NTNU – Trondheim
Norwegian University of
Science and Technology

Improving News Traders using CRF

Using Conditional Random Fields to reduce
Feature Space

Simen Kind Gulbrandsen

Master of Science in Computer Science
Submission date: June 2013
Supervisor: Herindrasana Ramampiaro, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Preface

This Mater thesis is written as part of the study program Datateknikk. It was written during the spring semester of 2013. The broad idea for the project was brought up by Associate Professor Pinar Öztürk of NTNU. The spesific idea for this thesis was developed in cooperation with Associate Professor Heri Ramampiaro of NTNU.

Trondheim, 2013-06-07

Simen Kind Gulbrandsen

Abstract

Since financial markets react to news very quickly it is necessary to react even quicker in order to make money. Normal text data has very high dimensionality and reducing the number of features needed to classify a document reduces the time needed to do so. This thesis looks at a way to reduce the feature space by use of Conditional Random Fields. To do this, a new data set is made using mandatory stock messages released to the Oslo Stock Exchange. The messages are combined with financial data on all trades completed in a three-year period. A Conditional Random Field is trained on the textual data and used to extract important features. The features are then used to train a Support Vector Machine classifier and a Random Forest classifier. Both are evaluated against using all features and using randomly selected features. The thesis find that reducing the number of features results in a 4 percentage point reduction in accuracy and a 81,25% reduction in run time. We conclude that it is possible to reduce the feature space without significant reduction in accuracy. We also conclude that using this method is not good enough for making a significant profit on the financial market. This is consistent with earlier work on feature reduction.

Sammendrag - Norwegian Abstract

Aksjemarkeder reagerer lynraskt på ny informasjon. Det er derfor nødvendig for et program som skal tjene penger på nyheter å reagere enda raskere. Klassifiseringstid er avhengig av antall features som må evalueres og en reduksjon i antall features vil derfor kunne øke ytelsen. Denne rapporten forsøker å gjøre dette ved hjelp av et Conditional Random Field. Oppgaven består av å lage et nytt data sett ved å kombinere tekstlig data fra Oslo Børs med numerisk data på alle handler utført i en treårs periode. Et Conditional Random Field blir trent opp på den tekstlige dataen og brukes til å hente ut viktige ord og uttrykk som beskriver teksten kortfattet. Disse ordene og uttrykkene blir brukt til å trene en Support Vector Machine og en Random Forest klassifiseringsmodell. Begge blir evaluert mot andre systemer i litteraturen, mot samme algoritme med alle ord som features og mot algoritmene med tilfeldig utvalgte features. Rapporten finner at en reduksjon av antall features fører til en 4 prosentpoeng reduksjon i nøyaktighet og en 81.5 prosent reduskjon i tid brukt til å bygge modellen. Vi konlkuderer med at det er mulig å redusere feature-rommet uten significant redusksjon i nøyaktighet. Vi konkluderer også med at denne metoden ikke er god nok i seg selv for å tjene mye penger på aksjemarkedet. Dette passer godt med tidligere arbeid på feature reduksjon.

Acknowledgment

I would like to thank the following persons for their great help during the work on this thesis:

Jørgen Møinichen and Kristoffer Danielsen for helping me annotate the news for the CRF.
Assistant Professor Arvid Holme for supplying me with the financial data for the OSE.
Associate Professor Heri Ramampiaro for valuable input during the writing of this thesis.

(SKG)

Contents

Chapter 1

Introduction

In this chapter the scope and limitations of the thesis will be presented, as well as previous work on the problem and the structure of the report.

1.1 Background

The financial markets have many positive effects on the economy [7]. For as long as there have been stock markets, there have been people who want to get rich by beating the market. Many investors read news and make up their mind whether a stock price will go up or down. People are steered by emotions and this is reflected in their investing [9][36][20]. If a terrible news-story emerges the stock price might fall, albeit for only a short time. The fundamental value of a stock is not decided by emotions, but in a short window after a news story is released the market could be influenced by impulsive investors. According to the Efficient Market Hypothesis [22], the stock prices will incorporate all information about the stock as soon as it is available. The hypothesis exist in three different forms: Weak, semi-strong and strong. The weak form includes all historical information. The semi-strong form adapts to both historical and current information and the strong form includes the same as semi-strong in addition to insider information. Empirical studies support that EMH is true in its semi-strong form [22]. Since markets need to adapt to the new information, there is a small window of opportunity before the prices change. If a computer could predict this movement and execute a sale or purchase before the price changes, then money can be made.

The number one reason why this is an interesting task is obviously a monetary one. Being able to predict stock prices in real time could prove very profitable. On a more technical level it is interesting to improve the techniques used in the classification of textual documents. Using all terms in a document collection could make the feature space used very big, even though most of the terms will have little informational value. If the message could be represented with the use of fewer terms, computational efficiency can be improved without sacrificing accuracy.

Previous work in this field includes a number of different news traders. [12] uses a Support Vector Machine to predict changes in the stock prices on the German stock exchange. The paper manually divide the data set in to different categories of news and do a two stage classification. First they train a classifier that predicts the news category, and then they train a classifier that predicts the trend in the 15 minutes following the release. [27] analyse a number of different text mining systems for stock markets. They also introduce their own system,

NewsCAT [25], [26], [28]. They conclude that many of the prototypes have some weaknesses. Ænalyst [19],[18], [31] is a system that uses 5 categories for classification; Surge, Slight+, No recommendation, Slight-, and Plunge, to predict the intraday stock movement using news from YAHOO!Finance. They use a Naive Bayes implementation for classification. A trend in these systems is the granularity of the data. NewsCAT is the one with the shortest, and it uses 15 seconds between price evaluation. It uses three categories; Buy, Short, and No Recommendation. [43] combines news with technical indicators to predict trends in the stock market. Only closing prices are used for training and simulating. Support Vector Machines are used for classification and they are able to make a modest profit. [39] combines text learning with genetic algorithms. They use news specific to one stock to classify it. They find that stocks with many news stories in the data set are predicted with higher accuracy.

[11] examines how selecting features impact the performance of Support Vector Machines. The paper concludes that they only find one way of selecting important features that increases performance compared to using all features. They found that Bi-Normal Separation, a metric they introduce themselves, performs better than using all features. They only use statistical metrics after the fact to filter the features, with no mention of using a sequence problem solution.

[5] and [16] show that the run time of the Support Vector Machines are very dependent on the memory requirements. Both in the training and the prediction phase a limiting factor is the memory. In the end of the training part of the algorithm, all dot products between the support vectors are held in memory. If the memory requirements exceed the available memory, then the algorithm will slow down significantly. He also shows that most time in each iteration is spent on evaluating Kernel functions. This step has a complexity og *O(qlf)* where f is the maximum number of non-zero features in the training examples. This shows that reducing the number of features will greatly speed up the run time. [41] shows that the number of Support Vectors have a dramatic impact on the efficiency of SVMs and [37] shows that the number of support vectors increase linearly with the number of training instances. From the article it follows that the critical amount of memory needed scales at least like $\mathscr{B}^2 n^2$ where n is the number of support vectors and $\mathscr{B}$ is the smallest classification error achieved with a given kernel function. This means that for both large sets and noisy sets, the memory complexity will be high. For time critical processes reducing the run time will increase performance. Trading stocks is a time critical problem because the market reacts quickly to news. [42] use a feature selection method for improving the run time of SVMs using gradient descent and minimizing the bounds on the leave-one-out error.

1.2 Problem Formulation

Investigate whether classification accuracy is reduced when using only important features as selected using information extraction methods for both Support Vector Machines and Random Forest. Create a data set based on financial data in combination with news and attempt to create a predicting system using a small subset of the term features in the corpus and compare the results to other systems and algorithms. Compare the performance of this system to using all term features and to using only random features.

1.3 Objectives

1. Create data set for training and evaluation purposes
2. Train a Conditional Random Field to reduce features needed for classification
3. Compare Random Forest to Support Vector Machines
4. Compare best case with results in the literature

1.4 Structure of the Report

The rest of the report is structured as follows. Chapter 2 presents some relevant work for our problem. Chapter 3 introduces the classification methods used for classifying the documents in the data set. It also describes the *sequence problem* and how it is solved using Conditional Random Fields. Chapter 4 will describe the data set used for this thesis and the processing of it. Chapter 5 describes how the empirical experiments were done and how the results were collected. Chapter 6 presents the results of said experiments. Chapter 7 discusses the results before chapter 8 concludes and presents what further work can be done on this problem.

Chapter 2

Related Work

In this chapter, previous work on the problem will be presented in more detail. This is done to find comparable results and to clarify what part of the problem is yet to be examined. The section presents 4 different news traders that work on the principle of combining financial news with textual data. The section will also present different work done on reducing features for the Support Vector Machine classifier.

2.1 News Traders

2.1.1 Ænalyst

[31] presented the system Ænalyst. A news trader that attempts to predict changes in trends for single stock market prices. It uses a piecewise linear regression to model the market fluctuations and attempt to both predict a change in trend and the reason for it. It links trends to news messages from YAHOO!Finance. The textual data is processed by using a Part-of-Speech tagger and a Named Entity Recognizer. It attempts to create links between entities by combining the two methods. If two entities are separated by a verb, then that combination is said to be linked. The example given in the paper is "Yesterday, Bill Clinton addressed the United Nations about concerns over nuclear testing". The entities identified are Bill Clinton and United Nations. They are separated by the verb "adressed". This is called a typed link. Using these links they create a hierarchy of subsumptions which are relationships between documents. They train a Naive Bayes classifier using entity pairs and the hierarchies as well as word distributions. The difference between this approach and the approach presented in this thesis is that our approach is to use the POS and NER tags to identify important features that are independent of the Named Entities, but where the NER-tag can be used to identify such features. This thesis also uses a Support Vector Machine and a Random Forest to classify documents. This thesis uses a naive way of representing trends with the simple goal of predicting the short term change in trends.

2.1.2 NewsCAT

[28] presents a system called NewsCATS or News Categorization and Trading System. The prototype of the program was created by the Institute of Information Systems of the University of Bern, starting in 2002. They attempt to predict the changes in trends in the period of 15 minutes after a news story is released. The system is has three parts. First it preprocessed the text documents collected as press releases. They use bag-of-words features and

uses a *Collection Frequency* multiplied by *Inverted document frequency* to weight the features. Where the system is different from similar work in the past, is where they introduce a manually created thesaurus of features assumed to drive stock prices. The system then categorizes the texts into different types of news to single out news that are not relevant to price setting. They use a heuristic to filter news that have led to higher volatility in the past. The final trading phase classifies the system using three categories: good, bad and neutral based on whether the trend in the period after release increases or decreases by 3% or more. The system is created to be able to support multiple classification methods. They use a temporal granularity of 15 seconds to measure stock prices. The thing done in this thesis as opposed to NewsCATS is that our thesis uses a Conditional Random Field to create the thesaurus automatically. In stead of having to come up with all the words and phrases, which has the risk of missing many words, this thesis tries to find general patterns that indicate that the features are important. This thesis also uses a smaller temporal granularity, actually considering all trades. NewsCATS limits the number of features to 15% of the documents used. Our system does not put a predefined number on the maximum features, but rather by coincidence the important number of features extracted is close to 15%.

2.1.3 Rule based system created B. Drury and J. Almeida

[8] creates a system that uses quotations in news to predict stock movements. The rationale behind the strategy is that influential people can make the stock move. They use an example where the CEO of the Ratner Group told an audience that the reason their products were so cheap was because they were "total crap". Obviously he meant that the products were not meant to last, and customers should know that, but the quote made the company lose 500 million pounds in market value and they had to replace the CEO and change the name of the company. Few examples are as drastic as the one they use, but the reasoning is solid. They separate the speakers into biased and non-biased people with regard to the company. A CEO of a company is expected to speak positively of their own company, while a analyst in a news paper should be expected to give honest opinions. It uses a rule based classifier, created by hand to see if a statement should lead to change in stock prices. They align all the news with the stock prices by classifying a news as positive if it is released on a day when the price increased by more than 1% and negative if the direction on the day of release was negative. Movement smaller than the threshold was classified as neutral. This thesis does not use rule based classifiers, as they have to be hand made and are dependent on the creator knowing what will make the stock price change. A classification method used for stocks has to be dynamic, and an automated system can be retrained every once in a while. Using closing and opening price is considered too coarse a time interval to evaluate the significance of a news story. In the news messages evaluated in this thesis, quotes are not prevalent, meaning the data set would shrink significantly if it was the only thing considered. If the quotes has to be collected from editorial news sites, then the news are already old when they are released.

2.1.4 Combining News with Technical Indicators by Y. Zhai et. al

[43] combines news with seven technical indicators. The textual data is preprocessed to remove stop words. The special thing about their preprocessing step is that they do not use each word directly, but they use a background thesaurus called WordNet that transforms the words into higher level features. This is a way of both increasing the flexibility of the system in regards to variation in vocabularies and it works to reduce the feature space. The news are

separated into two different categories; those corresponding to a specific stock and those related to the market in general. A classifier is trained for each of the types and combined with the indicators to predict the movement of the stock on the day after the message was released. One of the indicators used is *momentum*: $C_t - C_{t-4}$ where C_t is the closing price at day t. Another indicator is the *rate of change*: $\frac{C_t}{C_{t-n}} x100$ which corresponds to the percentage change over the last n days. This thesis again considers using temporal granularity of days to be too coarse. [43] does not say why they want to reduce the feature space, only that it is a consequence of using the WordNet system. Our system uses a different method to reduce the features with the intent of reducing the run time while not reducing the accuracy. [43] are able to do 11 trades that make a profit in total. This thesis considers the volume of news messages used in the paper to be small and will use a significantly higher number to train a classifier.

