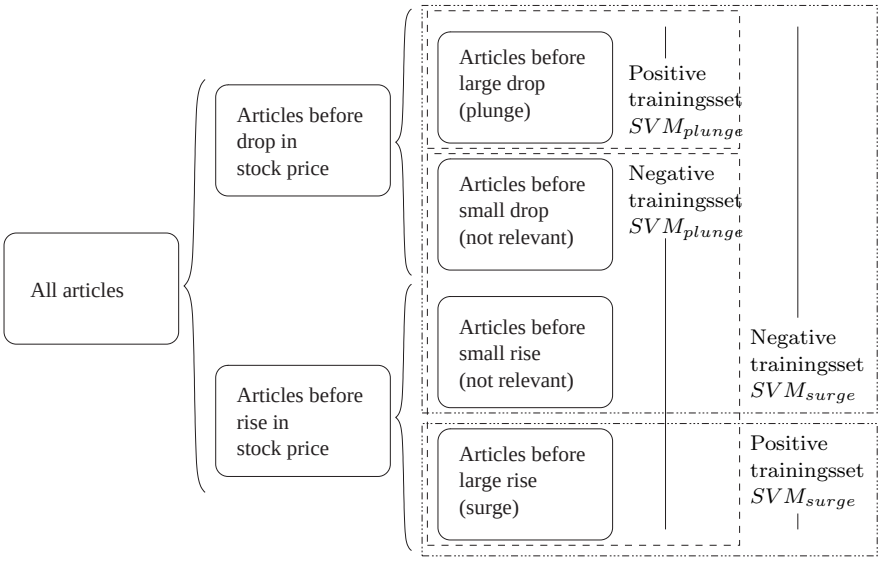

Figure 7.8: How we constructed the training set.

Although better separable, we however expect classifiers who are not based on prototype vectors to do better, because one template can say something about the context of another. If we have for example a template about a company making profit, this will probably have a positive effect on their stock price. If it would however occur together with a template specific to a certain source which is always wrong the meaning would be reversed.

With the incorporation of text classification our system almost takes its final form as can be seen in figure 7.9.

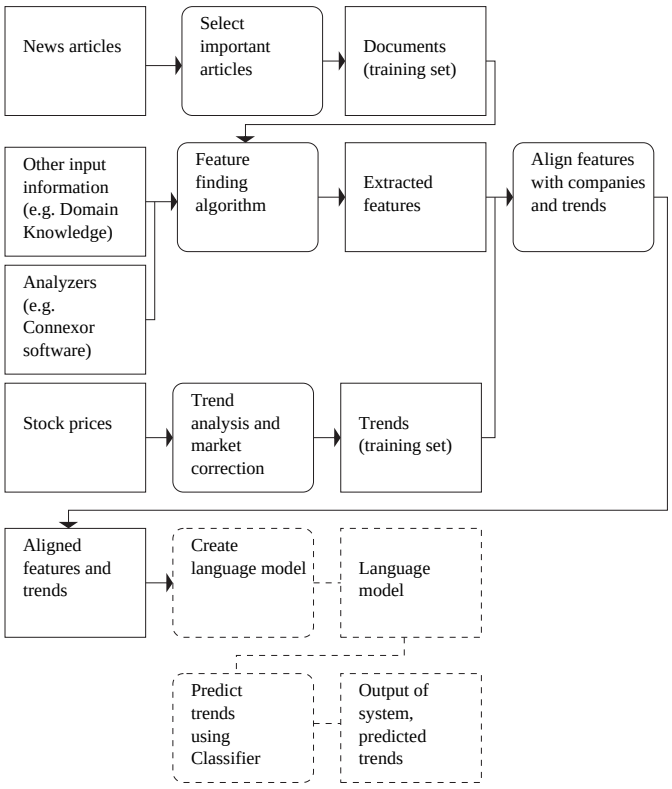

Figure 7.9: Contribution of text classification to our system.

Chapter 8

Evaluation

8.1 Evaluation of the prediction

8.1.1 Considerations

The evaluation is the last part of our system as seen in figure 8.1. The most uniform way to evaluate a system which has to classify something is by using a confusion matrix as described by [Kohavi and Provost, 1998]. This is a matrix showing the predicted and actual classifications. The size of the matrix is $l \times l$ where l is the number of different label values. An example matrix is given in table 8.1.

Other measures can be calculated using this matrix;

Accuracy $(nn + pp)/(nn + np + pn + pp)$

True positive rate (Recall, Sensitivity) $pp/(pp + pn)$

True negative rate (Specificity) $nn/(nn + np)$

Precision $pp/(np + pp)$

False positive rate $np/(nn + np)$

False negative rate $pn/(pn + pp)$

Confusion matrices for all our examples in this chapter are given in appendix E. We must note that we provide only one confusion matrix as seen in figure 8.2 while we in fact have two support vector machines. The reason why we chose to do so is that we want to be able to compare the results of SVMs to other classifiers which can have multiple result classes. To calculate the measures from our confusion matrices we can consider one category at a time and for this category calculate its matrix as seen in table 8.3 for the *not relevant* category.

Actual result	Predicted negative	Predicted positive
Negative	nn	np
Positive	pn	pp

Table 8.1: Example confusion matrix.

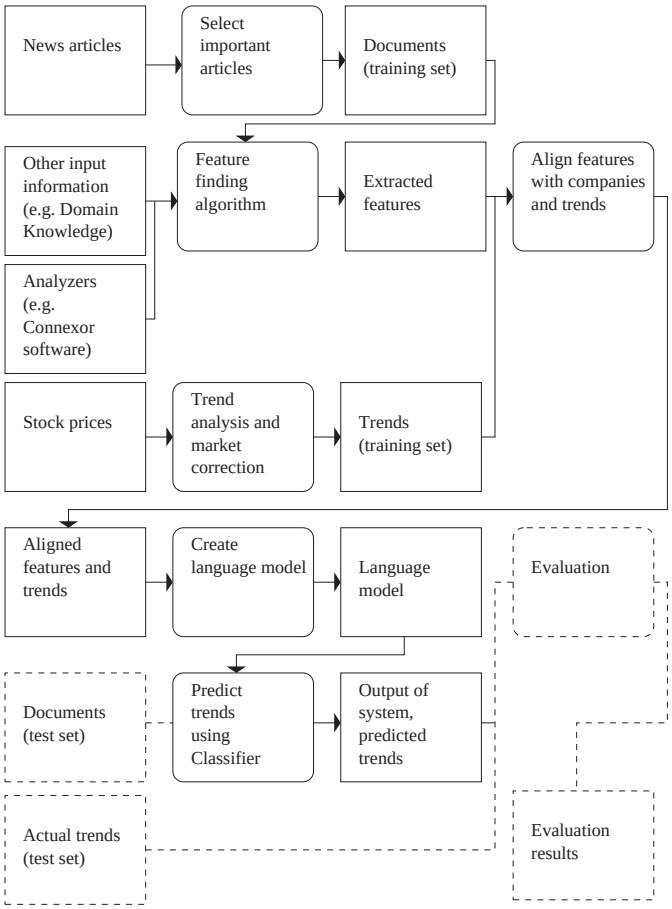

Figure 8.1: Contribution of evaluation to our system.

