Tilburg University

Master Thesis - ORMS

Sentiment analysis using support vector machines and emotion vectors

Author:
Thom Hopmans
(ANR: 439987)

A thesis submitted in partial fulfillment of the requirements for
the degree of Master of Science in Operations Research & Management Science

Supervisor:
Dr. J.C. Vera Lizcano

Company supervisor:
M. Zevenbergen MSc

November 1, 2014
Tilburg, The Netherlands

Acknowledgments

I would like to express my deepest gratitude to my supervisor Dr. Juan Vera Lizcano. His continuous support and guidance was extraordinary and his constant encouragement and advice always led me into the right direction. Most of all, I would like to thank him for spending so much of his valuable time into this thesis.

I would like to extend my gratitude to my colleagues at the Greenhouse Group for their valuable input and inspiring discussions. Special thanks go to Menno Zevenbergen, Max den Dopper and Loc Tran for supporting me throughout the whole project, providing valuable feedback and enabling me to learn a wide range of new skills.

I am especially grateful to my family. Without their love, support and encouragement, I would never have been able to finish this thesis. Especially my loving girlfriend, who was always there for me and never complained once.

Abstract

This master thesis focuses on creating an innovative and unique sentiment analysis tool for Dutch on-line news articles. By providing the tool lots of news articles of which the sentiment is already known, the tool is able to learn to recognize sentiments from news articles using machine learning techniques. The tool uses the learned knowledge about sentiments to predict the sentiment of other news articles in terms of positive, negative and neutral.

The sentiment analysis tool in this master thesis is created by solving four different subproblems, i.e. collecting news articles, obtaining sentiment labels for the collected news articles, processing the news articles to an usable machine learning format and creating a sentiment classification algorithm. The classification algorithm is created using the support vector machine (SVM) learning algorithm, a well-studied machine learning algorithm based upon OR optimization techniques. SVMs are designed to deal with binary classification problems and small, linear separable data sets. However, the sentiment analysis problem consists of multiple classifications and the size of the data set tends to be very large-scale and complex. Therefore, this master thesis also discusses multiple techniques such as the kernel trick and multi-class heuristics to enable solving the sentiment analysis problem using the binary classification capabilities of SVMs.

Combining the solutions of the four different subproblems results in a sentiment analysis tool that provides fairly accurate and reliable predictions about the sentiment of most Dutch on-line news articles. Lastly, an example is given of the sentiment analysis tool in practice. This example shows that users reading a negative on-line news article are more likely to click on an advertisement next to the article than users reading a positive on-line news article. The result of this example may significantly affect the decision making of a company with respect to advertising strategies.

Keywords: sentiment analysis, big data, text mining, machine learning, support vector machines

Contents

1. Introduction

Big data is considered as the next big frontier for innovation, competition and productivity (Manyika et al., 2011). In almost every sector of the economy, data is generated in vast volumes. For example, the largest Dutch food retailer Albert Heijn generates lots of payment transactions every day. These payment transactions provide a massive amount of information about the products that are being bought. Additionally, Albert Heijn also uses an intelligent system to link the transactions to customer details. This system enables Albert Heijn to extract more information from the data. For example, the system can see whether a particular product or brand performs really well for customers of a specific gender in a specific age category. Moreover, these customers might be more likely to buy different related goods. Therefore, this system allows for better personalized marketing targeting and most likely results in an increase of sales because of the competitive advantage the system provides. Similar examples can also be given for social media sites, smartphones and health care industry companies. All create vast amounts of data which can be used to create value. Manyika et al. (2011) provided several more examples of how big data adds value to almost any company. For example, big data can improve transparency for stakeholders by making useful insights more accessible or by supporting human decision making through automated algorithms.

The biggest challenge of big data is how to extract information from the data. Having a large data set consisting of 350 million rows of transaction data is worthless if there are no tools available to process and analyze such a large data set. *Machine learning* is a novel field of study that combines statistical and operations research techniques to create systems that can study and learn from large data sets. Mohri et al. (2012) defined machine learning as *"computational methods using experience to improve performance or to make accurate predictions"*. In this definition, an experience is 'past' information which is available for analysis. For example, an experience could be a data set consisting of all kinds of context-related information and attached a label (classification). For the previous Albert Heijn example, the experience could be a data set consisting of characteristics of many customers (gender, age, location) and for each customer the binary label 'bought product X'. Machine learning can then be applied to obtain predictions for whether a customer outside the data set will buy product X based on the customers characteristics. Possible pitfalls such as biased data or too few data points make it difficult to find reliable relationships between customer characteristics and the purchase of product X. Therefore, the quality and the size of the experience, e.g. the data set, is absolutely critical to obtain good predictions.