2.2 Feature Reduction

2.2.1 Using F-Score and Random Forest by Y. Chen and C. Lin

[6] uses several methods to reduce the feature space of the Support Vector Machine. The first thing they do is rank the features using a *F-Score*, which is defined as:

$$F(i) \equiv \frac{\left(\overline{x}_i^{(+)} - \overline{x}_i\right)^2 + \left(\overline{x}_i^{(-)} - \overline{x}_i\right)^2}{\frac{1}{n_+ - 1}\sum_{k=1}^{n_+}\left(x_{k,i}^{(+)} - \overline{x}_i^{(+)}\right)^2 + \frac{1}{n_- - 1}\sum_{k=1}^{n_-}\left(x_{k,i}^{(-)} - \overline{x}_i^{(-)}\right)^2} \tag{2.1}$$

where $\overline{x}_i.\overline{x}_i^{(+)}, \overline{x}_i^{(+)}$ are the a ith feature of the whole positive and negative data sets respectively. $k_{k,i}^{(+)}$ is the ith feature of the kth negative instance. This measure indicates the discrimination between the positive and the negative sets and the discrimination between each individual instance within the sets. The F-score is calculated for every feature and several thresholds are used to remove the features with low F-score. The features are then used with a Support Vector Machine to find the best threshold. Their second and third method uses F-score in combination with Random Forest and *radius margin* bound SVM. The Random Forest algorithm can rank the features used. In this thesis the program Weka is used which does not support this feature of the Random Forest algorithm.

2.2.2 Empirical Study of feature selection metrics by G. Forman

[11] does a empirical analysis of 12 metrics used for selecting features for Support Vector Machines. The paper introduces their own metric called Bi-Normal separation, defined as:

$$|F^{-1}(tpr) - F^{-1}(fpr)| \tag{2.2}$$

where F is the *Z-score* and tpr and fpr are the *true positive rate* and *false positive rate* respectively. To find the best features, every feature is evaluated using the metric and then the best k features are used for classification. The other metrics considered included accuracy, Chi-Squared, Information Gain and random. Their own metric performs better than any other, and in fact it is the only metric that performs better than using all features. They evaluate all features according to the metrics, but this thesis attempts to find the features by solving a sequence problem and only using important features.

2.2.3 Convex optimization of feature selection by M. Nguyen and F. De la Torre

[30] uses a convex energy-based method for combining feature selection and parameter learning using Support Vector Machines. Because the problem is convex, the solution is a global optimum. The mathematics are presented in [30] along with the proof. They are able to perform as well as a normal Support Vector Machine with the use of significantly fewer features. This thesis recognizes that the mathematical optimality is difficult to beat, but it is interesting to see whether the use of a Conditional Random Field also can find good features using a different approach.

Chapter 3

Classification and Information extraction

This section presents the algorithms that will be used in this thesis. Those are Support Vector Machines, Random Forest and Conditional Random Fields. Some other algorithms are presented to make the presentation of the Conditional Random Field easier.

3.1 Support Vector Machines

Support Vector Machines SVM is a state-of-the-art classification method. The algorithm works by creating a hyperplane that separates the data set into two classes with a maximum margin. This is a trivial task in a 2-dimensional system where the hyperplane will be a straight line. Sometimes such a line is not present, but this is solved by mapping the whole system into a higher-dimension.

The presentation of the Support Vector Machine follows the presentation in [23]. A hyperplane is defined by a intercept term b and a normal vector $\vec{w}$ perpendicular to the hyperplane. This vector is referred to as a *weight vector* [23]. A hyperplane perpendicular to a normal vector means that all points $\vec{x}$ on the hyperplane satisfies $\vec{w}^\top \vec{x} = -b$. A two class system is the norm for Support Vector Machines, with one class being denoted as +1 and the other as -1. To classify a data point, we check which side of the hyperplane it is. Formally, with a data point $\mathbb{D} = \{(\vec{x}_i, y_i)\}$ where $\vec{x}_i$ is the point in space and the y_i is the class label. Given this, a linear classifier is then:

$$f(\vec{x}) = sign(\vec{w}^\top \vec{x} + b)) \tag{3.1}$$

The sign operator outputs whether the argument is greater or lower than zero. The output is either -1 or +1. This is used as the prediction, -1 means the classifier predicts the one class, and +1 is an indication of another class. To have some metric as to how good a classification is we can use the distance from the decision boundary. If the classification of the data point is far away from the hyperplane, the confidence of the classification is good. [23] defines the "*functional margin* of the i^{th} example $\vec{x}_i$ with respect to a hyperplane $< \vec{w}, b >$ as the quantity $y_i(\vec{w}^\top \vec{x}_i + b)$. The functional margin of a data with respect to a decision surface is then twice the functional margin of any of the points in the data set with minimal functional margin." Because we can just replace $\vec{w}$ by $5\vec{w}$ or $b = 5b$ to make the function margin as wide as we want, we call the point $\vec{x}'$ the closes point on the hyperplane to $\vec{x}$. Then

$$\vec{x}' = \vec{x} - yr\frac{\vec{w}}{|\vec{w}|} \tag{3.2}$$

r denotes the distance from $\vec{x}$ to the decision boundary and y is a factor that when multiplied with only changes the sign of the two $\vec{x}$ on either side of the boundary. Since $\vec{x}'$ lies on the decision boundary, it satisfies $\vec{w}^\top \vec{x}' + b = 0$. Since $\vec{x}' = \vec{x} - yr\frac{\vec{w}}{|\vec{w}|}$ then $\vec{w}^\top(\vec{x}' - yr\frac{\vec{w}}{|\vec{w}|}) + b = 0$. We solve this for r which gives us

$$r = y\frac{\vec{w}^\top \vec{x} + b}{|\vec{w}|} \tag{3.3}$$

We call the closest points to the hyperplane for support vectors, hence the name of the method. [23] defines the "*geometric margin* of the classifier [as] the maximum width of the band that can be drawn separating the support vectors of the two classes." This geometric margin is invariant to scaling parameters, and so we can introduce any scaling parameter we want and not affect the geometric margin. In order to make the maths prettier we impose that all functional margins are at least 1 and for some data vectors are equal to one. This means that $y_i(\vec{w}^\top \vec{x}_i + b) \geq 1$ for all vectors in the set. Our goal is to maximize the geometric margin, given as $p = \frac{2}{|\vec{w}|}$ which follows from the fact that all distances from the hyperplane is $r_i = \frac{y_i(\vec{w}^\top \vec{x}_i + b)}{|\vec{w}|}$. Maximizing $p = \frac{2}{|\vec{w}|}$ is equal to minimizing $\frac{|\vec{w}|}{2}$. This gives us a minimization problem where we want to find $\vec{w}$ and b such that:

- $\frac{1}{2}\vec{w}^\top \vec{w}$ is minimized
- for all $\{\vec{x}_i, \vec{y}_i\}, \vec{y}_i(\vec{w}^\top \vec{x}_i + b) \geq 1$

Solving this is solving a quadratic optimization problem. The procedure for solving it is to rewrite it as a dual problem. In the dual problem we introduce a *Lagrange Multiplier* α_i for each constraint $\vec{y}_i(\vec{w}^\top \vec{x}_i + b) \geq 1$ which transforms the problem into:

Find α_i,...α_N such that $\sum \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x}_i^\top \vec{x}_j$ is maximized, and

- $\sum_i \alpha_i y_i = 0$
- $\alpha_i \geq 0$ for all $1 \leq i \leq N$

The solution is then:

- $\vec{w} = \sum \alpha_i y_i \vec{x}_i$
- $b = y_k - \vec{w}^\top \vec{x}_k$ for any $\vec{x}_k$ such that $\alpha_k \neq 0$

This results in most α_i to be zero. Those that aren't zero, are *support vectors*. Finally we have a classification function:

$$f(\vec{x}) = sign(\sum_i a_i y_i \vec{x}_i^\top \vec{x} + b) \tag{3.4}$$

The problem with a solution like this is that it is prone to over fitting because it assumes that the whole data set is separable with no noise. If you optimize this system to fit every data point you would end up with something like Figure 3.1.

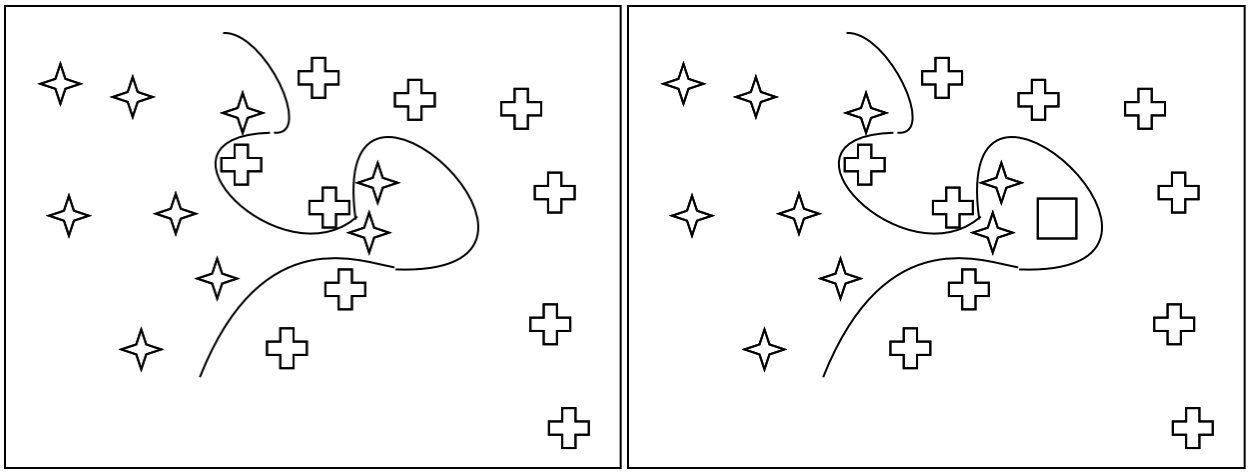

Figure 3.1: When the hyperplane is made to fit the data set perfectly, we have a problem when we introduce a new data point (square). Intuitively we would think that it should be a cross, given its position on the right hand side in the vector space, but since the two noise stars made the hyperplane curve, it is wrongly classified as a star.

To overcome the problem of over fitting, we have to accept some slack when we define the hyperplane. If the algorithm accepts that some of the data points are present on the wrong side of the hyperplane, the decision boundary becomes more general. This is often called *soft margin classification*. It is done by introducing an error constant ξ that lets the algorithm make mistakes in the initial creation of the hyperplane at a cost proportional to ξ. The formula for the *Soft Margin SVM* is then according to [23]:
Find $\vec{w}$, b and $\xi \geq 0$ so that:

- $\frac{1}{2}\vec{w}^\top \vec{w} + C\Sigma_i \xi_i$ is minimized
- and for all $\{(\vec{x}_i, \vec{y}_i)\}, \vec{y}_i(\vec{w}^\top \vec{x}_i + b) \geq 1 - \xi_i$

The *soft margin SVM* uses a trade-off between errors in classification and the width of the decision boundary. Every point that is misclassified is penalized according to ξ. The parameter C is a "*regularization term*" [23] that is set by the user and tells the algorithm how to deal with errors in training. A large C imposes a high cost for errors in classification, making the decision boundary smaller. The dual problem formulation for a *soft margin SVM* then becomes.

Find $\alpha_i, ...\alpha_N$ so that $\Sigma\alpha_i - \frac{1}{2}\Sigma_i\Sigma_j\alpha_i\alpha_j y_i y_j \vec{x}_i^\top \vec{x}_j$ is maximized and

- $\Sigma_i \alpha_i y_i = 0$
- $0 \leq \alpha_i \leq C$ for all $1 \leq i \leq N$

Notice that neither ξ or their Lagrange multiplier is in the formulation. We solve the dual problem as:
$\vec{w} = \Sigma\alpha y_i \vec{x}_i$
$b = y_k(1 - \xi_k) - \vec{w}^\top \vec{x}_k$ for $k = \text{argmax}_k \alpha_k$

Again we only need the dot product between data points to classify the elements. In the standard SVM all non-zero α_i indicates a support vector. This makes the computations simpler, but in a soft margin classification the data points inside the boundary will also have a

non-zero α_i. If the data set is large this could lead to a problem with computational complexity. Because linear SVMs complexity is dependent on both the number of features and the number of instances, they can be quite slow on big data sets.[23].