Actual result	Predicted plunge	Predicted surge	predicted not relevant
Plunge	pp	ps	pn
Surge	sp	ss	sn
Not relevant	np	ns	nn

Table 8.2: Confusion matrix as given in appendix E.

Actual result	Predicted negative	Predicted positive
Negative	$pp + ps + sp + ss$	$pn + sn$
Positive	$ns + np$	nn

Table 8.3: Example confusion matrix for the *not relevant* category.

To evaluate the prediction of the stock market, precision and recall fail to give a clear answer about the goodness of a system compared to a random one. If, for example in a certain period news articles led in 3% of the cases to a *plunge*, a system which has a precision for its *plunges* of 30% and is wrong in 70% of the cases when it predicts a *plunge*, is still better than random because a random system would be wrong in 97% of the cases it predicts a *plunge*.

We will therefore compare the results of the predicted classes, with the original division of the data space. As intuitive result we present for each test set the percentage classified as *surge* and *plunge* in the whole data space and in the parts respectively classified *surge* and *plunge*. If our system works as we hope the percentage of *plunge* in the part classified as *plunge* will rise compared to the whole data space and so will the percentage *surges* in the part classified as *surge*.

As a more statistical viable approach we will for both the plunges and surges for each test also give the 80% and 95% confidence intervals for the percentages as described by [Mitchell, 1997].

8.1.2 About the data set

An interesting point about our research is the use of a unique new data set, namely 12 months of pharmaceutical news articles (about 14.000 documents for 23 companies) and 12 months of pharmaceutical stock prices.

Contrary to standard data sets described by [Fürnkranz, 1998], where in the REUTERS newswire data set unigrams are often enough to classify many of the articles, or the Ken Lang's 20-newsgroup data, where the many quotations lead to a over-optimistic accuracy estimates for n-grams, our data set contains almost no quotations and the classification is much more dependent on relations between words as we described in chapter 6.1.

8.1.3 The tests

For the final system we used SVMs, trained as described in chapter 7.7 where we chose as a *surge* the 50% most steep positive trends and as *plunge* the 50% most steep negative trends. As we noted already in chapter 4, we only have closing prices available. Because of this and because the main reason for trend analysis is to compensate for the first undefined reactions of the stock price (see chapter 4.4) we define the difference between two closing prices on two consecutive days as a trend. The articles which are aligned with each trend are exactly those articles which appear between those days; from the closing time of the stock market on the first day till the closing time of the last.

As parameters for the SVM we weighted all features with IDF' (see chapter 7.5, equation (7.36)), used only features with a minimum appearance of 3 (see chapter 5.2) and used a radial based kernel with $\gamma = 1$ (see chapter 7.6.1) .

What we want to show is, how well our system performs *predicting* the influence of news articles. We therefore chose as training data the first months of our data, starting at 18 December 2002, till the starting date of our test set. The data sets can be found in table 8.1.3.

We conducted the following tests on this data:

The reference test The test described above over all data sets, used to see if we are able to usefully classify articles.

Id	Training set	Test set
A	18 December 2002 till 1 April 2002	2 April 2003 till 1 June 2003
B	18 December 2002 till 1 June 2003	2 June 2003 till 1 August 2003
C	18 December 2002 till 1 August 2003	2 August 2003 till 1 September 2003
D	18 December 2002 till 1 September 2003	2 September 2003 till 1 October 2003
E	18 December 2002 till 1 October 2003	2 October 2003 till 1 November 2003
F	18 December 2002 till 1 November 2003	2 November 2003 till 1 December 2003
G	18 December 2002 till 1 December 2003	2 December 2003 till 1 January 2004

Table 8.4: Data sets.

Selected articles What if we select articles which we suspect to have a more immediate influence? We constructed a set of articles for which we exclude press releases and some market reflections and tested if it they can be used for better predictions.

Several test sets or one We suspect prediction is better when take into account the latest developments, what is the difference in performance if we test all data from 2 April 2003 till 1 January 2004 in one time, trained with the data before 1 April 2002?

Market correction In chapter 4.4 we proposed market correction to improve performance, are the results of our system improved using the parameters as described in this chapter?

Headlines In chapter 2 we suggested that only looking at headlines could improve the results. Is this the case?

8.1.4 Results

We will here only give the main results, details in form of confusion matrices can be found in chapter E.

Data set	# Doc.	Surge org.	Surge class.	Plunge org.	Plunge class.
A	1577	37.66%	48.12%	31.52%	35.29%
B	1770	39.04%	49.44%	33.22%	27.38%
C	623	13.16%	13.33%	37.40%	56.25%
D	1014	22.78%	36%	39.55%	43.94%
E	1343	27.77%	32.89%	38.05%	43.66%
F	784	37.37%	46.15%	31.63%	39.06%
G	763	34.21%	45.83%	25.43%	21.62%
Total	7874	32.07%	40.67%	33.93%	37.28%

Table 8.5: The reference test.

Data set	# Doc.	Surge org.	Surge class.	Plunge org.	Plunge class.
A	1147	39.06%	46.67%	31.39%	35.71%
B	1278	39.44%	47.46%	33.02%	30%
C	466	13.52%	12.5%	36.91%	66.66%
D	776	22.68%	24.39%	39.69%	35.29%
E	1019	28.46%	33.78%	38.17%	52%
F	584	32.87%	31.25%	33.56%	62.16%
G	573	35.95%	60%	23.91%	23.81%
Total	5843	32.16%	38.51%	33.96%	41.35%

Table 8.6: Selected articles.