1.1 Sentiment analysis

The research for this master thesis is done as part of an internship at the Greenhouse Group in Eindhoven. The Greenhouse Group is the umbrella organization for six digital marketing companies, all having a strong focus on on-line media. The Greenhouse Group wants to know how the sentiment of text can affect consumer behavior of Dutch consumers. One method to investigate how sentiment in text affects consumer behavior is by analyzing the sentiment of on-line news articles. The sentiment of these on-line news articles can then be related with and compared to various Dutch consumer behavior-related statistics. Therefore, the goal and the deliverable of the internship was to create a tool that is able to predict the sentiment of any Dutch on-line news article, i.e. a *sentiment analysis tool*. In general, sentiment analysis with respect to text documents involves classifying documents as positive, negative or neutral.

The sentiment analysis tool can be used to improve decision-making across the company and provides several interesting marketing-related purposes. For example, the tool makes it possible to determine the associated sentiment of an advertorial [1]. The tool can also be used to determine the sentiment of an on-line news article on which an ad is to be displayed. Knowing the sentiment of an on-line news article makes it possible to prevent painful combinations of advertisements and news articles, e.g. a company advertisement next to an article that is negative about that company.

The sentiment analysis tool for analyzing on-line news articles might also help analyze other types of text. This allows for a wide range of new applications, e.g. an application for sentiment analytics in customer relationship management systems. The sentiment of incoming mail of customers can be predicted and negative mails can be given a higher priority or immediately be forwarded to the correct contact person. A more futuristic example of sentiment analysis is improving automated human conversations between customers and organizations. If a computer is able to determine the sentiment of user messages, the computer can greatly reduce the number of possible replies. Such automated human conversation technology would certainly provide any company a big competitive advantage.

1.2 Research topic and problem definition

The scope of the sentiment analysis tool in this master thesis is limited to researching the relationship between sentiments and Dutch consumer behavior. More precise,

[1] An advertorial is an advertisement in the form of a news article.

the sentiment analysis tool is used to obtain a better understanding of the relationship between advertisement performance and the sentiment of on-line news articles. Chapter 10 discusses whether the click-through ratio (CTR) of an advertising campaign is higher for on-line news articles of one particular sentiment with respect to other sentiments. The general assumption is that a higher CTR results in more profit. Therefore, it is desirable to know how the sentiment of on-line news articles affects the CTR.

To investigate the relationship between advertisement performance and the sentiment of on-line news articles a tool that recognizes the sentiment of on-line news articles is required. The creation of such an accurate sentiment analysis tool is the main problem of this master thesis and is divided into multiple sub-problems. Together, the solutions of the sub-problems are used to solve the main problem. The sub-problems are roughly divided into four categories, i.e. *collecting data*, *obtaining sentiment labels*, *processing data* and the creation of a *classification algorithm*.

The first sub-problem is collecting data, i.e. creating a large data set of Dutch on-line news articles. It is assumed that the collected on-line news articles are all well-structured and grammatically and syntactically correct. The latter avoids the need of additional operations such as correcting spelling mistakes in order to obtain reliable and meaningful data. To create an algorithm that predicts the sentiment of any on-line news article, it is also important that this algorithm can first learn from other on-line news articles for which the sentiment is known. Otherwise the algorithm is not able to learn what sentiment different words contain, similar as humans for example do not know how to read a foreign language without first knowing the meaning of several foreign words. The data set from which an algorithm learns is called the *training set*. The necessity of a *sentiment-labeled* training set leads to the second sub-problem, i.e. obtaining sentiment classifications for all collected on-line news articles. These sentiment classifications are also important to check whether the predictions of the classification algorithm are correct, i.e. to check the performance of the algorithm.