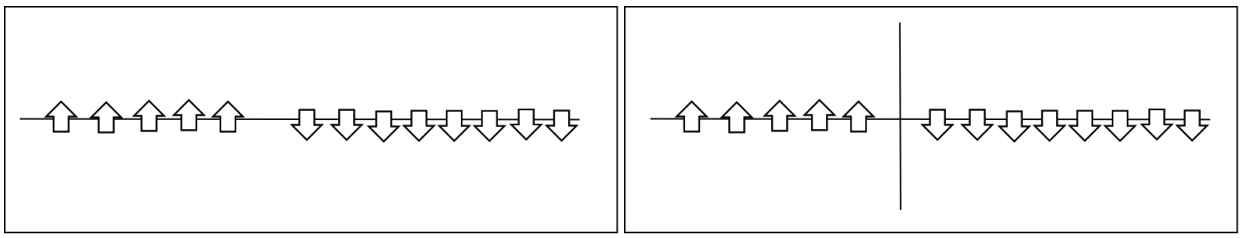

Figure 3.2: The two classes UP and DOWN are linearly separable and so we do can simply draw the line that creates the biggest margin.

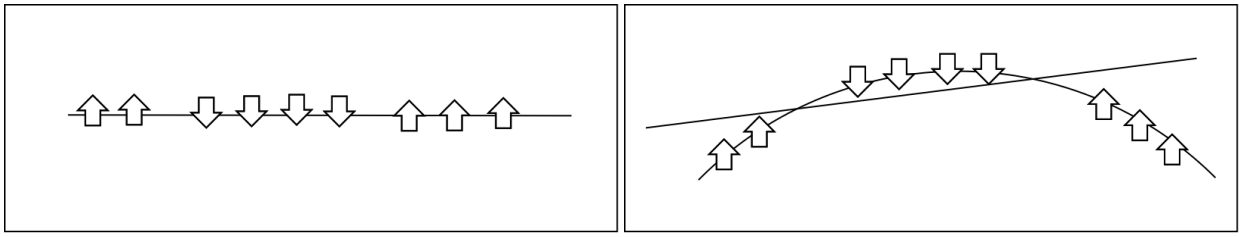

Figure 3.3: The two classes are not linearly separable in the vector space it is mapped in, but by mapping it to a higher dimension space we see that the classes become linearly separable and again we can draw the line that separates them.

To train a classifier on a data set with many dimensions the algorithm maps all data points into a very high dimension. The only thing that is needed in order to make the algorithm converge is the dot product of the Support Vectors. Calculating the cross products of a very high dimension system is complex and could not be done in reasonable time if done in the traditional way. This is where *kernel functions* come into play. They are functions where the value equals the dot product of two points in an arbitrarily high dimension; $k(x, y) = \phi(x) \cdot \phi(y)$

This can be used in the classification formula for an SVM, giving us the power of raising the data set to a high dimension and reduce the computational complexity. After the SVM has been trained, classifying a new instance is pretty straight forward. We can simply raise it to the dimension we want and see what side of the boundary it falls.

3.2 Random Forest

[4] introduces the Random Forest classifier. It is based on decision trees, but where other classification algorithms train one tree to make its decisions the Random Forest algorithm creates many.

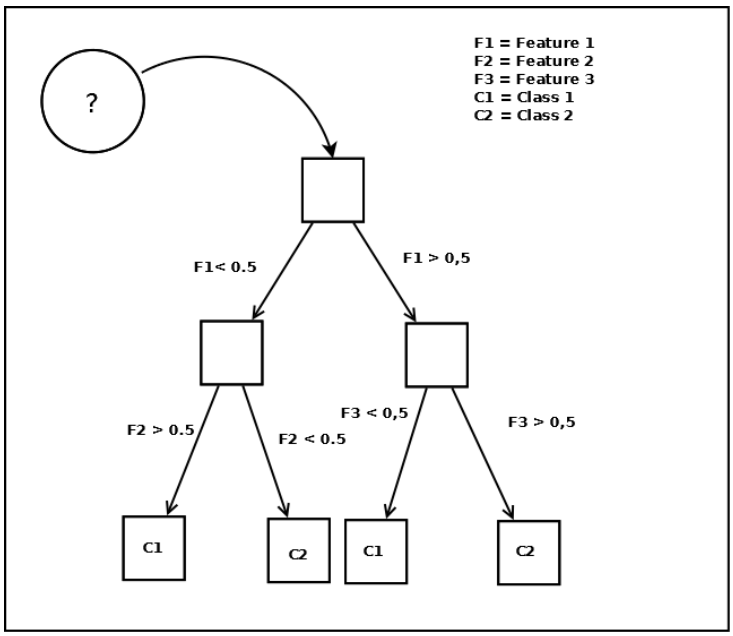

Figure 3.4: A single decision tree that evaluates three variables. An instance with the feature vector (0.8,0.4,0.2) would be classified as Class 1, and an instance with feature vector (0.3,0.6.0.2) would be classified as Class 2.

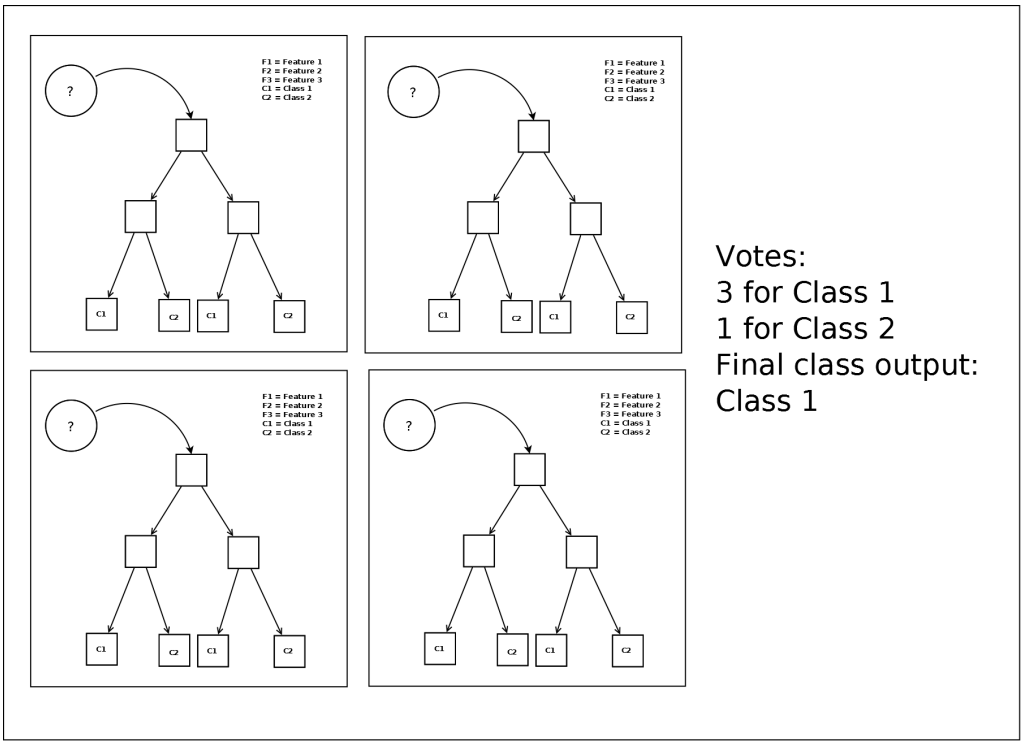

Figure 3.5: Random Forest generates multiple trees, where each node evaluates on a random variable. The majority vote from all the trees are the prediction of the whole model.

The trees are made as follows:

- Sample as many cases in the training set as there are instances, but do it with replacement.
- Choose the number of random variables N to be evaluated in each node. N should be much smaller than the total number of variables. At each node N random variables are picked and the best split is performed on these variables.
- The trees are grown to their largest extent possible without pruning.

After the trees are created, classification is done using a voting scheme. Any new instance is sent through every tree created and the output class of each individual tree is returned. The class that gets the most votes is the classifiers returned prediction.

Random Forests are shown to be resistant to noise and need only a few variables in each tree to perform well [4]. On some data sets the optimal solution is found using only one random variable in each tree. On bigger and more complex data sets a larger amount of variables lead to an increase in strength and a decrease in error-rates.

One of the strong features of the Random Forest algorithm is its ability to rank the importance of each feature. It does this by looking at the Out-of-bag-error *OOB* of the whole set and compares it to the Out-of-bag-error if the one variable is removed. To calculate the OOB, the algorithm reserves a part of the training set. These instances are sent down the trees. The OOB is then the ratio of the times when the voted class is different from the real class. The features that have the biggest impact on the OOB are most important to the classification. This information can be used to reduce the features needed for classification. Unfortunately the program Weka used in this thesis does not support feature ranking, and therefore this is not done as part of the project

The error of the forest is dependent on two things: The correlation between any two trees and the strength of each individual tree. Reducing the correlation or increasing the individual strength of the trees will make the total error rate go down. Due to the Strong law of Big Numbers the Random Forest algorithm does not overfit. This means that no matter how many trees you create and how many variables you choose, the error-rate will converge to an asymptote value. The proof of this is shown in [4] and the reader is encouraged to read the paper themselves.

3.3 Conditional random Fields

To introduce the Conditional Random Fields, the general sequence problem that it solves is presented as well as other solutions and the problem with them that CRFs solve.

3.3.1 Sequence problem

Many applications for classification are not compatible with the document classification methods that are defined above. Often the state of the current node is dependent on the surrounding states and it cant be evaluated on its own data alone. A very common problem is Part-of-Speech tagging. The goal is to tag each word in a sentence or text with their lexical

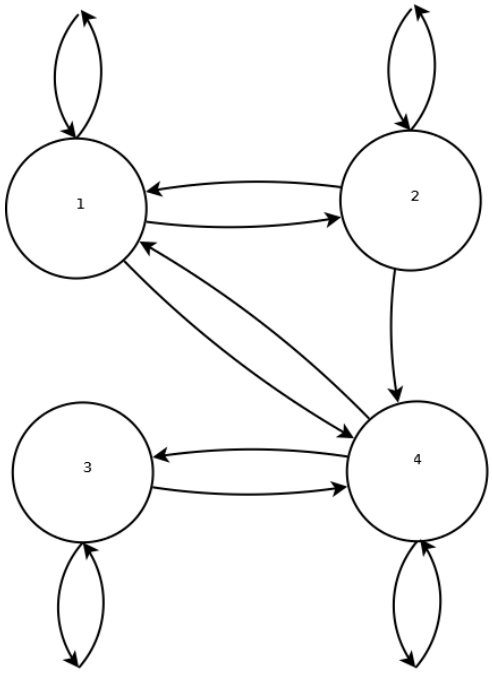

Figure 3.6: A Markov Model. Each arrow is a transition between states with a probability associated with it. No arrows from one state to another indicates that the transitional probability is zero. The sum of the probabilities in to a state is 1 and the same is true for all probabilities going out.

category. From basic grammar we have some notions of how a sentence is structured. To identify what lexical class we need to look at the other words in the sentence. To formalize this problem:

We have a set of nodes $x_i, ...x_N$. Find the states y_i that maximizes $p(y_i|x_{i...N})$.

Solving sequence problems have been done using some different methods including Hidden Markov Models - A generative model, meaning they assign a joint probability distribution to pairwise observations trying to maximize the joint probability of the training set. According to [17] these solutions are dependent on enumerating all possible observation sequences. This makes them less useful for many interactive features and long range dependencies.

3.3.2 Hidden Markov Models

A well known approach to solving sequence problems is the Hidden Markov Model or HMM-method. A Markov Model is a stochastic model that models N states and the probability of the state changing from one to another from time t_i to time t_{i+1} [34]. An example of a Markov Model is shown in Figure 3.6.

Knowing the probabilities from one state to another, we can easily find the most likely state to be in at time t. In a real world problem we usually don't know the transitional probabilities. In a sequencing problem we know the states at all times in our training set, but we don't know what model can explain these results. A HMM is a Markov Model where the transitional probabilities are hidden. Solving a HMM-problem is maximizing the probability that a model is correct given the observed states. This is done using the Baum-Welch reestimation formulas [2]. This creates a model that can be used on new test examples. The classification

happens by going through the model and find the label associated with the most likely state at time *t*.

3.3.3 Maximum Entropy Markov Models

The Maximum Entropy Markov Models or MEHMM are conditional probabilistic sequences [17] and retain all advantages of the HMMs [24] but are not generative. They are conditionally probabilistic and are able to find the next state by not only looking at the current state, but also take into account previous observations. [17]: "In MEMMs, each source state has a exponential model that takes the observation features as input, and outputs a distribution over possible next states."

3.3.4 The Label Bias Problem

Many non-generative finite state models can have a problem called *the label bias problem*. These models include; classical probabilitic automata [32], discriminating Markov models [3], maximum entropy taggers [35] and Maximum Entropy Markov Models, in addition to non-probabilistic sequence tagging and segmentation models with independently trained next-state classifiers [33] [17]. The label bias problem occurs when state machine have few outgoing transitions. Because the transition from a state only competes with other states going from that state, there will be a bias towards states with few outgoing transitions. If the state only has one out-transition it will all but ignore the observation. In Figure 3.7 a system prone to the label bias problem is shown. Both state 1 and 4 each have only one output transition. The node then has to transfer all its mass to the next state without regarding the observation at all. We see that the end result is only dependent on the first node. When a system like this is trained on a data set, if one of the words occur more often than the other, that word will win in classification every time.