Data set	# Doc.	Surge org.	Surge class.	Plunge org.	Plunge class.
Total	8006	31.39%	37.36%	34.14%	30.81%

Table 8.7: Several test sets or one.

Data set	# Doc.	Surge org.	Surge class.	Plunge org.	Plunge class.
A	1577	36.02%	42.10%	34.94%	24.32%
B	1770	41.13%	56.87%	28.59%	23.26%
C	623	23.76%	28.13%	39.17%	52.38%
D	1014	35.11%	35.92%	31.16%	29.51%
E	1343	28.67%	20.55%	27.85%	38.04%
F	784	28.95%	29.51%	35.96%	45.33%
G	763	31.98%	35.90%	26.61%	25.35%
Total	7874	33.73%	38.79%	31.45%	32.08%

Table 8.8: Market correction.

Data set	# Doc.	Surge org.	Surge class.	Plunge org.	Plunge class.
A	1577	37.66%	61.53%	31.51%	11.76%
B	1770	39.04%	76.47%	33.22%	25.00%
C	623	13.16%	18.18%	37.40%	66.67%
D	1014	22.78%	33.33%	39.55%	$\frac{0}{0}$
E	1343	27.77%	44.82%	38.05%	83.33%
F	784	37.37%	36.36%	31.63%	64.71%
G	763	34.21%	66.78%	25.43%	33.33%
Total	7874	32.07%	55.76%	33.93%	48.84%

Table 8.9: Headlines.

We will list for each test in table 8.10, for the total of all test sets, for both *surge* and *plunge* the 95% confidence interval of the correctly classified articles with next to it the distribution in the whole data set.

Test	Surge pred.	Surge org.	Plunge pred.	Plunge org.
The reference test	0.4067 ± 0.0444	0.3207	0.3728 ± 0.0361	0.3393
Selected articles	0.3851 ± 0.0669	0.3216	0.4135 ± 0.0521	0.3396
Several or one	0.3736 ± 0.0393	0.3139	0.3081 ± 0.0406	0.3414
Market correction	0.3879 ± 0.0418	0.3373	0.3208 ± 0.0343	0.3145
Headlines	0.5576 ± 0.1056	0.3207	0.4884 ± 0.0758	0.3393

Table 8.10: 95% Confidence intervals.

In table 8.11 we will list the 80 % confidence intervals together with the confidence that our classification for this test is better than random for the specified trend. This is 97.5% if the distribution in the whole data set falls outside the 95% confidence interval for the specified classification, 90% if this distribution falls outside the 80% confidence interval and otherwise we have no confidence (0%).

Test	Surge	Plunge	Confidence surge	Confidence plunge
Reference	0.0290	0.0235	97.5%	90%
Selected articles	0.0437	0.0340	90%	97.5%
Several or one	0.0257	0.0265	97.5%	0%
Market correction	0.0273	0.0224	97.5%	0%
Headlines	0.0690	0.0495	97.5%	97.5%

Table 8.11: 80% Confidence intervals and confidence in classifier better than random.

8.1.5 Explanation of results

In general we can see in the confusion matrices in appendix E that often articles are predicted to be in the *not relevant* category. This has two causes. First, sometimes an article only contains one sentence and the feature which can be extracted out of this sentence is not yet seen by the system or does not occur at least three times in the training set. In this case we can only predict *not relevant*. Another factor is that the correlation between news articles and stock prices is not very high and combined with the fact that for both SVMs the negative training set is the largest, our system will favor the *not relevant* category. For the different tests we can conclude the following:

The reference set

From the results for the reference set we can conclude that there is a fair correlation and our system predicts better than random.

Selected articles

For the selected articles we note that there is approximately the same amount of correlation, in any case not better than what we expected giving that this class should only contain articles with much influence. Maybe this will change when we look at intraday prices, but this remains to be seen.

Several test sets or one

Because this set was trained with the smallest training period and did not take account the latest developments it was expected that it would perform worse than our reference set. Because it predicts so badly we can however also argue that it is not possible to train the system once and predict with it forever. This too is however only a pointer for feature research.

Market correction

According to chapter 4.4 we should adjust for the market influences to gain better results. It is certainly the case that for example an oil crisis would lead to a drop in the whole medical sector and therefore could for example nullify the effects of an acquired patent. As market we took the collection of all pharmaceutical stocks. However the results with our algorithm are disappointing. The test has the same confidence for *surges* as the reference set, however the difference in percentages as seen in table 8.10 is not as good and for *plunges* we cannot even prove correlation.

Maybe this is because of the α and β we chose in chapter 4.4. Another possibility is that for pharmaceutical stocks there is not much correlation between different companies. At least not so much that it exceeds the effect of news articles.

Headlines

When we look at the results of our test with headlines we see an enormous improvement on the reference set. We must however note that the articles which are actually predicted *surge* or *plunge* are very few because of the reasons explained before. But maybe if precision is very important because one actually want to make trade decisions based on such a system this could be a viable approach to build upon.

8.2 Evaluation of discovering features

An interesting sidestep is too look at the discovered features; are they pure random or can we attach meaning to them, in appendix E.2 we listed some examples.

There is no clear way of evaluating our features except confronting an expert with them and ask if the extracted features capture the meaning of a sentence. On the other hand, words, which are also often used for classification, do not capture the meaning of the sentence at all and we already noticed that the features must be generic as well. Therefore, and combined with the fact that our features are especially designed to capture negation, favorability and to be company independent, we did not develop a way for evaluating our features and leave this for future research.

To have some sort of evaluation we did look at the features considered by our classifier to be the most relevant to articles about patents in table 8.2.

Notice that whether an article is about patents is not depended on the company it discusses so we only have the *Company*-marker and not *This company* or *Other company*. Furthermore we have removed the negation and favorability

values from our algorithm because for the classification in the group of articles about patents these values have no influence. What we can see is that, next to expected features as a company who gets a ruling, also other interesting features are discovered. For example, we can argue if it is really the case that companies often issue statements in articles about patents. Much research must still be done in this field.

Agent	Relation	Target
#$Company	issue	statement
file	amend	case
#$Company	get	ruling
#$Company	get	#$PattentApproval
#$Patent	be	valid
representative	be	immediately
#$Company	settle	charge
#$Court-Judicial	uphold	#$Company
office	revoke	patent
#$Company	initiate	challenge

Table 8.12: Patent specific features.