The third sub-problem focuses on processing the on-line news articles to an usable sentiment classification algorithm format. It is assumed that it is possible to convert raw text into the proper algorithm format without losing much of the sentiment the on-line news articles contain. The fourth sub-problem is creating a sentiment classification algorithm that can accurately predict the sentiment of any Dutch on-line news article. Based on a relevant literature review (see Chapter 2) it is assumed that machine learning algorithms are capable for use as classification algorithm. Moreover, the same literature review states that the *support vector machine* (SVM) learning algorithm is well-suited for this task and provides state-of-the-art results. It is also assumed that creating a classification algorithm based on the articles of one (1) on-line news site is sufficient to obtain accurate predictions about the sentiment of articles from different news sites, because the articles of different news sites are similarly structured.

1.3 Approach

In order to construct the sentiment analysis tool, the four above discussed sub-problems are solved. For the first sub-problem, i.e. collecting data, ‘experiences’ are needed from which a machine learning classification algorithm can learn to make accurate sentiment predictions. In this case, these experiences consist of a large data set of on-line news articles that are given a sentiment classification. For some machine learning projects, the process of collecting data can be difficult due to time, storage or budget constraints. However, collecting on-line news articles is relatively simple, because there are a lot of on-line news sites that give free access to their news articles. The process of collecting on-line news articles is discussed in Chapter 3.1.

Obtaining sentiment classifications for on-line news articles is more challenging. All news articles can be classified manually, but when considering a data set of more than 50.000 news articles, this is not a task to do manually due to time and budget constraints. A method to obtain classified data is by letting readers of news articles do this classification task, preferably for free. Some on-line news sites allow readers to classify articles as belonging to one out of six different emotions. The result is that every article has six different emotion probabilities. Converting these emotion probabilities into one sentiment label (positive, negative or neutral) for each article is discussed throughout Chapters 7, 8 and 9.

Using a large data set of over 50.000 on-line news articles introduces several new problems. The first is encoding the articles to an usable format for machine learning algorithms. That is, every article needs to be represented as a vector of numerical features. Features can be seen as keywords representing words or phrases that contain sentiment or other textual relationships such as negations. Note that the values corresponding to these features have to be numerical, because machine learning algorithms are often based on mathematical operations that are not possible to apply on raw text. The process of obtaining numerical features is called feature extraction and is discussed in Chapter 3.2. The number of features quickly grows as the number of articles that need to be encoded increases, because there are, for example, more unique words. A small data set of Dutch news articles can easily contain more than 200.000 unique words, where every word can be seen as an unique feature. When also relationships between different words are added as features, the number of features increases drastically. A large number of features is difficult to handle effectively on an ordinary computer, because of memory constraints and scaling issues. Moreover, most of the feature values of an article are equal to zero and are thus often redundant. Therefore, the number of features must be reduced without losing too much of the information within an article. This process is called feature selection and is discussed in Chapter 3.3.

Chapter 4 focuses on applying machine learning to create an algorithm that can predict the sentiment of on-line news articles. The support vector machine (SVM) learning model is used to obtain an accurate sentiment analysis algorithm. Note that the sentiment analysis problem is basically a classification problem, i.e. assigning a sentiment classification to every news article. For classification tasks the SVM learning model is well-suited and provides state-of-the-art predictions (Cristianini and Shawe-Taylor, 2000). Although the size of the original data set is reduced significantly by applying feature selection, the resulting data set is still very large. Chapter 5 focuses on how to efficiently and effectively apply the SVM model on this large data set. This is very important in order to reduce the solving time and improve the performance of the SVM model. Chapter 6 discusses how to measure the performance of a SVM and different SVM configurations.

In Chapter 7, a 'simple' binary sentiment analysis is performed. That is, the SVM algorithm was used to predict binary sentiment labels, i.e. positive or negative, of on-line news articles. The 'simple' refers to the simple and intuitive method the sentiment of the news articles was labeled. The labeling method enabled the SVM model to learn to recognize sentiments from the labeled news articles. This simple labeling method is based on the six different emotion probabilities of each article. Using the results of the simple binary sentiment analysis, several conclusions were stated. One of these conclusions lead to the introduction of a new sentiment, i.e. the neutral sentiment. Chapter 8 explains why the neutral sentiment is an important sentiment to consider. However, the additional sentiment makes it difficult to apply the SVM machine learning techniques, because SVMs are focused on binary classification tasks. Therefore, this chapter also discusses three approaches to solve the multi-sentiment problem with three sentiments using SVMs. Chapter 9 discusses why the 'simple' labeling method used to automatically obtain sentiment labels for the news articles performs poorly. A new labeling method is proposed and the new results for the multi-sentiment analysis are analyzed and evaluated.