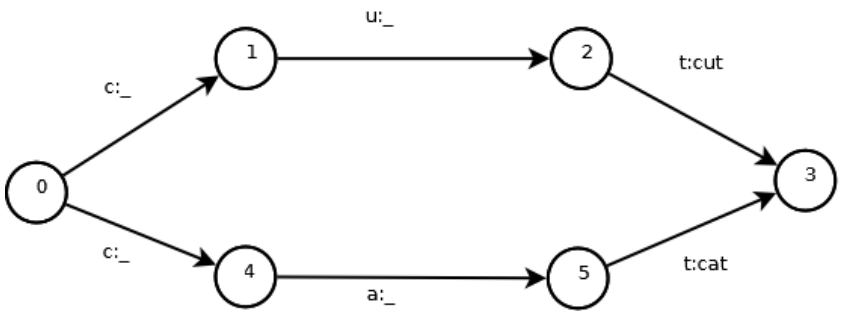

Figure 3.7: The label bias problem, after [17]. _ means the null output, meaning that no results are given from that node.

3.3.5 Conditional Random Fields

[17] introduces a solution to the sequence problem that is unaffected by the label bias problem is called Conditional Random Fields. Over the sequence to be labelled, let $\mathbf{X}$ be a random variable and $\mathbf{Y}$ is a random variable over the corresponding label sequence. For all $\mathbf{Y}_i$ in $\mathbf{Y}$, $\mathbf{Y}_i$ is part of a finite label alphabet $\mathscr{Y}$. Like the problem of Part-of-Speech tagging, $\mathbf{X}$ can be natural language words and sentences and $\mathbf{Y}$ ranges over all the possible Part-of-Speech tags.

The goal is to construct a conditional model $p(\mathbf{Y}|\mathbf{X})$. In [17] a CRF is defined as:

Definition. *Let G = (V,E) be a graph such that* $\mathbf{Y} = (\mathbf{Y}_v)_{v\in V}$*, so that* $\mathbf{Y}$ *is indexed by the vertices of G. Then* $(\mathbf{X},\mathbf{Y})$ *is a* conditional random field *in case, when conditioned on* $\mathbf{X}$*, the random variables* $\mathbf{Y}_v$ *obey the Markov property with respect to the graph:* $p(\mathbf{Y}_v|\mathbf{X},\mathbf{Y}_w, w \neq v) = p(\mathbf{Y}_v|\mathbf{X},\mathbf{Y}_w, w \sim v)$*, where* $w \sim v$ *means that w and v are neighbors in G.*

Being globally conditioned on *X*, as opposed to the HMM local conditioning, the CRF is not affected by the *label bias problem*. The simple form of CRF is the case where the CRF is a linear chain where $G = (V = 1,2,...m, E = (i, i+1))$. $\mathbf{X}$ does not need a graphical structure and does not need to have the same structure as $\mathbf{Y}$. For simplicity this paper considers $\mathbf{X} = (\mathbf{X}_1,\mathbf{X}_2,...,\mathbf{X}_n)$ and $\mathbf{Y} = (\mathbf{Y}_1,\mathbf{Y}_2,...,\mathbf{Y}_n)$.

A clique is defined in graph theory for undirected graphs G = (V,E), as a sub set of all vertices where for every two vertices in the sub set, there exists an edge connecting the two. In a tree structure and the simplest form of tree is a chain, the clique will be the edges and vertices. [17] claims this makes the joint distribution over $\mathbf{Y}$ given $\mathbf{X}$, as a consequence of the fundamental theorem of random fields [14] have the form:

$$p_\theta(\mathbf{y}|\mathbf{x}) \propto exp\left(\sum_{e\in E,k} \lambda_k, f_k(e,\mathbf{y}|_e,\mathbf{x} + \sum_{v\in V,k} \mu_k g_k(v,\mathbf{y}|_v,\mathbf{x})\right) \tag{3.5}$$

where $\mathbf{x}$ is a data sequence and $\mathbf{y}$ is a label sequence and $\mathbf{y}|_s$ is the set of components in $\mathbf{y}$ associated with the vertices in subgraph *S*. f_k and g_k are *features*. An example of this could be that $\mathbf{X}_i$ ends with "-ly" and the tag $\mathbf{Y}_i$ is "Adjective". If the feature is present, then f_k or g_k are 1 and if not they are 0.

Solving the problem is done by estimating the parameters $\theta = (\lambda_1,\lambda_2,... : \mu_1,\mu_2,...)$ from a dataset $\mathscr{D} = (\mathbf{X}^{(i)},\mathbf{Y}^{(i)})_{i=1}^N$ with the observed distribution $\tilde{p}(x,y)$. This is done by maximizing the log-likelihood objective function $\mathscr{O}(\theta)$:

$$\mathscr{O}(\theta) = \sum_{i=1}^{N} \log p_\theta(\mathbf{y}^{(i)}|\mathbf{X}^{(i)}) \propto \sum_{x,y} \tilde{p}(\mathbf{x},\mathbf{y}) \log p_\theta(\mathbf{y}|\mathbf{x}) \tag{3.6}$$

Estimating these parameters is not in the scope of this thesis. To see how it is done in practice, see [17] and [38].

Chapter 4

Collection and processing of data set

The data set is a combination of financial data and textual data. There were no easily availible data set for this problem and therefore a new one was created. This chapter will present how the data set was created and combined and how it was processed.

4.1 Financial Data

The financial data were a collection of all trades on the Oslo Stock Exchange (OSX) between the second of November 2009 and the 25th of September 2012. In total there were 55415021 transactions in the period. The format of the original files where a text file for each ticker symbol for every day that contained every transaction of that stock on that particular day. Each trade was given like this:

time	price	quantity	board	source	buyer	seller	initiator
20100208T090214	79.	72		Auto trade	NON	DNM	S

Table 4.1: Trade: Aker Solutions on 2010-02-08 at time 09:02:14

In order to make the data easier to work with a program was written that parsed all the files and saved the transactions on a time line for each ticker symbol. This resulted in 270 different time series. The number of trades for each stock varied greatly, from almost none to millions of transactions. The time lines were made by creating a Trade object for each transaction sorted by a unique ID number. Each time-line was saved as an ArrayList Java-object. In order to work on the time line a binary search method was implemented to find the closest transaction to a given time. To find the average price in a period after a message was released, the transaction number was iterated until the last trade before the end of the period.

4.2 Textual Data

The textual data were collected from the web-page NewsWeb[1], which is an outlet for mandatory news released by the Oslo Stock Exchange. All companies on the OSE is required to notify the market of many important events, i.e. insider trading and contracts of a certain size. NewsWeb should be the first place such news are released, with the possible exception of

[1] www.newsweb.no

the companies own web sites. The documents collected were all mandatory stock specific messages published in the time windows of the financial data. This resulted in 74 392 messages. Each message was parsed from the pages HTML-code and the following fields where extracted:

- Date: The time and date the message was published.
- Symbol: The ticker symbol indicating what company the message is concerning.
- Title: The title of the message.
- Message: The body of the message.
- Category: The type of message it is, i.e: Mandatory Notification of Trade

In order to further process the documents, the message and title parts where tokenized using the Stanford Natural Language Processing pack (NLP). NLP was also used to find the Part-of-Speech (POS) tag for each token and the Entity of the token was found using the NLP Named Entity Tagger (NER).

4.2.1 On removal of stop words

Stop words are terms that contribute very little to the classification of a document. They are words that occur extremely often in normal texts. It is common to remove these words during the pre-processing stage. In this thesis a Conditional Random Field is to be trained and used on proper sentences. In that regard, removing the stop words could deteriorate the CRFs performance and stop words were not removed. The stop words are removed when creating a baseline algorithm that uses all features.

4.2.2 On stemming and lemmatization

Stemming is a technique to reduce the feature numbers [21]. Only the stem of the word is used and all variations of the words are counted as a single features. Run, Runs, Running would all be instances of the stem Run. Lemmatization is very similar to stemming, but instead of finding a stem it finds the normalized form of the word. An example where the two are nor equal is the three words compute, computing and computer. The stem would be comput, but the normalized form of the word is compute. With the same rationalization as in the previous section, we do not perform stemming or lemmatization on the data set. It will not be done for training the baseline algorithm either as it has been shown to not increase performance [15].

4.3 Combining the two data sets

4.3.1 Tag the files according to stock movement

The class for the documents are defined according to the trends in the stock price after the message was released. Two different methods for obtaining this trend were used. First a simple method was implemented that calculated the average price of the stock in a given time frame after the news was released. If the average price was either significantly higher or lower based on a given threshold the class was set accordingly as either 'pos' or 'neg'. If the average

price didn't change more than the threshold value, then the document was classified as 'zero'.

The other method for obtaining the trends were implemented using linear regression. Linear regression tries to find a straight line that models the correlation between one or more variables. In our data set the only variables are time and price. The resulting line is represented as a $y = Ax + B$ equation. Here A is the slope of the curve and B represents the interception of the y axis, at the time of the released news. y is the price variable and x is the time in seconds. We can find out how much the price has risen or decreased by solving for x. To classify the documents we selected a threshold for what would be considered an increase or decrease as a percentage. The change in price could therefore be found as $1 - (y(x)/y(0)$. Linear regression gives us the possibility of checking the significance of a model. This tells us what probability the curve has of modelling the actual data. The lower the confidence the better, as it tells us what percentage chance there is for the model to be wrong. By only selecting the trends that have a significance lower than a given value, we can be confident that the data we use are relevant. To classify the documents we say that those with a positive trend and those with a negative trend and a satisfactory confidence are classified as such. Any trend that deviates little from zero or are insignificant are classified as 'zero'. Using this method performed much worse than the naive method of using average price, and therefore the simple trend tagging was used in the rest of the report.

Many messages were removed from the corpus because there were not enough data to classify them. If there were less than 10 trades in a trading period, then the message was not classified. The same was true of messages that were released outside the OSE opening hours. The only way to use them for classification would be to consider the closing price the day before versus the opening price the next day, but for short interval trading purposes it is possible that other factors influence the price change and therefore this report considered these news to add uncertainty to the data set. The total number of instances in the data set after this was 27265.

4.3.2 Finding the optimal period to trade

The whole system is made with the goal of short term trading. Therefore it would be smart to classify data using the time frame that gives the most money per transaction. In a real trading situation a dynamic process might be better, but during preprocessing this has to be done statically. To find the optimal point to hold the stocks, a program was created that checked what duration in whole minutes after the release of all news would give the most profit, when the criteria was that the average price was more than 0.5% higher or lower than the starting price to indicate a trend. Simulating trades on all these trends resulted in 44 minutes to give the best profit. Because of this result, the period of 44 minutes will be the one used in this thesis and what the classification algorithms will be trained for.

4.3.3 Tag the messages for CRF

In order to make the Conditional Random Field work, the messages had to have a specific format. In the training set the last class has to be the class to be trained for. In this training set two classes were used. Important (I) or non-important (0). In order to classify the individual documents two students agreed to help. They read through some of the articles and marked the interesting terms and phrases that would have an impact on the stocks asking price.

The terms were then tagged to comply with the format the program needed. A typical CRF training file is presented in table 4.2:

Term	PosTag	Nertag	Significance
Aker	NNP	ORGANIZATION	O
Solutions	NNPS	ORGANIZATION	O
wins	VBZ	O	I
drilling	NN	O	O
riser	NN	O	O
contract	NN	O	I

Table 4.2: Excerpt from CRF training file

4.3.4 Extract important features

The conditional random field found 3600 new terms based on the 392 terms tagged in the training documents. A program was written that found all combinations of the features that where found in close proximity of each other and checked the document frequency of all the combinations and features. If all terms in a phrase was detected in close proximity to each other, that would mean that the phrase was present. All features that occurred in less than 5 documents in the corpus was removed. The final number of features were 7690.

4.3.5 Creating feature vectors

A normal way to represent documents is by use of a vector, where each dimension represents one feature. It is also normal to weight the terms using some different schemes. A normal scheme would be the TFxIdf-weighting method. TF stands for term frequency and Idf means inverted document frequency. Term frequency is the number of times the term occurs in the specific document. This measure tells us how prevalent the term is in the document and it is intuitive that the more often a term occurs in a document the better that term describes the document. To define inverted document frequency we first define the document frequency DF. DF is the number of documents in the whole collection that contains the feature. A term that occurs in all documents will not give us much help in discriminating between classes. We want terms that occur in few documents to be weighted higher. The inverted document frequency does this, and is defined as:

$$Idf = \log(\frac{N}{df}).$$

Here N is the total number of documents in the collection and df is the document frequency. By multiplying the TF with the Idf we get a weight that increases with many occurrences of a term within a document and decreases with the number of documents that contain the term. This was done for all the important features and all files were represented as vectors in a feature space. Each dimension on the vector corresponds to a feature. This way, it is possible to use spacial classification methods like Support Vector Machines.

Chapter 5

Approach

This section will present how the experiments will be done and how the algorithms are used and configured. The implementation was done in Java.

5.1 Training CRF

To train a conditional random field two things are needed. Firstly, a set of features that describe the states of each word and secondly the dependencies between terms. In this thesis the first is done by tagging each word with its Part-of-Speech tag and its Named Entity as well as a manually annotated training set as seen in Table 4.2. The second is done using a template file that tells the algorithm what nodes to look at in relation to the current one. A section of the template file is shown in table 5.1.