Chapter 9

Conclusions

This thesis is neither completely theoretical nor a complete practical one. It has fairly solid background on the subjects it discusses, yet it leads to a practical system which can be evaluated using standard evaluation methods.

We have combined an automated system for predicting the influence of news articles on the stock price with comprehensive theoretical evidence why it should work and why it should work better than statistical analysis of past prices. Our knowledge about the stock market also made it possible to have argued choices for a specific domain; pharmaceutical companies, a specific ontology and to evaluate market compensating algorithms. We improved on previous work on this subject by introducing more extensive features and better evaluation measures.

To find these features we argued how to incorporate "the latest most suitable developments in Natural Language Processing" by discussing systems for sentence analysis, domain knowledge and representation of our features which led to an algorithm which extracts features using a template. These features are one hand reasonably small but on the other hand still capture important aspects which could influence the stock price. This is done by taking into account negation and favorability, abstracting from companies and using WordNet and domain knowledge for generalization. The algorithm can already be used in practice, we do need however a better evaluation of the extracted features.

By explicitly defining the problem as a text classification problem, we were able to compare existing techniques in this area and motivate why these techniques would work for our features too. Especially this thesis suggests why Support Vector Machines are well suited for our features.

The validation of the system is done using a unique data set, for which results are given that are significantly better than random and based on which some suggestions are given for feature work. Especially it is seen that looking only at headlines gives very good results.

We began our thesis with the following problem statement:

> "we want to develop a system which is able to use the latest most suitable developments in Natural Language Processing to model the reaction of the stock market to news articles and predict further reactions."

Can we really predict the reaction of stock market to news articles with this

system? One important consideration must be taken into account. Namely that we used closing prices to test the influence of news articles. On one hand correlation could be better if we took prices during the day because, as we mentioned in chapter 4, the immediate reaction on an article is better predictable. On the other hand correlation could be worse because a stock could already have risen during the morning and our system would classify most articles as positive which talk about the rise of the stock price which happened during the morning and not the event which caused the rise to happen.

But when we have real time stock prices there is at least no argument to say we can *not* predict it.

Chapter 10

Suggestions for future work

10.1 Intraday stock prices

As we noted in our conclusions, the most important step to answer the question if we can predict the reaction of the stock market on news articles is to have stock prices during the day (intraday prices). We already implemented the necessary features in our system to deal with this data and showed that the system worked for closing prices. So if it is possible to acquire this data it will not take much work to answer this question.

If the system would be used in practice it pays off coupling it to a Bloomberg terminal to let it suggest interesting articles to traders and learn directly from the real time data the Bloomberg system provides. This would however take much more time than just proving the system works for intraday stock prices.

10.2 Improvement of classification

In our motivation to use SVMs for classification based on our features (in chapter 7.6.2) we suggested to test with the features which have the lowest Information Gain, to see if they are really relevant. In this way we would be able to better support the theoretical argumentation why SVMs will do well for our features.

If intraday stock prices are incorporated it will also be worthwhile to look at topic detection and tracking (TDT) because in this way one can detect if a news article really contains new information and the effect will probably be the largest.

The last improvement for classification is to eliminate the high number of articles classified as *not relevant* as explained in chapter 8.1.5 by using word based classification in addition to our features. In this way the chance of only having unknown features in a new article is almost eliminated. The combination of different features was also suggested by Tan *et al.* and was found by them to be successful, it must however first be tested for our domain.

10.3 Improvement of features

A suggestion to improve our features is to implement anaphora resolution. In this way one can find more features, because references to earlier mentioned companies are also taken into account. In this way the sentences "GlaxoSmithKline took over Corixa. *It* also took over Pfizer", would lead to two templates, where GlaxoSmithKline is part of both of them. This should however be evaluated carefully because there are many companies mentioned in news articles and the risk of referring to the wrong company is very high. Another way of improving our features is by using more domain knowledge and instead of only using it as a single generalization step (see chapter 6.5.2), trying to use more complex relations of our domain knowledge.

As we mentioned in chapter 6.4.3 an important question remaining is how to use statistics to acquire knowledge. An example would be the use of Information Gain to extract important features for a category or, as described by Yangarber *et al.*, using features to get to documents in which one can find new features. Thereby for example be able to verify our rules of thumb mentioned in chapter 4.5.1. But it would also be possible to discover unknown patterns; new rules of thumb.

We think that here, the field of Information Technology will have its most important role the next decades; to find previously unknown patterns by analyzing and interpreting texts, images and sound.

Bibliography

[Agrawal and Mandelker, 1992] J.F. Jaffe Agrawal, A. and G.N. Mandelker. The post merger performance of acquiring firms: A re-examination of an anomaly. *Journal of finance*, 47(4):1605–1621, 1992.

[Ahmad *et al.*, 2002] K. Ahmad, P.C.F. Oliveira, M. Casey, and T. Taskaya. Description of events: an analysis of keywords and indexical names. In *Third International Conference on Language Resources and Evaluation*, 2002.

[Basu *et al.*, 2001] Sugato Basu, Raymond J. Mooney, Krupakar V. Pasupuleti, and Joydeep Ghosh. Evaluating the novelty of text-mined rules using lexical knowledge. In *Knowledge Discovery and Data Mining*, pages 233–238, 2001.

[Breatley and Meyers, 2000] R. Breatley and S. Meyers. *Principles of Corporate Finance*. McGraw-Hill, Irwin, 7 edition, 2000.

[Brehm *et al.*, 2002] Sharon S. Brehm, Saul M. Kassin, and Steven Fein. *Social Psychology*. Houghton Mifflin Co., 2002.

[Brown and Warner, 1980] S.J. Brown and J.B. Warner. Measuring security price performance. *Journal of Financial Economics*, pages 205–258, 1980.

[Buckley *et al.*, 1994] C. Buckley, G. Salton, and J. Allan. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of the seventeenth annual international ACM-SIGIR conference on research and development in information retrieval*. Springer-Verlag, 1994.

[Chan *et al.*, 2001] 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*, 9:195–217, 2001.

[Chandrasekaran *et al.*, 1999] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins. What are ontologies, and why do we need them? *IEEE Intelligent Systems*, 14(1):20–6, 1999. Publisher: IEEE, USA.