Chapter 10 provides a practical application of the sentiment analysis tool. This practical application discusses the hypothesis that the sentiment of an on-line news article affects the click-through rate of an advertisement shown next to the news article. In Chapter 11 a summary of all the results is given and the final conclusions are stated. Lastly, an overview of possible future research topics is provided.

2. Previous work

The wide range of possible applications of sentiment analysis makes sentiment analysis an increasingly popular topic of research. Initial work on sentiment analysis was based on text categorization techniques. Early examples include categorizing text by author or news paper (Argamon-Engelson et al., 1998) and recognizing the genre of text in terms of informative and non-informative or fiction and non-fiction (Karlgren and Cutting, 1994). Turney (2002) is considered as one of the first that used basic text categorization techniques for sentiment analysis purposes. His work focused on classifying the sentiment of movie reviews by polarity, i.e. in terms of positive and negative. Pang et al. (2002) were the first that proposed several text categorization machine learning techniques such as support vector machines to solve the movie review sentiment analysis problem.

Pang et al. (2002) provided the start of a wide variety of research on performing sentiment analyses using machine learning techniques such as support vector machines. The majority of the research can be divided into four categories. One category focuses on finding the best machine learning technique to solve a particular sentiment analysis problem. The SVM algorithm turns out to consistently provide a high performance for solving sentiment analysis problems (Pang et al., 2002; Ye et al., 2009). The second category of research focuses on finding the best methods to translate text into usable machine learning formats such as numeric feature vectors (Melville et al., 2009; Yang and Pedersen, 1997). The third category of research focuses on performing a sentiment analysis for a particular source of data using support vector machines. Initially this category focused on extracting sentiment from sources such as movie reviews (Mullen and Collier, 2004) or customer reviews (Gamon, 2004). However, the rise of social media such as Twitter and Facebook created a whole new source of easily collectible text that often contain a clearly-defined sentiment. Recent research therefore often focuses on extracting sentiments from data sources such as Tweets using SVMs (Go et al., 2009; Pak and Paroubek, 2010). The last category focuses on researching the support vector machine learning technique. Relevant research in this category is discussed throughout the whole master thesis. However, Chapter 4 and 5 focus especially on the support vector machine learning technique and these chapters discuss multiple relevant research papers.

Turney (2002) and Pang et al. (2002) both applied sentiment analysis on a document level, i.e. they determined the sentiment of the whole text. More recent research extended the scope of sentiment analysis to sentence or word level. One interesting recent research project on sentiment analysis at sentence/word level is done by Socher et al. (2013). Socher et al. (2013) presented the 'Stanford Sentiment Treebank', a database containing 215.154 unique phrases extracted from 11.855 sentences. The big and unique advantage of this treebank is that all 215.154 phrases are labeled on a fine-grained scale from very negative to very positive by three different human persons. Furthermore, the community is allowed to add more sentiment scores or phrases. To analyze the sentiment of any sentence using the approach of Socher et al. (2013), the sentence is first chopped down to individual words. Using a new recursive neural network machine learning technique, these individual words and corresponding sentiment scores are combined to obtain an overall sentiment for the whole sentence. The sentiment predictions of this new sentiment analysis approach turned out to be very accurate and very good at capturing negations in sentences.

Note that all mentioned previous work on sentiment analysis is applied on English documents. With respect to the Dutch language there is very few scientific work on applying sentiment analysis on Dutch documents and no scientific work on applying sentiment analysis on Dutch on-line news articles. The latter makes the results of this thesis very interesting as it is unique in its kind.

3. Data

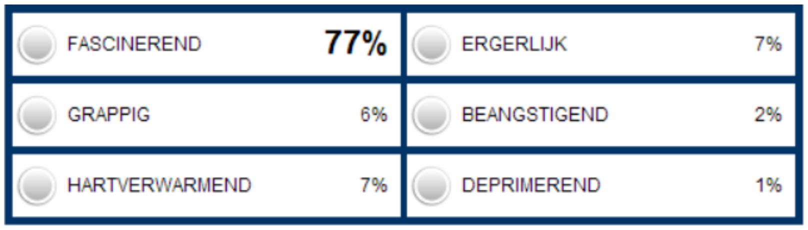
In order to construct a sentiment analysis tool using machine learning techniques, data is needed from which a machine can learn to recognize sentiments. In this chapter, it is discussed how the necessary data was collected and how it was labeled in terms of a sentiment. Moreover, several methods to reduce the collected data to an usable machine learning format are discussed. The latter was done using the concepts of feature extraction and feature selection.