Unigram
U00:%x[-2,0]
U01:%x[-1,0]
U02:%x[0,0]
U03:%x[1,0]
U04:%x[2,0]
U05:%x[-1,0]/%x[0,0]

Table 5.1: Excerpt from template file - U00: says that the current state should evaluate the term in the second to last position before it. U05: says that the current state should both evaluate the term of the previous state and its own in combination.

The actual Conditional Random Field implementation used for the work was CRF++ [1].

5.1.1 Part-of-Speech

The Part-of-Speech tag or PoS tag is the lexical group the word is a part of. Most people are familiar with the structure of a sentence and the PoS-tag analyses this structure automatically. It outputs whether a token is a verb, adjective, noun etc. and whether it is active or passive among other things. This process can be done using a Conditional Random Field, but this thesis have used a PoS tagger from the Stanford Natural Language Processing Group [2].[40]

[1] https://code.google.com/p/crfpp/
[2] http://nlp.stanford.edu

5.1.2 Named Entity

The Named Entity of a token is a class label of what the word describes. It tells us whether the token is an Organization, a place, a name, a date etc. Different models have a different number of output classes. This thesis uses the Named Entity Recognizer bundled in the Stanford Natural Language Processing Pack [10]. The model used only has the output classes: Organization, Location and Person.

5.1.3 Manual tagging

To tag the corpus with important features, two students with an economical background helped by manually reading through a number of articles and mark the terms and phrases they considered to be important to convey whether the message is positive or negative. They were given free reign over their process of evaluating the semantics of the documents. The documents were not marked with the corresponding trends, meaning they had to make up their own opinion as to the implication of the documents. There were some difference in the way the two of them selected words. One of them looked mainly for economical terms and noted that in order to make a qualified guess as to the impact of the news, some fundamental numbers for the company is needed. Some important features could still be extracted just based on words. The other person put bigger weight on terms that occur early in the document and in the title. Most documents had some terms that clearly stated the overall semantics of the document. Terms like "Wins" and "Contract" will together indicate a positive news and "CEO" and "arrested" should imply a negative message.

5.2 Classifying documents

As written in section 4.3.1, the system uses trends in stock price to tag documents as either positive, negative or neutral. Many files were removed from the data set because of lacking data to classify it according to their trend.

The Conditional Random Field program returned a set of features. These were used to create a feature vector for each document. By going through all the documents and recording the frequency of the terms and weighting their importance using normal TFxIDF weighting. These features were not only singleton words but also a combination of words occurring in close proximity within the document.

The program Weka, developed by the University of Waikato [13], was used for the act of classification. It has built in support for both Support Vector Machines and the Random Forest algorithm. It takes input in the form of a ".arff" file with a specific input format. The feature vectors for the documents were loaded into the program and many iterations of the algorithms were run. Information about each run was recorded, including run time and classification accuracy. The algorithms were first used on a subset of the whole set, and then tested on the whole data set to see whether the predicting power of the algorithm were good for unknown vectors. Then the algorithm was trained and tested using 4 fold evaluation on the wole data set.

5.3 Baseline algorithm

To see whether the accuracy is reduced when fewer features are used, the results are compared to the use of all features except stop words. In order to check the validity of the features found using the Conditional Random Field, another baseline is made using the same number of features, selected randomly from the full set of single term features found in the whole data set. To make the comparison fair, stop words are removed before the random features are selected. A program was written that checked the *collection frequency* of each term, which is the total number of times a word is present in the whole data set. A word in this case is a string of characters separated by a space or newline character. This resulted in some words occurring very often. The most frequent words are removed. The selection of random terms is done multiple times to ensure a non-biased result.

Chapter 6

Results

This chapter will present the evaluation metrics used in the results of the experiments. This is done to make the reader able to read the accuracy tables that are presented in the latter part of the chapter. The chapter presents both the confusion matrices and the detailed accuracy, both on a per-class basis and on the overall classification. The results are presented as is, and will be further discussed in the next chapter.

6.1 Evaluation Metrics.

Precision is a very normal way to evaluate classification tasks. It says what fraction of the returned documents are correctly classified.

$$Precision = \frac{TP}{TP + FP} \tag{6.1}$$

In the equation above, TP stands for True Positive, and FP stands for False Positive. A True positive in this context means a correctly classified document in one direction. For example. If a document that according to its trend is classified as positive, and the algorithm also returns a positive, it is a true positive. On the other hand, if the true class of a document is negative, and the algorithm returns a positive results, it will be a False Positive.

Recall is another standard metric. It tells us how many of the interesting documents are returned. FN means a False Negative, which would be a document that is returned as negative, but in reality is positive. TN is true negative and means that the algorithm correctly classified a negative document.

$$Recall = \frac{TP}{TP + FN} \tag{6.2}$$

F1-measure is the harmonic average between Precision and Recall. Because both precision and recall is important metrics the F-measure combines them and it is one of the most important metrics used for evaluating classification results. It is defined as:

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{6.3}$$

To change the importance of Precision and Recall in this function, a more general F-measure is defined as:

$$F_\beta = (1 + \beta^2) \frac{Precision * Recall}{(\beta^2 * Precision) + Recall)} \tag{6.4}$$

By adjusting β we can adjust the importance of precision and recall after our own preference. Setting β to one gives the standard F1-score.

The strategy supported by a news trader is two-fold: Either buy the stock when the indication is positive or short sell the stock when the indication is negative. Shorting a stock is a form of trade that profits from the stock price declining. It is done by loaning a set number of stocks from a third party, and selling it for the current market price. After a agreed upon time the stocks are to be paid back and are therefore bought in the market for the then current price. A decrease in stock price would give a net profit. In regards to evaluation this means that we should consider whether the system is able to predict with enough certainty to profit from either of these strategies. The individual precision or rather the confusion matrices for each class is therefore interesting.

6.1.1 Evaluating algorithm runs

We use two different evaluation methods. One of them is splitting the data set into one training set and a test set in a defined ratio. The other is using K-Fold evaluation. This means that we run the algorithm K times and in each run we use the K-1 first parts as a training set and test on the last part. We also shift what part of the set we use for testing. This reduces the bias when the data set is not evenly distributed. If a certain class is under-represented in the training set, it could impact the classification accuracy. Using K-fold evaluation reduces this problem by averaging over many runs. For this thesis we use 4-fold evaluation.

6.2 Results for the Conditional Random Field

The manual tagging described in section 5.1.3 resulted in 392 tagged words. After the Conditional Random Field had run its course, the total number of terms identified were 3600. Important features that occured in close proximity to each other were combined and any combinations of them were generated. These combinations were then regarded as new features. Manual overlook of the returned terms showed both very relevant and some irrelevant terms. Some of the terms returned were two terms put together either by mistake in the parsing of the messages or in the original news. Some terms were also gibberish, i.e "no001.056444-6", and some of the terms were non-descript, i.e a single character. To remove the features that would give nothing to the classification, the terms that occured in less than 5 documents were removed. The resulting number of features were 7690.

6.3 Results using All Features

First the algorithms are run using all single term features with stop words removed. The Random Forest returned 10900 positive documents while the Support Vector Machine returned 9503. The overall difference in precision is 1.6% with the SVM being slightly better. Tables 6.1 - 6.3 show the confusion matrices and detailed accuracy for the two algorithms. The best results are written in bold.

a	b	c	<– classified as
4777	1938	2787	a = pos
2976	2391	2319	b = neg
3147	1958	4972	c = zero

Table 6.1: All Features - Random Forest - 10 trees

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.503	0.345	0.438	0.503	0.468	0.611	pos
	0.311	0.199	0.38	0.311	0.342	0.594	neg
	0.493	0.297	0.493	0.493	0.493	0.639	zero
Weighted Avg.	0.445	0.286	0.442	0.445	0.442	0.617	

Table 6.2: All Features - Random Forest - Detailed Accuracy By Class

a	b	c	<– classified as
4324	1957	3221	a = pos
2467	2595	2624	b = neg
2712	1699	5666	c = zero

Table 6.3: All Features - Support Vector Machine

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.455	0.292	0.455	0.455	0.455	0.599	pos
	0.338	0.187	0.415	0.338	0.372	0.594	neg
	0.562	0.34	0.492	0.562	0.525	0.626	zero
Weighted Avg.	0.462	0.28	**0.458**	**0.462**	**0.4581**	0.608	

Table 6.4: All Features - SVM - Detailed Accuracy By Class

6.4 Random Features

To evaluate the extracted features, both algorithms are run using the same number of features as the Conditional Random Field extracted, selected at random. The selection was done three times and the algorithms were run once for each collection of random features. The accuracy for the runs fluctuated more using the Random Forest algorithm than using the Support Vector Machine. The confusion matrices and detailed accuracy for all runs are shown in Tables 6.5 - 6.16.

6.4.1 SVM

a	b	c	<– classified as
3032	1092	5378	a = pos
1665	1116	4905	b = neg
1657	729	7691	c = zero

Table 6.5: Random Features - Support Vector Machine. Full set - Run 1

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.319	0.187	0.477	0.319	0.382	0.581	pos
	0.145	0.093	0.38	0.145	0.21	0.536	neg
	0.763	0.598	0.428	0.763	0.548	0.588	zero
Weighted Avg.	0.434	0.313	**0.432**	0.434	0.395	0.571	

Table 6.6: Random Features - Support Vector Machine. Full set - Run 1 - Detailed Accuracy By Class

a	b	c	<– classified as
3186	926	5390	a = pos
1783	1144	4759	b = neg
1612	707	7758	c = zero

Table 6.7: Random Features - Support Vector Machine. Full set - Run 2

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.335	0.191	0.484	0.335	0.396	0.584	pos
	0.149	0.083	0.412	0.149	0.219	0.544	neg
	0.77	0.59	0.433	0.77	0.554	0.595	zero
Weighted Avg.	0.443	0.308	0.445	0.443	0.405	0.577	

Table 6.8: Random Features - Support Vector Machine. Full set - Run 2 - Detailed Accuracy By Class

a	b	c	<– classified as
2991	1062	5449	a = pos
1672	1216	4798	b = neg
1671	698	7708	c = zero

Table 6.9: Random Features - Support Vector Machine. Full set - Run 3

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.335	0.191	0.484	0.335	0.396	0.584	pos
	0.149	0.083	0.412	0.149	0.219	0.544	neg
	0.77	0.59	0.433	0.77	0.554	0.595	zero
Weighted Avg.	0.443	0.308	0.445	0.443	0.405	0.577	

Table 6.10: Random Features - Support Vector Machine. Full set - Run 3 - Detailed Accuracy By Class

6.4.2 Random Forest

a	b	c	<– classified as
3988	1682	3832	a = pos
2390	1799	3497	b = neg
2583	1597	5897	c = zero

Table 6.11: Random Features - Random Forest - Run 1

The Random Forest runs had better F-measure, while the Support Vector Machine runs had better Precision. Precision will be shown to be more important from a financial point of view in the discussion section.

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.42	0.28	0.445	0.42	0.432	0.6	pos
	0.234	0.167	0.354	0.234	0.282	0.582	neg
	0.585	0.426	0.446	0.585	0.506	0.619	zero
Weighted Avg.	0.429	0.302	0.42	0.429	0.417	0.602	

Table 6.12: Random Features - Random Forest - Run 1 - Detailed Accuracy By Class

a	b	c	<– classified as
4360	2074	3068	a = pos
2664	2370	2652	b = neg
2906	2137	5034	c = zero

Table 6.13: Random Features - Random Forest - Run 2

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.459	0.314	0.439	0.459	0.449	0.599	pos
	0.308	0.215	0.36	0.308	0.332	0.582	neg
	0.5	0.333	0.468	0.5	0.483	0.623	zero
Weighted Avg.	0.431	0.293	0.428	0.431	0.429	0.603	

Table 6.14: Random Features - Random Forest - Run 2 - Detailed Accuracy By Class

a	b	c	<– classified as
4251	2362	2889	a = pos
2574	2611	2501	b = neg
2870	2312	4895	c = zero

Table 6.15: Random Features - Random Forest - Run 3

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.447	0.306	0.438	0.447	0.443	0.599	pos
	0.34	0.239	0.358	0.34	0.349	0.582	neg
	0.486	0.314	0.476	0.486	0.481	0.625	zero
Weighted Avg.	0.431	0.29	0.43	0.431	**0.43**	0.604	

Table 6.16: Random Features - Random Forest - Run 3 - Detailed Accuracy By Class

6.5 Results using extracted Features

Both the Random Forest and the Support Vector Machine was run with the CRF-extracted features. The total number of features were 7960. Multiple runs where run for each algorithm. The different way the algorithms were evaluated were using 4-fold evaluation, first on a small subset of the total data set, using only 6685 instances of the whole 27265. The algorithms were also run, using the same number of instances for training and evaluating on the rest of the full set. And finally the full set was evaluated using 4-fold evaluation. The Random Forest algorithm was also evaluated on the small set using 4-fold evaluation, but increasing the number of trees created from 10 to 100.