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

[Ciravegna and Petrelli, 2001] F. Ciravegna and D. Petrelli. User involvement in customizing adaptive information extraction from texts: position paper. In *IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*, 2001.

[Collier, 1996] R. Collier. Automatic template creation for information extraction, an overview. Technical report, University of Sheffield, 1996.

[Con, 2003] Connexor demo site, 2003.

[Cortes and Vapnik, 1995] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[Dow, 2003] Dow jones newswires website, http://www.dowjonesnews.com/, 2003.

[Dunham, 2002] M.H. Dunham. *Datamining - Introductory and advanced topics*. Pearson Education, 2002.

[Dutoit and Nugues, 2002] Dominique Dutoit and Pierre Nugues. A lexical network and an algorithm to find words from definitions. In Frank van Harmelen, editor, *ECAI2002, Proceedings of the 15th European Conference on Artificial Intelligence*, pages 450–454, Lyon, July 21-26 2002. IOS Press, Amsterdam.

[Fawcett and Provost, 1999] Tom Fawcett and Foster Provost. Activity monitoring: Noticing interesting changes in behavior. In Chaudhuri and Madigan, editors, *Proceedings on the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 53–62, San Diego, CA, 1999.

[Fisher, 1936] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[Fürnkranz, 1998] J. Fürnkranz. A study using n-gram features for text categorization. Technical report, Austrian Institute for Artificial Intelligence, 1998.

[Gaizauskas and Humphreys, 1996] R. Gaizauskas and K. Humphreys. Xi: A simple prolog-based language for cross-classification and inheritance. In *7th International Conference on Artificial Intelligence: Methodology, Systems, Applications*, 1996.

[Hahn and Markó, 2002] Udo Hahn and Kornél G. Markó. An integrated, dual learner for grammars and ontologies. *Data Knowl. Eng.*, 42(3):273–291, 2002.

[Hiemstra, 2001] Djoerd Hiemstra. *Using language models for information retrieval*. PhD thesis, University of Twente, 2001.

[Hommes, 2002] C.H. Hommes. Modeling the stylized facts in finance through simple nonlinear adaptive systems. In *National Academy of Sciences U.S.A.*, 2002.

[Inp, 2003] Inpharm website, http://www.inpharm.com/, 2003.

[Joachims, 1997] T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, 1997.

[Joachims, 1998] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, 1998.

[Jurafsky and Martin, 2000] D. Jurafsky and J. H. Martin. *SPEECH and LANGUAGE PROCESSING: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall, 2000.

[Keijola, 2003] Matti Keijola. *On smart and Natural Language Technology Support of Strategy Work*. PhD thesis, Helsinki University of Technology, 2003.

[Khoo *et al.*, 1999] C.S.G. Khoo, S. Chan, Y. Niu, and A.A. Ang. A method for extracting causal knowledge from textual databases. *Journal of Library & Information Management*, pages 48–63, 1999.

[Kivinen and Warmuth, 1995] Jyrki Kivinen and Manfred K. Warmuth. The perceptron algorithm vs. winnow: linear vs. logarithmic mistake bounds when few input variables are relevant. In *Proceedings of the eighth annual conference on Computational learning theory*, pages 289–296. ACM Press, 1995.

[Kohavi and Provost, 1998] R. Kohavi and F. Provost. Glossary of terms. special issue on applications of machine learning and the knowledge discovery process. *Machine Learning*, 30(2/3):271–274, 1998.

[Korczak *et al.*, 2001] J. J. Korczak, P. Lipinski, and P. Roger. Evolution stategy in portfolio optimization. In *Proceedings of 5th International Conference on Artificial Evolution*, 2001.

[Krovetz, 1993] Robert Krovetz. Viewing morphology as an inference process. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 191–202. ACM Press, 1993.

[Lavrenko *et al.*, 2000] Victor Lavrenko, Matthew D. Schmill, Dawn Lawrie, Paul Ogilvie, David Jensen, and James Allan. Language models for financial news recommendation. In *CIKM*, pages 389–396, 2000.

[Lie, 1998] D.H. Lie. Automatic summary generation: a natural language processing approach. Master's thesis, University of Twente, 1998.

[Lin, 2003a] Link grammar demo site, http://www.link.cs.cmu.edu/link/submit-sentence-4.html, 2003.

[Lin, 2003b] Link grammar website, http://www.link.cs.cmu.edu/link/, 2003.

[Luhn, 1957] H.P. Luhn. A statistical approach to mechanized encoding and searching of literacy information. *IBM Journal of Research and Development*, 1957.

[Marsh and Perzanowski, 1999] E. Marsh and D. Perzanowski. Evaluation of ie technology: Overview of results. In *MUC-7*, 1999.

[McGreevy, 1995] M.W. McGreevy. A relational metric, its application to domain analysis and an example analysis and model of a remote sensing domain. Technical report, NASA TM-110358. Ames Research Center, 1995.

[Miller *et al.*, 1990] George A. Miller, Richaxd Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to wordnet: An on-line lexical database. *Journal of Lexicography*, 1990.

[Mitchell, 1997] Tom Mitchell. *Machine Learning*, chapter 5. McGraw Hill, 1997.

[Mollá and Hutchinson, 2002] D. Mollá and B. Hutchinson. Dependency-based semantic interpretation for answer extraction. In *Australasian NLP Workshop*, 2002.

[Nagy, 2001] J. Nagy. *Comparing regression lines.* Arizona State University, 2001.

[Nasukawa and Yi, 2003] Tetsuya Nasukawa and Jeonghee Yi. Sentiment analysis: capturing favorability using natural language processing. In *Proceedings of the international conference on Knowledge capture*, pages 70–77. ACM Press, 2003.

[Nicolas *et al.*, 2003] S. Nicolas, G.W. Mineau, and B. Moulin. sesei: A cg-based filter for internet search engines. In *ICCS 03*, 2003.

[Ope, 2003a] Opencyc tutorial, http://www.opencyc.org/doc/tut/, 2003.

[Ope, 2003b] Opencyc website, http://www.opencyc.org/, 2003.

[Ordelman, 2003] Roeland Ordelman. *Dutch Speech Recognition in Multimedia Information Retrieval.* PhD thesis, University of Twente, 2003.

[Papineni, 2001] K. Papineni. Why inverse document frequency? In *Proceedings of the NAACL*, 2001.