3.1 Obtaining labeled data

To let a machine learn to recognize sentiments from on-line news articles, it is necessary to provide the machine a large collection of on-line news articles. However, the machine also needs to know what sentiment the news articles contain. Otherwise, the machine is not able to learn what sentiment different words contain, similar as humans for example do not know how to read a foreign language without first knowing the meaning of several foreign words. Therefore, the large collection of on-line news articles requires that each article is labeled as a particular sentiment, i.e. positive, negative or neutral. Therefore, several steps were performed to create the necessary large set of sentiment-labeled on-line news articles.

The first step was to obtain a large collection of on-line news articles. A nice property about on-line news articles with respect to other sources, such as Twitter messages, is that the on-line news articles are often well-structured and grammatically and syntactically correct. Several software tools are available for collecting content from internet web pages. For this project, with the help of colleagues these tools were adjusted such that is was possible to collect news articles from major Dutch news sites. The result was a data set consisting of more than 2 million on-line news articles, dating from 2000 to 2014.

The second step, labeling the collected on-line news articles based on their sentiment, was more difficult. One ‘simple’ option would be to manually label all news articles. However, this drastically increases the amount of work and therefore significantly reduces the number of articles that can be labeled due to time or budget constraints. One of the major Dutch news sites, however, provides a method to obtain the emotions that readers associate with an article. This news site shows next to each article a graphical user interface (GUI) in which readers can select the emotion of the article. To be precise, readers can classify a page as 'fascinerend' (fascinating), 'grappig' (funny), 'hartverwarmend' (heart-warming), 'ergerlijk' (annoying), 'beangstigend' (frightening) and 'deprimerend' (depressing). A useful property of the GUI is that it also presents for each emotion the percentage of readers that classified the news article as that emotion. It is assumed that these voting percentages denote the emotion probabilities of an article. An example of the GUI is given in Figure 3.1. Note that in this example, 77 % of the readers classified this news article as fascinating.



FASCINEREND	77%	ERGERLIJK	7%
GRAPPIG	6%	BEANGSTIGEND	2%
HARTVERWARMEND	7%	DEPRIMEREND	1%

Figure 3.1: Example of the GUI readers can use to vote for the emotion they associate to a news article.

The above GUI partly solves the problem of how to obtain labeled on-line news articles, because this is now done for free by the readers of the articles. Unfortunately, these emotion probabilities are not flawless and introduce a certain degree of uncertainty. For example, readers might not vote their true emotion due to the sensitivity of the articles content, e.g. political articles, or readers can deliberately vote a different sentiment (called 'trolling'). The question is how to obtain a sentiment label , i.e. positive, negative or neutral, from the emotion probabilities. Particularly challenging though is how to interpret sentiments. What one person considers as a positive article, someone else could consider as a negative article. That is, sentiments are individual and personal. The 'simplest' way is to label the message as the sentiment with the highest probabilities of positive or negative emotions. This simple method however raises the question which emotions are positive or negative and in what amount the emotions contribute to the sentiment. The problem of how to obtain a sentiment label from the six emotion probabilities is further investigated in Chapter 7.1.

3.2 Feature extraction

The collected on-line news articles first need to be converted to a proper and usable machine learning format, before a machine learning algorithm can be applied. Most machine learning algorithms only work with numerical input vectors of a fixed size. Therefore, the news articles, consisting of large phrases of words and punctuation, need to be converted to numerical vectors of a fixed size without losing the content of the article in the process. To obtain vectors of a fixed size, it is necessary to create 'features', e.g. measurable article properties. In text analysis, a feature often refers to a word, a phrase, a number or a symbol. All news articles are then measured in terms of these features, resulting in numerical vectors of the same size for all news articles.

The whole process of converting text to numerical vectors is called feature extraction and is often done in three steps: *tokenization*, *counting* and *weighting*. In the remainder of this section, the three steps and possible approaches are described. Note that feature extraction is a multi-disciplinary technique that requires both linguistic and statistical techniques. The approaches described in this section are largely based on research by Berry and Kogan (2010) and O'Keefe and Koprinska (2009).