6.5.1 SVM

The confusion matrices and detailed accuracy for the Support Vector Machine runs are presented in Tables 6.17 - 6.22.

a	b	c	<– classified as
1112	445	831 |	a = pos
647	539	733 |	b = neg
694	368	1316 |	c = zero

Table 6.17: SVM - Small set

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.466	0.312	0.453	0.466	0.459	0.594	pos
	0.281	0.171	0.399	0.281	0.33	0.573	neg
	0.553	0.363	0.457	0.553	0.501	0.606	zero
Weighted Avg.	0.444	0.29	0.439	0.444	0.437	0.592	

Table 6.18: Support Vector Machine - Small Set - Detailed Accuracy By Class

a	b	c	<– classified as
3034	1055	3287 |	a = pos
1893	1314	2835 |	b = neg
1997	958	4894 |	c = zero

Table 6.19: SVM - 22% of full set used for training

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.411	0.28	0.438	0.411	0.424	0.582	pos
	0.217	0.132	0.395	0.217	0.28	0.549	neg
	0.624	0.456	0.444	0.624	0.519	0.591	zero
Weighted Avg.	0.435	0.303	0.428	0.435	0.418	0.576	

Table 6.20: Support Vector Machine - Full Set 22% split - Detailed Accuracy By Class

a	b	c	<– classified as
3572	1194	4736 |	a = pos
2004	1559	4123 |	b = neg
1992	943	7142 |	c = zero

Table 6.21: SVM - Full set - 4 fold evaluation

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.376	0.225	0.472	0.376	0.419	0.593	pos
	0.203	0.109	0.422	0.203	0.274	0.559	neg
	0.709	0.515	0.446	0.709	0.548	0.603	zero
Weighted Avg.	0.45	0.3	0.448	0.45	0.426	0.587	

Table 6.22: Support Vector Machine - Full Set - Detailed Accuracy By Class

6.5.2 Random Forest

Tables 6.23 - 6.30 shows the results for the Random Forest Algorithm. It is worth noting that using 100 trees increased the accuracy of the classification, but running it on the whole data set took to increase the number of trees further in this thesis.

a	b	c	<– classified as
1258	492	638	a = pos
716	626	577	b = neg
813	512	1053	c = zero

Table 6.23: Random Forest - Small set - 10 trees

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.527	0.356	0.451	0.527	0.486	0.613	pos
	0.326	0.211	0.384	0.326	0.353	0.581	neg
	0.443	0.282	0.464	0.443	0.453	0.618	zero
Weighted Avg.	0.439	0.288	0.437	0.439	0.436	0.606	

Table 6.24: Random Forest - Small Set - Detailed Accuracy By Class

a	b	c	<– classified as
3647	1451	2278	a = pos
2475	1690	1877	b = neg
2712	1533	3604	c = zero

Table 6.25: RT - 22% of full set used for training - 10 trees

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.494	0.373	0.413	0.494	0.45	0.585	pos
	0.28	0.196	0.362	0.28	0.315	0.566	neg
	0.459	0.31	0.464	0.459	0.462	0.602	zero
Weighted Avg.	0.42	0.299	0.417	0.42	0.416	0.586	

Table 6.26: Random Forest - Full Set 22% used for training - Detailed Accuracy By Class

a	b	c	<– classified as
4876	1948	2678	a = pos
3037	2302	2347	b = neg
3248	1901	4928	c = zero

Table 6.27: RT - Full set with 4 fold evaluation - 10 trees

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.513	0.354	0.437	0.513	0.472	0.612	pos
	0.3	0.197	0.374	0.3	0.333	0.583	neg
	0.489	0.292	0.495	0.489	0.492	0.634	zero
Weighted Avg.	0.444	0.287	0.441	0.444	0.44	0.612	

Table 6.28: Random Forest - Full Set 10 Trees- 4 Fold Evaluation - Detailed Accuracy By Class

a	b	c	<– classified as
1344	377	667 |	a = pos
775	555	589 |	b = neg
822	427	1129 |	c = zero

Table 6.29: RT - Small set - 100 trees

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.563	0.372	0.457	0.563	0.504	0.627	pos
	0.289	0.169	0.408	0.289	0.339	0.597	neg
	0.475	0.292	0.473	0.475	0.474	0.632	zero
Weighted Avg.	0.453	0.285	**0.449**	**0.453**	**0.446**	0.62	

Table 6.30: Random Forest - Small Set 100 Trees - 4 Fold Evaluation - Detailed Accuracy By Class

The tables show that the Random Forest algorithm performed better than the Support Vector Machines, when using the extracted features from the Conditional Random Field. The best results were from using 100 trees in the Random Forest, ant this run performed best overall for Precision, Recall and F-Measure.

Chapter 7

Discussion

This chapter will discuss the results presented in Chapter 6. The sources of error identified for this thesis will be presented first, before discussing the implications of the results. This chapter also discusses the big difference that small increases in accuracy make when applying a real trading strategy to the system. A higher precision is shown to be more important than recall.

7.1 Sources of error

In Section 4.3.2 it was found that 44 minutes was the optimal time to hold stocks to maximize profits. This analysis was done in a rather crude way, and only looked at periods up to 120 minutes. It is possible that the optimal period is not correct or that the optimal period is outside the range examined. A classification algorithm is dependent on the classes in the training set really being correct. Many news stories will undeniably contribute little to the change in stock prices. Sometimes other occurrences could be the real reason a stock has changed that is not covered by mandatory notifications. A message related to one stock could have an impact on other stocks and so could make concurrently released notifications hard to classify correctly. It is also possible that a message that would normally make the stock move, does not on the account of high expectations etc. Such things can not be identified by the system in this thesis, but having a sufficiently big data set should overcome most of these problems. A source of error would then be whether our data set is big enough to correctly model the problem.

One problem that came up late in the project, was that the original parsing of the web pages sometimes ended in two words occurring as one. This could result in the Conditional Random Field training to be erroneous and the extraction of terms for the feature vectors to be less accurate.

In a real world application there could be a problem with getting the information to a trader fast enough. The stock prices change rapidly and if the system takes a long time to complete its analysis and send its order to the broker, the system could miss the upswing.

7.2 Compare the algorithms

When we look at the results from the Support Vector Machine versus the Random Forest, we see that there is not a big difference between them result wise. The SVM has an F1-score of 45 % and the same is true of the Random Forest algorithm. They are both better at predicting a positive class than a negative class. This could be the result of a skew in the distribution of classes. There are less negative classes than positive and neutral classes. One could theorize that having less of one class makes the algorithm worse at predicting that class. This is a known problem with SVMs on very skewed distributions [1]. In these cases the distribution are skewed on a more extreme scale, as much as 99.9 % towards one class. In our case it is less extreme and we should think that there are enough training examples to properly train the algorithm. One simple explanation for the skewness is that the differences between the positive and the negative examples aren't great enough. We know that "From buy to sell" and "From sell to buy" would have the same feature vector if those were the only terms and we only consider singleton terms. However the implications are obviously opposite. In our way of considering multiple terms in a feature, we look at a distance of 10 between words. Perhaps we would see better results by reducing the distance.