[Pastra *et al.*, 2002] K. Pastra, Saggion H., and Y. Wilks. Extracting relational facts for indexing and retrieval of crime-scene photographs. In *roceedings of the 22nd International Conference on Knowledge Based Systems and Applied Artificial Intelligence*, pages 121–134, 2002.

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

[Peramunetilleke and Wong, 2002] Desh Peramunetilleke and Raymond K. Wong. Currency exchange rate forecasting from news headlines. In *Proceedings of the thirteenth Australasian conference on Database technologies*, pages 131–139. Australian Computer Society, Inc., 2002.

[Pro, 2003] Protégé website, http://protege.stanford.edu/, 2003.

[Raghubir and Das, 1999] Priya Raghubir and Sanjiv Ranjan Das. A case for theory-driven experimental enquiry. *Financial Analysts Journal*, 55(6):56–79, 1999.

[Reidsma, 2001] Dennis Reidsma. *Juggling word graphs, a method for modeling the meaning of sentences using extended knowledge graphs.* PhD thesis, University of Twente, 2001.

[Roux *et al.*, 2000] Claude Roux, Denys Proux, Franois Rechenmann, and Laurent Julliard. An ontology enrichment method for a pragmatic information extraction system gathering data on genetic interactions. In *Proceedings of the ECAI2000 Workshop on Ontology Learning*, 2000.

[Salton and McGill, 1983] G. Salton and M.J. McGill. *An Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[Savasere *et al.*, 1995] A. Savasere, E. Omiecinski, and S. B. Navathe. An efficient algorithm for mining association rules in large databases. *The VLDB Journal*, pages 432–444, 1995.

[Sebastiani, 2002] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002.

[Seo *et al.*, 2002] Young-Woo Seo, Joseph A. Giampapa, and Katia P. Sycara. Text classification for intelligent agent portfolio management. In *International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2002.

[Sparck-Jones, 1972] K. Sparck-Jones. A statistical interpretation of term specifity and its application in retrieval. *Journal of documentation*, 1972.

[Tan *et al.*, 2002] Chade-Meng Tan, Yuan-Fang Wang, and Chan-Do Lee. The use of bigrams to enhance text categorization. *Inf. Process. Manage.*, 38(4):529–546, 2002.

[Tapanainen and Järvinen, 1997] P. Tapanainen and T. Järvinen. A nonprojective dependency parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 64–71, 1997.

[Vapnik, 1982] V. Vapnik. *Estimation of Dependences Based on Emperical Data*. Springer Verlag, 1982.

[Wilks and Catizone, 2000] Y. Wilks and R. Catizone. Can we make information extraction more adaptive? Technical report, University of Sheffield, 2000.

[Wilks, 2000] Y. Wilks. Ir and ai: traditions of representation and anti-representation in information processing. Technical report, University of Sheffield, 2000.

[Willems, 1993] Mark Willems. *Chemistry of language, a graph-theoretical study of linguistic semantics*. PhD thesis, University of Twente, 1993.

[Xu *et al.*, 2002] F. Xu, A. Kurz, J. Piskorski, and S. Schmeier. Text extraction and mining of term relations from unrestricted texts in the financial domain. In *Proceedings of BIS 2002*, 2002.

[Yang, 1999] Yiming Yang. An evaluation of statistical approaches to text categorization. *Inf. Retr.*, 1(1-2):69–90, 1999.

[Yangarber *et al.*, 2000a] R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. Automatic acquisition of domain knowledge for information extraction. In *COLING 2000: The 18th International Conference of Computational Linguistics*, 2000.

[Yangarber *et al.*, 2000b] R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. Unsupervised discovery of scenario-level patterns for information extraction. In *Proceedings of the Sixth Conference on Applied Natural Language Processing, (ANLP-NAACL 2000)*, pages 282–289, 2000.

Appendix A

Example of a MUC-6 template for management successions

```
<TEMPLATE> :=
  DOC_NR: "NUMBER" ^
  CONTENT: <SUCCESSION_EVENT> *
  <SUCCESSION_EVENT> :=
    ORGANIZATION: <ORGANIZATION> ^
    POST: "POSITION TITLE" | "no title" ^
    IN_AND_OUT: <IN_AND_OUT> +
    VACANCY_REASON: {DEPART_WORKFORCE, REASSIGNMENT,
        NEW_POST_CREATED, OTH_UNK} ^
  <IN_AND_OUT> :=
    PERSON: <PERSON> ^
    NEW_STATUS: {IN, IN_ACTING, OUT, OUT_ACTING} ^
    ON_THE_JOB: {YES, NO, UNCLEAR}
    OTHER_ORG: <ORGANIZATION> -
    REL_OTHER_ORG: {SAME_ORG, RELATED_ORG, OUTSIDE_ORG} -
  <ORGANIZATION> :=
    ORG_NAME: "NAME" -
    ORG_ALIAS: "ALIAS" *
    ORG_DESCRIPTOR: "DESCRIPTOR" -
    ORG_TYPE: {GOVERNMENT, COMPANY, OTHER} ^
    ORG_LOCALE: LOCALE_STRING {{CITY, PROVINCE,
        COUNTRY, REGION, UNK} *
    ORG_COUNTRY: NORMALIZED-COUNTRY-or-REGION |
        COUNTRY-or-REGION-STRING *
  <PERSON> :=
    PER_NAME: "NAME" -
    PER_ALIAS: "ALIAS" *
    PER_TITLE: "TITLE" *
```

Appendix B

Connexor dependency functions

The different tags the Connexor FDG [Tapanainen and Järvinen, 1997] can produce:

main Main element: main nucleus of the sentence; usually main verb of the main clause.

qtag tag question

v-ch verb chain: auxiliaries + main verb

pm Preposed marker: grammatical marker of a subordinated clause. The marker (subordinating conjunction) itself does not have a syntactic function in the subordinated clause.

pcomp Prepositional element: The head of a nominal construction (NP or non-finite clause or nominal clause) that together with a preposition forms a prepositional phrase. Usually a preposition precedes its complement, but also tropicalized complements can occur.

phr verb particle: certain preposition-adverb homographs that form a phrasal verb with a verb.

subj Subject: the head of an NP that agrees in number with the verb of the clause. Often signals the semantic category called agent.

agt agent: the agent by-phrase in passive sentences.

obj object

comp subject complement: the head of the other main nominal dependent of copular verbs.

dat indirect object: Ditransitive verbs can take three nominal dependents: subject, indirect object and object

oc object complement: a nominal category that occurs along with an object for completing verbs.

copred copredicative

voc vocative

ins instrument

tmp time

dur duration

frq frequency

qua quantity

man manner

loc location

sou source

goa goal

pth path

cnt contingency (purpose or reason)

cnd condition

meta clause adverbial

cla clause initial adverbial

ha heuristic PP attachment

qn quantifier

det determiner

neg negator

attr attributive nominal

mod other postmodifiers

ad attributive adverbial

cc Coordination: The coordinating conjunction and one coordinated element are linked to the other coordinated element. Multiple coordinated elements are chained together. The upmost element in the chain shows the functional role of the coordinated units.