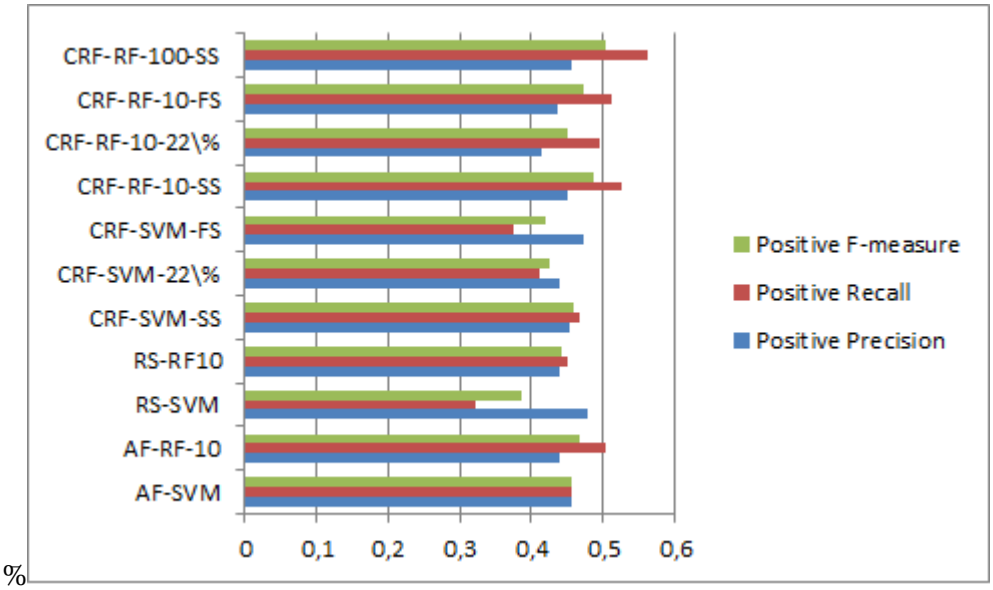

Figure 7.1: Precision, Recall and F-measure - Positive class

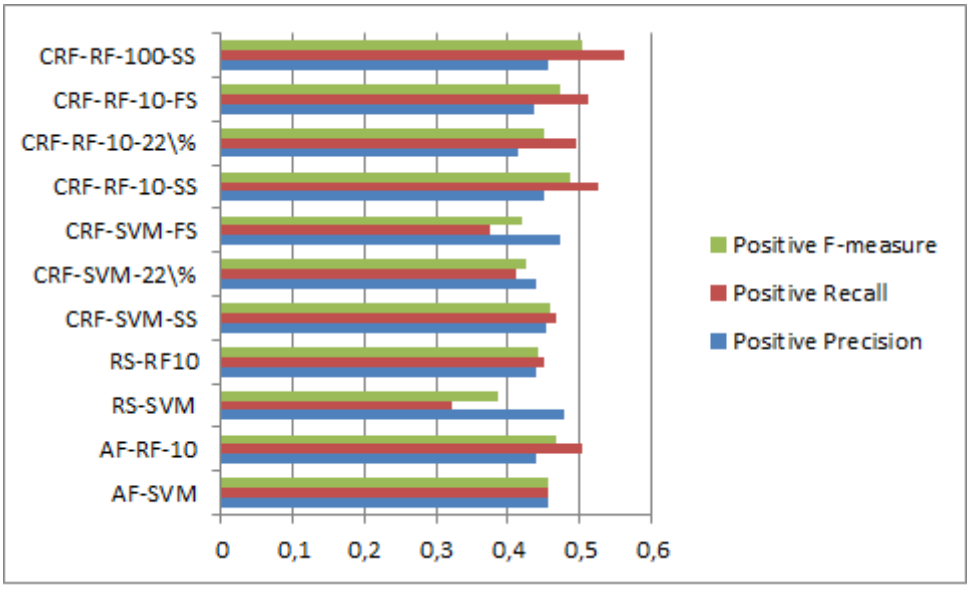

Figure 7.2: Precision, Recall and F-measure - Negative class

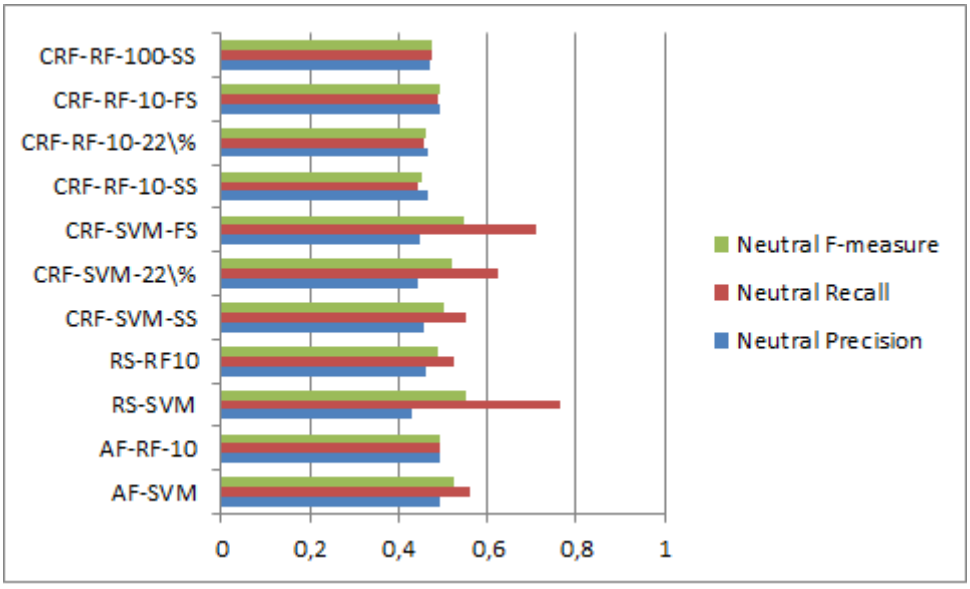

Figure 7.3: Precision, Recall and F-measure - Neutral class

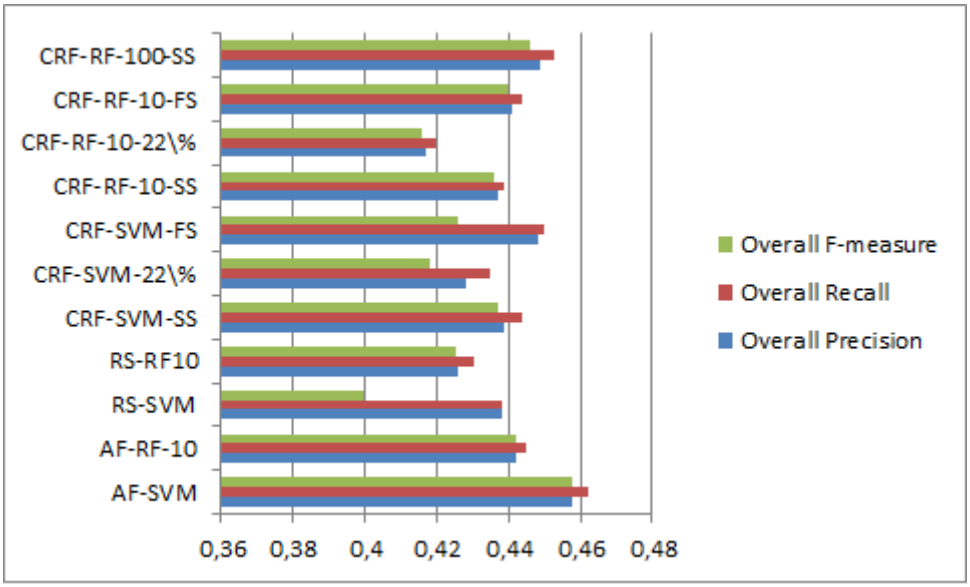

Figure 7.4: Precision, Recall and F-measure - Overall

From Figure 7.4 we see that a SVM taking all features into consideration perform better than all the other instances for overall classification. The difference is quite small (note the scale of the figure). From this we can take that using a Conditional Random Field does not improve the results of the SVM algorithm. Indeed it seems to do the opposite. When comparing the features extracted by the CRF with using random features (RS-SVM and RS-RF in the figures), it is apparent that the Conditional Random Field performs better. The margins are not very big, but both the RF and the SVM perform better on CRF features. When only looking at the positive class in figure 7.1 it seems the random forest algorithm performs best when evaluating over 100 trees. The recall of the random forest algorithm is higher in most of the cases but has a lower precision. A reason why the Random Forest algorithm performs a little better with the CRF features could be because the Random Forest in it self is selecting features at random and creates a tree evaluating on these features. The CRF will in this case reduce the space it selects features from, removing unnecessary features, which should improve the result.

Although the difference in accuracy is relatively small, there is another aspect to be considered - time. The time it takes to build a model was very dependent on the number of features as seen in figure 7.5. We see that the SVM model using the full set of features took 5 times longer to make than the one with the conditional random field selected features. What is somewhat surprising is that the Random Forest model for randomly selected features used 5 times longer than for the CRF selected features. The time was tested using three different sets of randomly selected features and the shown time is the average of those runs. It is probable that the reason for the big difference is that the random features have problems splitting the data set properly and therefore has to build very deep trees in order to satisfy the stop criteria. The SVM actually took less time to create on the random features than on the CRF selected features. It is also possible that external factors had an impact on the run time, like load on the system.

Being a system meant to predict stock movement, a good way to evaluate its real world application would be to see whether it can make a profit. The system is trained a threshold of the average price being 0.5 percent higher after 44 minutes. We can safely assume that the price after 44 minutes will be more than 0.5 percent higher in the training set, and if we assume a 0.5 percent profit from the correctly classified positive examples, a loss of 0.5 from the ones classified as positive but that in reality were negative, and no change from the ones that are actually neutral, we can calculate the profit as $(0.5 * TP) - (0.5 * FP) = 0.5(TP - FP)$. FP are only the ones that are really negative. In a real world trading environment the neutral class wrongly classified as positive will make us lose money in the form of transaction costs. This means that the most important metric is average profit per transaction. There is also a strategy called *selling short*, where you bet against the market. Consider if you borrow 100 stocks from a third party and agree to pay him back 100 shares in 44 minutes. You sell the stock on the market at the current price, predicting that the price will go down. After 44 minutes you buy the stock back at market price and give them back to the third party. If the market goes down profit is made. With the same assumptions as earlier this would yield profits in the same way, but only for the negative class. A true positive here is then the times when stock prices actually decline, and a false positive is a returned negative trends that turns out to be positive. The results for the various algorithms are shown in 7.1 and 7.2.

From tables 7.1 and 7.2, it becomes clear that there is a big correlation between precision

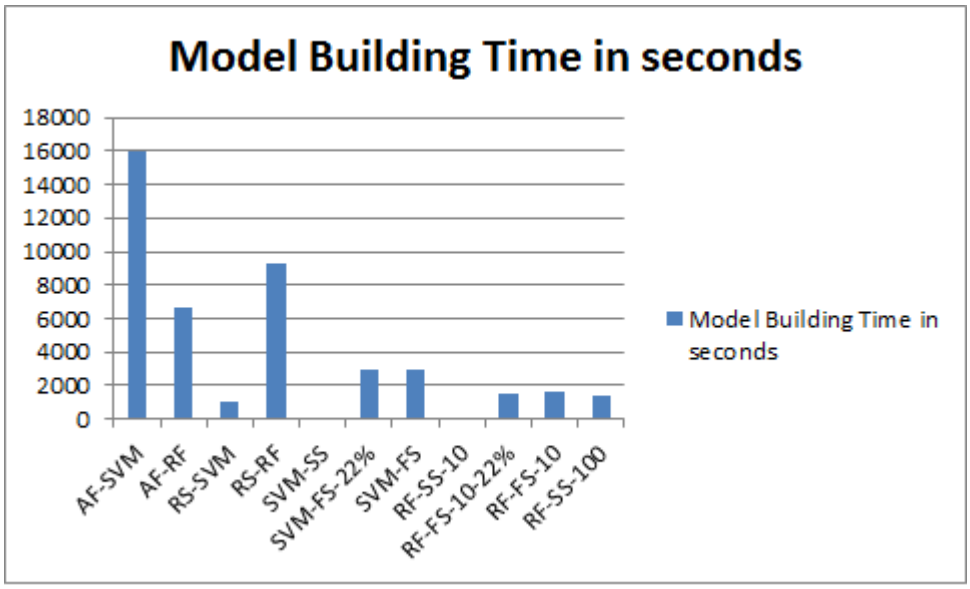

Figure 7.5: Time used for building model

Algorithm	TP	FP	TP%	FP%	Additive profit%	Avg Profit per trade%	#OfTrades
AF-SVM	4324	2467	63.7	36.3	928.5	0.098	9503
AF-RF	4777	2976	61.7	38.3	900.5	0.083	10900
RS-SVM	3070	1707	64.3	35.7	681.5	0.106	6424
RS-RF	4200	2543	62.3	37.7	828.5	0.087	9529
SVM-SS	1112	647	63.2	36.8	232.5	0.094	2453
SVM-FS-22%	3034	1893	61.6	38.4	570,5	0.082	6924
SVM-FS	3572	2004	64.1	35.9	784.0	0.103	7568
RF-SS-10	1258	716	63.7	36.3	271.0	0.097	2787
RF-FS-10-22%	3647	2475	59.6	40.4	586.0	0.067	8834
RF-FS-10	4876	3037	61.6	38.4	919.5	0.082	11161
RF-SS-100	1344	775	63.4	36.6	284.5	0.097	2941

Table 7.1: Results for the positive class

Algorithm	TP	FP	TP%	FP%	Additive profit%	Avg Profit per trade%	#OfTrades
AF-SVM	2595	1957	57.0	43.0	319.0	0.051	6251
AF-RF	2391	1938	55.2	44.8	226.5	0.036	6287
RS-SVM	1159	1027	53.0	47.0	66.0	0.022	2937
RS-RF	2260	2039	52.6	47.4	110.5	0.018	6314
SVM-SS	539	445	54.8	45.2	47.0	0.035	1352
SVM-FS-22%	1314	1055	55.5	43.5	129.5	0.039	3327
SVM-FS	1559	1194	56.6	43.4	182.5	0.049	3696
RF-SS-10	626	492	56.0	44.0	67.0	0.041	1630
RF-FS-10-22%	1690	1451	53.8	46.2	119.5	0.026	4674
RF-FS-10	2302	1948	54.2	45.8	177	0.029	6151
RF-SS-100	555	377	62.2	37.8	89.0	0.065	1359

Table 7.2: Results for the negative class

and the average profit of the trades. The Support Vector Machines returned a better average profit, with the exception being Random Forest with 100 trees for the negative class. There is a big reason why average profit per trade is more important than total additive profit, namely transaction costs. Transaction costs occur in every trade. Currently the lowest transaction cost available from Nordnet [1] is 0.035%. From now on a basis point is 0.01% or one in ten thousand. Transaction cost is deducted twice per round trip, once when buying and once when selling. This means that a transaction have to make more than 7 basis points just to make a profit. Slightly higher in reality because the transaction costs are calculated from the selling price. Some of the algorithms are able to perform better than 7 basis points on the positive class but none do on the negative class. The negative class will therefore not be considered any more. To see the difference between the different algorithms, a program is run that calculates the end results after all the trades are done. The simulation assumes that we use one fifth of our capital in every trade. The reason for this assumption is that there can occur multiple trades during the time we hold our stocks before selling. The results are shown in figure 7.6 and 7.7. Without the transaction costs the extra recall of the random forest algorithm makes up for the lesser average profit, but when the transaction costs are considered the Support Vector Machine on the whole set is supreme. The random forest is penalized for its many transactions that incur transaction costs.

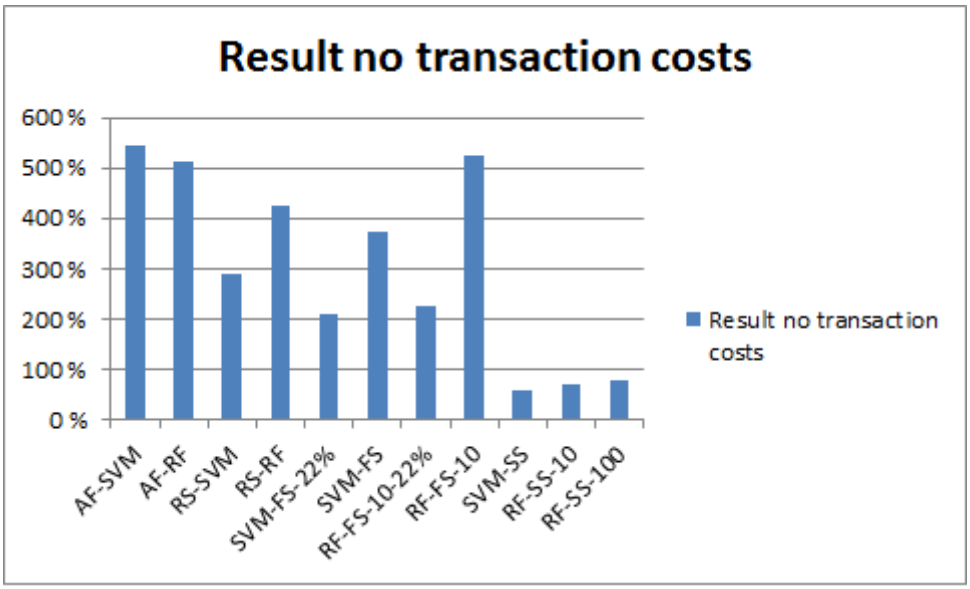

Figure 7.6: Results without transaction costs

Using the best performing algorithm in Figure 7.6, we can see from Figure 7.8 that the profit is very dependent on the size of the transaction costs. It is therefore imperative for a real trading system to negotiate a lower transaction cost with the moving houses or trade directly with the stock exchange.

An interesting point to consider is that the total money paid in transaction costs are less dependent on the actual transaction cost percentage. If we look at Figure 7.9 we see that the total paid transaction costs paid actually flatten out at around 0.05% and then start to decline. This is because, when we look at Figure 7.8, we see that the system loses money

[1] www.nordnet.no

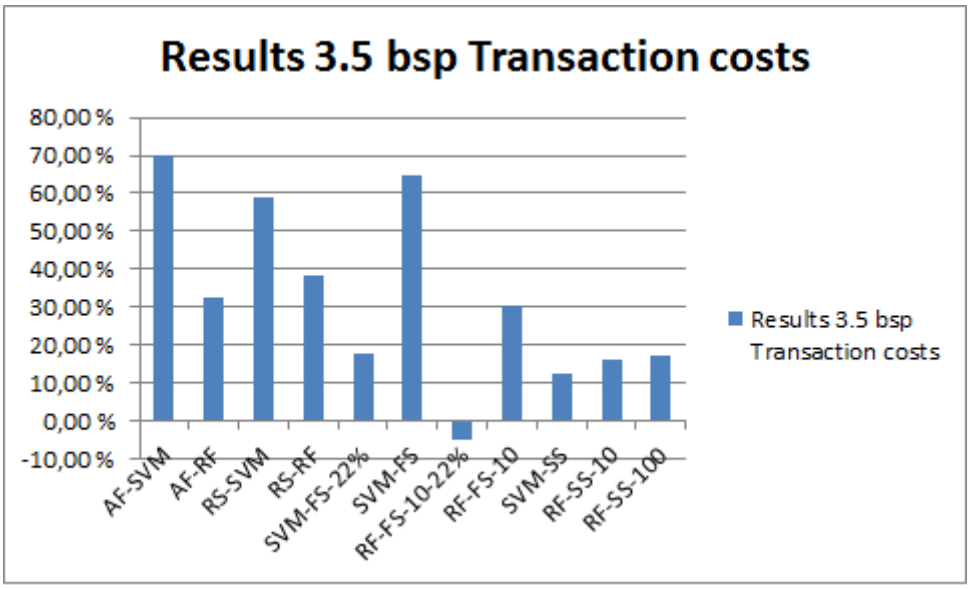

Figure 7.7: Results with transaction costs

Figure 7.8: Results versus transaction costs

around the same point. When the capital goes down, so does the transaction costs paid per investment. What is important to take away from these two graphs is that reducing the transaction costs from 0.04% to 0.03% doubled our profits, but the total transaction costs paid only declined by 11%. That means that it would be in both the brokers and our own interest to pay the nominal difference in order to get a lower transaction cost.