Appendix C

Pharmaceutical concepts

In this chapter we will list the concepts we added to the OpenCyc hierarchy and used for generalization.

PharmaceuticalProducingCommercialOrganization All companies

Patent patent

Divident dividend

PattentApproval approval, fda ok

PositiveTrigger up, well, increase, ...

NegativeTrigger down, negative, decrease, ...

Regulator regulator, fda, european union, ...

Evaluating trial, test, study

Court-Judicial court, tribunal, judicature, ...

MedicalTreatmentEvent therapy, treatment

Manager ceo, management, president, ...

Research research

Stock share

DrugProduct medicine, drug, medication, ...

totalCharge price

MedicalPatient patient

eventOutcomes result

Product product

income income, revenue, ...

shareholders shareholder, investor, ...

competingAgents competitor

Employee employee

expects expectation, outlook, ...

Appendix D

Pseudocode of template filling

MAIN-ANALYSIS(G, v) recursively searches in de FDG tree (G) for a company, while the last found verb is stored in v. If a company is found and the sentence contains a possible filling for the template (see chapter 6.5.1) PROCESSPATTERN is called, with as argument a FDG tree with as head the last found verb.

```
MAIN-ANALYSIS(G, v)
 1  h ← root[G]
 2  if h = verb
 3     then v ← h
 4  if h = company ∧ v ≠ NIL
 5     then PROCESSPATTERN(v)
 6  X ← firstsubtree[G]
 7  while X ≠ NIL
 8        do
 9           MAIN-ANALYSIS(X, v)
10           X ← nextsubtree[G]
```

PROCESSPATTERN(G) is called with a certain FDG tree of which the head is a verb. This method normally gets the subject and object as agent and target for a certain verb, adds the values for negation and favorability and returns the filled template. However it also takes care of exceptions; for example if instead of the *object*, the *manner* is the target or if the potential target is the main verb in an active verb chain and we try to fill the template with this verb as main verb. Furthermore it completes words which are divided over the tree such as "buy up", which is divided over two nodes.

```
PROCESSPATTERN(G)
 1  relation ← root[G]
 2  agent ← GETPART(G, subject)
 3  if agent = NIL
 4     then verbchain ← GETPART(G, verbchain)
 5          if verbchain ≠ NIL
 6             then agent ← GETINTERESTINGSUBELEMENT(verbchain)
 7  target ← GETPART(G, object)
 8  if target = NIL
 9     then target ← GETPART(G, manner)
10          if target = NIL
11             then target ← GETPART(G, subjectComplement)
12                  if target = NIL
13                     then targetGoa ← GETPART(G, goal)
14                          target ← GETPART(targetGoa, prepositionalComponent)
15                  if target = NIL
16                     then targetQua ← GETPART(G, quantity)
17                          if SYNTAX(targetQua, adverbial)
18                             then target ← targetQua
19  if target ≠ NIL
20     then if SYNTAX(target, mainVerbInActiveVerbchain)
21             then subTemplate ← PROCESSPATTERN(target)
22                  if ISVALIDTEMPLATE(subTemplate)
23                     then return subTemplate
24                     else subTarget ← GETTARGET(subTemplate)
25                          if subTarget ≠ NIL
26                             then relation ← target
27                                  target ← subTarget
28  if relation ≠ NIL
29     then phrComp ← GETPART(relation, verbParticle)
30          if phrComp ≠ NIL ∧ MORPH(phrComp, adverb)
31             then TEXT(relation) ← TEXT(relation) + TEXT(phrComp)
32  if HASNEGATION(relation)
33     then VALUE(relation) ← −10
34     else VALUE(relation) ← 10
35  if target ≠ NIL
36     then haComp ← GETPART(target, heuristicPrepositionalPhraseAttachment)
37          if haComp ≠ NIL
38             then TEXT(target) ← TEXT(target) + TEXT(haComp)
39  VALUE(target) ← GETINFLUENCE(target) target ← GETPART(G, object)
40  agent ← GENERALIZE(agent)
41  relation ← GENERALIZE(relation)
42  target ← GENERALIZE(target)
43  return TEMPLATE(agent, relation, target)
```

GETPART(*inputword*, *dependencyFunction*) returns the word which depends on *inputword* with as dependency *dependencyFunction* and returns NIL if the *inputword* was NIL or there is no word which has the specified dependency on *inputword*.

IsInteresting(*inputword*)is true if *inputword* is contained in our ontology (see C) and is not a positive or negative trigger.

GetInterestingSubElement(*inputword*) Searches for interesting words which depend, possibly in more steps, on *inputword* and returns the one which it found first. Interesting words in this case are words for which IsInteresting(*inputword*) holds, or that are *subjects*.

IsValidTemplate(*template*) returns is a template is valid, meaning it has an agent, relation and target which are all not nil.

Syntax(*inputword*, *syntax*) is true, if the syntax of word *inputword* is *syntax*.

Morph(*inputword*, *morphologicalFunction*) is true, if the morphological function of word *inputword* is *morphologicalFunction*.

GetInfluence(*inputword*) returns the total sum of the values of all children. For PositiveTrigger's as defined in appendix C the value is ten, for NegativeTrigger's the value is minus ten and for all other words the value is zero.

HasNegation(*inputword*) is true if in one of the words which depend on *inputword* are depended with a negation-relation. These words must not be main verbs in an active verb chain.

Generalize(*inputword*) returns a generalization of the word by using WordNet to get the synset, the domain knowledge in OpenCyc to get a concept, the context to get *ThisCompany* or *OtherCompany* and maps prices, numbers and times to certain tags.

Appendix E

Results

E.1 Confidence matrices of the predictions

E.1.1 Reference set

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	36	33	428
Surge	37	64	493
NR	29	36	421

Table E.1: Data set A.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	23	46	519
Surge	36	89	566
NR	25	45	421

Table E.2: Data set B.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	18	22	193
Surge	0	8	74
NR	14	30	264

Table E.3: Data set C.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	29	23	349
Surge	10	27	194
NR	27	25	330

Table E.4: Data set D.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	31	56	424
Surge	18	50	305
NR	22	46	391

Table E.5: Data set E.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	25	20	203
Surge	18	30	245
NR	21	15	207

Table E.6: Data set F.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	8	7	179
Surge	15	22	224
NR	14	19	275

Table E.7: Data set G.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	170	207	2295
Surge	134	290	2101
NR	152	216	2309

Table E.8: Total.

E.1.2 Selected articles

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	15	18	327
Surge	13	35	400
NR	14	22	303

Table E.9: Data set A.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	12	17	393
Surge	14	28	462
NR	14	14	324

Table E.10: Data set B.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	6	9	157
Surge	0	3	60
NR	3	12	216

Table E.11: Data set C.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	12	16	280
Surge	11	10	155
NR	11	15	266

Table E.12: Data set D.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	13	31	345
Surge	3	25	262
NR	9	18	313

Table E.13: Data set E.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	23	9	164
Surge	7	10	175
NR	7	13	176

Table E.14: Data set F.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	5	5	127
Surge	9	18	179
NR	7	7	216

Table E.15: Data set G.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	86	105	1793
Surge	57	129	1693
NR	65	101	1814

Table E.16: Total.

E.1.3 Several test sets or one

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	163	164	2406
Surge	177	204	2132
NR	189	178	2393

Table E.17: Total.

E.1.4 Market correction

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	18	49	484
Surge	29	64	475
NR	27	39	392

Table E.18: Data set A.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	20	46	440
Surge	33	120	575
NR	33	45	458

Table E.19: Data set B.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	11	26	207
Surge	3	18	127
NR	7	20	204

Table E.20: Data set C.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	18	33	265
Surge	16	37	303
NR	27	33	282

Table E.21: Data set D.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	35	32	307
Surge	20	30	335
NR	37	84	463

Table E.22: Data set E.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	34	22	226
Surge	20	18	189
NR	21	21	233

Table E.23: Data set F.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	154	220	2102
Surge	146	301	2209
NR	180	255	2307

Table E.24: Total.

E.1.5 Headlines

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	2	7	488
Surge	4	24	566
NR	11	8	467

Table E.25: Data set A.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	4	3	581
Surge	6	39	646
NR	6	9	476

Table E.26: Data set B.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	2	6	225
Surge	1	2	79
NR	0	3	305

Table E.27: Data set C.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	0	9	392
Surge	0	6	225
NR	0	3	379

Table E.28: Data set D.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	20	6	485
Surge	2	13	358
NR	2	10	447

Table E.29: Data set E.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	11	2	235
Surge	4	4	285
NR	2	5	236

Table E.30: Data set F.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	3	1	190
Surge	3	4	254
NR	3	1	304

Table E.31: Data set G.

Actual result	Predicted Plunge	Predicted Surge	Predicted NR
Plunge	42	34	2596
Surge	20	92	2413
NR	24	39	2614

Table E.32: Total.

E.2 Some example features

Agent	Relation	Target
it	generate (10)	sale (0)
#$ThisCompany	trim (10)	forecast (0)
#$ThisCompany	report (10)	loss (0)
development	prompt (10)	titan (0)
#$OtherCompany	present (10)	they (0)
zyprexa	get (10)	boost (0)
acquisition	provide (10)	#$OtherCompany
deal	be (10)	#$Positive (10)
attitude	be (10)	schizophrenic (0)
charge	be (10)	adequate (0)
#$stock	trade (10)	#$price (0)
#$regulator	approve (10)	restylane (0)
#$product	start (10)	age (0)
#$OtherCompany	post (10)	loss (0)
#$OtherCompany	be (-10)	#$ThisCompany (0)
#$stock	gain (10)	cent (0)

Table E.33: Some example features

Appendix F

Used and suggested tools for similar and future work

In this chapter we will shortly list different sources of tools and information we found during our project. We did not test them unless mentioned in main part of this thesis but we think they look reasonable and are worth a try.

F.1 Information sources

For experimentation there are two large corpora of news articles available; the Reuters-21578 corpus[1] and the new Reuters corpus called *Reuters Corpus Volume 1*[2]

As language sources we described WordNet[3] and The Integral Dictionary[4] in this thesis.

F.2 Language analysis

A free tool to get functional dependency structures is Link Grammar[5], we however used, the commercial available, Connexor Oy FDG[6]. They are both described in chapter 6.2.2.

As a general architecture for text engineering, we recommend GATE[7] for which already many language processing components exist.

N-gram language modeling can be done using The SRI Language Modeling Toolkit[8].

To do lexical analysis of texts one can use Wordsmith[9]

Andrew McCallums Bag of Words Library (libbow)[10] is a library of C code

[1] http://www.daviddlewis.com/resources/testcollections/reuters21578/
[2] http://about.reuters.com/researchandstandards/corpus/
[3] http://www.cogsci.princeton.edu/∼wn/
[4] http://elsap1.unicaen.fr/dicosyn.html
[5] http://www.link.cs.cmu.edu/link/
[6] http://www.connexor.com/
[7] http://gate.ac.uk/
[8] http://www.speech.sri.com/projects/srilm/
[9] http://www.lexically.net/wordsmith/
[10] http://www-2.cs.cmu.edu/∼mccallum/bow/

intended for writing statistical text-processing programs.

F.3 Generating statistical information

General datamining for Java can be done using Weka[11].

F.4 Representing ontologies

In this thesis we mentioned OpenCyc[12] for knowledge representation and Protégé[13] as a tool for for constructing ontologies. Furthermore we described Xi[14] as a tool to create small clear ontologies in Prolog.

[11] http://www.cs.waikato.ac.nz/ml/weka/
[12] http://www.opencyc.org/
[13] http://protege.stanford.edu/
[14] ftp://ftp.dcs.shef.ac.uk/home/kwh/xi.tar.gz