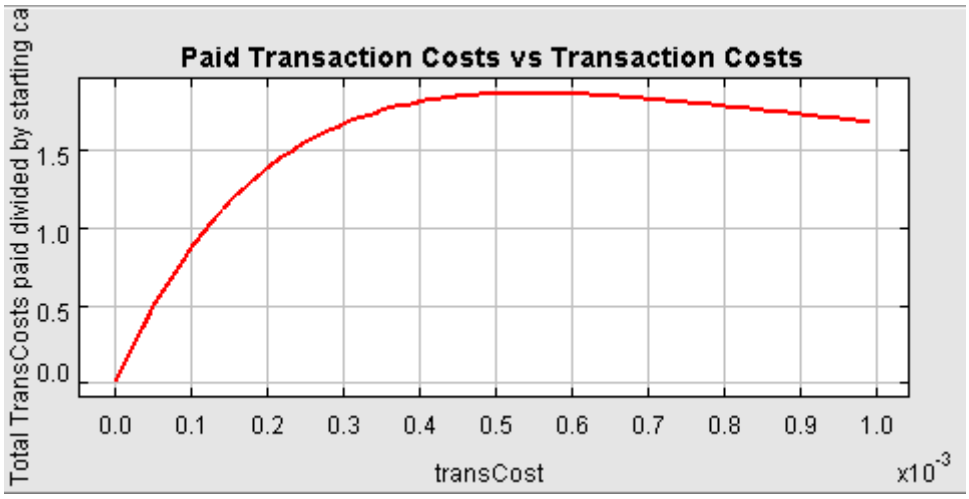

Figure 7.9: Total paid transaction costs over the duration of the trading divided by the starting capital

7.3 Compare with Random Features

We see from Figure 7.10 that the CRF features perform better using standard classification metrics. This suggest that there is something to earn from using a CRF when we want to reduce the feature space. However if we look at the profitability of each algorithm, we see that Support Vector Machines greatly outperform Random Forests and that the result differ for the two algorithms. The SVM perform better with the CRF-extracted features, while the Random Forest perform best with the random features. This could simply be at random and it could be that repeating the random selection enough times would make the results converge.

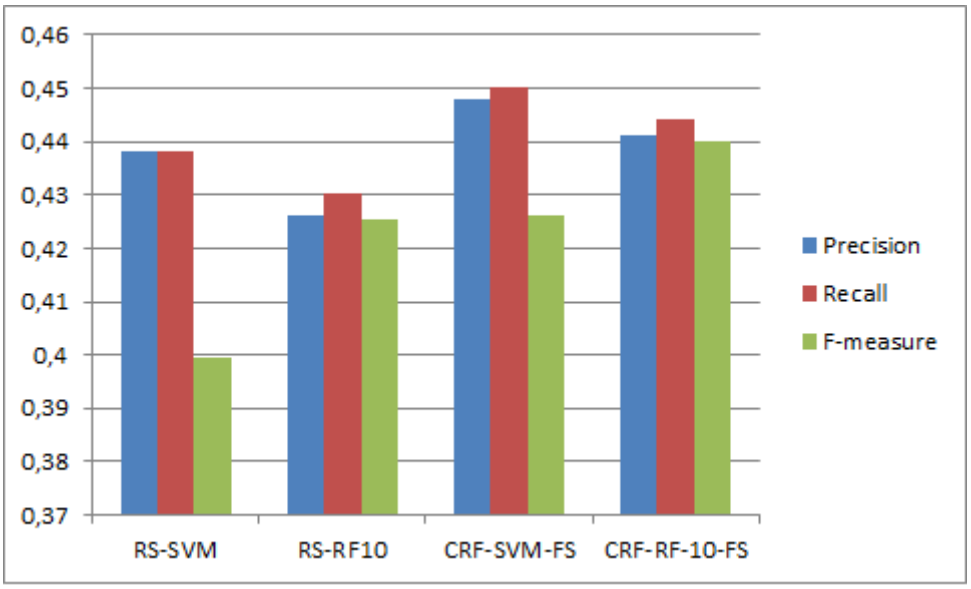

Figure 7.10: Precision, Recall and F-Measure for the positive class - Random versus CRF

Algorithm	Simulated profit
RS-SVM	58,73 %
RS-RF	38,20 %
SVM-FS	64,70 %
RF-FS-10	30,60 %

Table 7.3: Random features versus CRF-extracted features - profit

7.4 Compare to other work

[27] compares a number of different systems for trading on the stock market and it is natural to compare our system to these. Of the systems surveyed only 5 report a profit in basis points per round trip. NewsCAT [28] has the highest profit per round trip with 29 bps. The system has less than 500 trades per year, which would not yield the as high a profit as a more frequent trader system. Ænalyst [18] report a profit of 23 bps per round trip and more than 100000 trades per year. This seems very good, but [27] finds some weaknesses in the reporting, i.e. it being unlikely that as many news as reported is released for any one company as frequently as the writer reports. Another weakness reported is that only the most profitable stocks are reported. The actual profit in a real world scenario would probably be much lower. The other three systems had a profit of 13, 10 and 10 bps per round trip. Our system has at its best 10.6 bps per round trip. Our system is then comparable to these systems.

7.5 Final Thoughts

Although the simulations done indicates that the system can make a profit, one very important aspect has not been discussed. If the system wants to trade a stock instantly it will often have to pay more and when selling the same stock it would have to sell for less than the mar-

ket price. Considering this and the transaction costs, the system presented in this thesis is probably only good enough to break even.

Chapter 8

Conclusion and Further work

This chapter will conclude on the material presented in this thesis and evaluate whether the goals presented in Chapter 1 is completed. It will also present what the writer considers to be an interesting continuation of this work and how the results could be improved.

8.1 Conclusion

This thesis set out to see if reducing the features used for Support Vector Machines and Random Forest using a Conditional Random Field would reduce the performance of the classification. A data set was created that enabled analysis to be done on financial and textual data combined. A Conditional Random Field was trained and executed that found important features in the corpus. This thesis has shown that it is possible to reduce the number of features used by a Support Vector Machine without significant loss in accuracy. The simulated results in chapter 7.2 shows that the difference between using all features and using the extracted features were small. Using random features had a worse performance but also performed relatively well. The Random Forest algorithm performed almost as good with the CRF-extracted features as with all features. When it comes to using the system for trading, Support Vector Machines outperform Random Forests. Using more trees increased performance and using random features made the algorithm perform worse. The run time of the algorithms were drastically reduces with the smaller number of features. In a system meant to react instantly to news, reduction in time means increased profit.

The system perform on par with some news traders in the literature, but is not on par with the best ones. Adjusting for transaction costs and the cost of trading instantly, the system is probably only good enough to break even. Considering this being a student project and given both the time limit and the experience of the writers of other systems, this is a satisfactory result.

8.2 Further work

This thesis used a small training set for the Conditional Random Field. Some more work should be done to see if manually tagging more texts would yield a greater result. The corpus only used mandatory stock news which is a limited resource. Only about 70000 documents were found in the period, and a classification algorithm could benefit from more training examples. Including other news sources would give the system a more varied data set. The fea-

tures used in the Conditional Random Field were not numerous, only the term itself, PoS-tag and NER-tag. It would be interesting to see whether including other features would change the terms that are returned.

Support Vector Machines can be improved using a technique called boosting [29]. [4] claims that the Adaboost algorithm introduced in [29] is a form or Random Forest. For that reason, boosting was not included in this thesis, but it could be interesting to see whether using Adaboost could improve the performance of the Support Vector Machines.

As it was shown in section 7.2, the monetary gain of a system based on this work is very dependent on the transaction costs. From a more economical view, it would be interesting to analyse what trading strategy would lead to the biggest profit and to find the optimal deal to make with the broker with regard to both transaction cost and the frequency of the transaction costs.

Bibliography

[1] R. Akbani, S. Kwek, and N. Japkowicz. Applying support vector machines to imbalanced datasets. In *Machine Learning: ECML 2004*, pages 39–50. Springer, 2004.

[2] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171, 1970.

[3] L. Bottou. Une approche théorique de l'apprentissage connexionniste; applications à la reconnaissance de la parole. 1991.

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

[5] C.-C. Chang and C.-J. Lin. Training v-support vector classifiers: theory and algorithms. *Neural computation*, 13(9):2119–2147, 2001.

[6] Y.-W. Chen and C.-J. Lin. Combining svms with various feature selection strategies. In *Feature Extraction*, pages 315–324. Springer, 2006.

[7] A. Demirgüç-Kunt and R. Levine. *Financial structure and economic growth: A cross-country comparison of banks, markets, and development*. the MIT press, 2004.

[8] B. Drury and J. J. Almeida. Predicting market direction from direct speech by business leaders. *1stSymposium on Languages, Applications and Technologies*, page 163, 2012.

[9] E. F. Fama. Market efficiency, long-term returns, and behavioral finance. *Journal of financial economics*, 49(3):283–306, 1998.

[10] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[11] G. Forman. An extensive empirical study of feature selection metrics for text classification. *The Journal of Machine Learning Research*, 3:1289–1305, 2003.

[12] S. Groth and J. Muntermann. A text mining approach to support intraday financial decision-making. *Available at SSRN 1134833*, 2008.

[13] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[14] J. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. *Unpublished manuscript*, 1971.

[15] D. Harman. How effective is suffixing? *JASIS*, 42(1):7–15, 1991.

[16] T. Joachims. Making large scale svm learning practical. 1999.

[17] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

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

[19] V. Lavrenko, M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen, and J. Allan. Mining of concurrent text and time series. In *KDD-2000 Workshop on Text Mining*, pages 37–44. Citeseer, 2000.

[20] L. L. Lopes. Between hope and fear: The psychology of risk. *Advances in experimental social psychology*, 20(3):255–295, 1987.

[21] J. B. Lovins. *Development of a stemming algorithm*. MIT Information Processing Group, Electronic Systems Laboratory, 1968.

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

[23] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[24] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, volume 951, pages 591–598, 2000.

[25] M.-A. Mittermayer. Forecasting intraday stock price trends with text mining techniques. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2004.

[26] M.-A. Mittermayer. *Einsatz von Text Mining zur Prognose kurzfristiger Trends von Aktienkursen nach der Publikation von Unternehmensnachrichten*. dissertation. de, 2005.

[27] M.-A. Mittermayer and G. Knolmayer. *Text mining systems for market response to news: A survey*. Institut für Wirtschaftsinformatik der Universität Bern, 2006.

[28] M.-A. Mittermayer and G. F. Knolmayer. Newscats: A news categorization and trading system. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 1002–1007. IEEE, 2006.

[29] G. Ngai, D. Wu, M. Carpuat, C.-S. Wang, and C.-Y. Wang. Semantic role labeling with boosting, svms, maximum entropy, snow, and decision lists. In *Proceedings of SENSEVAL*, volume 3, 2004.

[30] M. H. Nguyen and F. De la Torre. Optimal feature selection for support vector machines. *Pattern recognition*, 43(3):584–591, 2010.

[31] P. Ogilvie and M. Schmill. Ænalyst-electronic analyst of stock behavior, 1999.

[32] A. Paz. *Introduction to probabilistic automata*. Academic Press, 1971.

[33] V. Punyakanok and D. Roth. The use of classifiers in sequential inference. *arXiv preprint cs/0111003*, 2001.

[34] L. Rabiner and B. Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[35] A. Ratnaparkhi et al. A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*, volume 1, pages 133–142. Philadelphia, PA, 1996.

[36] H. Shefrin. *Beyond greed and fear: Understanding behavioral finance and the psychology of investing*. Oxford University Press, 2000.

[37] I. Steinwart. Sparseness of support vector machines—some asymptotically sharp bounds. *Advances in Neural Information Processing Systems*, 16:1069–1076, 2004.

[38] C. Sutton and A. McCallum. An introduction to conditional random fields. *arXiv preprint arXiv:1011.4088*, 2010.

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

[40] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[41] V. Vapnik. *The nature of statistical learning theory*. springer, 1999.

[42] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. *Advances in neural information processing systems*, pages 668–674, 2001.

[43] Y. Zhai, A. Hsu, and S. K. Halgamuge. Combining news and technical indicators in daily stock price trends prediction. In *Advances in Neural Networks–ISNN 2007*, pages 1087–1096. Springer, 2007.