3.2.1 Step 1: tokenization

The first step in obtaining numerical features is *tokenization*. In text analysis, tokenization is described as obtaining meaningful basic units from large samples of text (Webster and Kit, 1992). For example, in physics speed can be expressed in meters per second. In text analysis, large strings of text can be expressed in terms of basic units called tokens. Often, these tokens correspond to words. Therefore, a simple tokenization method to obtain tokens for the sentence 'I am happy, because the sun shines' is by splitting them on whitespaces. This splitting results in seven tokens, i.e. 'I', 'am', 'happy,', 'because', 'the', 'sun', 'shines'. It is now possible to express the original sentence in terms of these tokens.

This simple tokenization method however provides several new problems. For one, this method does not filter out any punctuation and thus the token 'happy,' contains a comma at the end. This implies that the token 'happy**,**' and 'happy' are two different tokens, although both tokens contain the same sentiment in sentiment analysis. Therefore, all types of punctuation are filtered out, because punctuation is almost never relevant for the sentiment in on-line news articles. Note that it would have been relevant if the text would contain smiley faces, because in that case punctuation combinations like ':-)' contain a lot of sentiment. Therefore, if the source of the data is for example Twitter messages, punctuation needs to be filtered out in a more intelligent way. The smiley face example emphasizes the fact that every data source needs its own feature extraction method.

Second, using the simple tokenization method it is possible to obtain the tokens 'works' and 'working'. However, these tokens are just different forms of the same word, i.e. 'to work'. The same argument holds for tokens where one is the plural form of the other. For the sentiment analysis, it is assumed that tokens originating from the same word contain the same amount of sentiment. Therefore, the tokens can be reduced to their stem and used as a single token. To do this, a stemming algorithm that reduces every word to its stem is required. Note that such a stemming algorithm is language specific. Unfortunately, most stemming algorithms are focused on the English language, because an English stemming algorithm is commercially more interesting than a Dutch stemming algorithm. Therefore, a simple Dutch stemming algorithm was created based on a Dutch dictionary containing the stem of the majority of the Dutch words.

The third problem that arises is how to deal with combinations or negations of words. For example, just using the individual tokens ‘happy’ and ‘not’ may not completely cover the more strongly negative sentiment of the combination ‘not happy’. Similarly, the individual tokens ‘very’ and ‘happy’ may not completely cover the more strongly positive sentiment of the combination ‘very happy’. Therefore, instead of creating tokens of only one word, also tokens of two or three consecutive words are used, called respectively bi-grams and tri-grams. Using also bi- and tri-grams, the sentence *‘I am not happy’* translates to the tokens *‘I’, ‘am’, ‘not’, ‘happy’, ‘I am’, ‘am not’, ‘not happy’, ‘I am not’* and *‘am not happy’.* It is assumed that the token *‘not happy’* has a much more negative impact on the sentiment than the token *‘happy’* has a positive impact. Therefore the entire sentence is classified as negative when the token *‘not happy’* is present, even though the presence of the positive token *‘happy’.*

Note that this tokenization method does not take into account the position and the order of the words. For example, after tokenization it can not be said at what position in the original sentence or article the token *‘happy’* occurred. Also, the token itself does not mention anything about the words in front or after it. There are other tokenization methods which take the position and order of words into account as well. For example, Part-Of-Speech (POS) tagging also adds additional information such as the word-class of a token, e.g. whether a token occurs as a verb, adjective, noun or direct object. However, this requires the use of a POS-tagging algorithm which is a very complex machine learning problem itself. Moreover, most POS-tagging algorithms are developed for the English language. Therefore, using POS-tagging was considered out of the scope for this thesis, but is a very interesting future research topic.

3.2.2 Step 2: counting

In the second step, the frequency of each token in every article is counted. These frequencies are used in the next step for weighting but also in Chapter 3.3.1 to perform a basic feature selection, i.e. to reduce the number of features. Note that a typical property of text analysis is that the majority of the tokens are only used in a couple of articles. Therefore, the frequency of most tokens for an article is zero.

3.2.3 Step 3: weighting

In the last step, the tokens and token frequency counts from the previous steps are used to convert all articles to an usable numerical machine learning format. This is done by encoding each article to a numeric vector whose elements represent the tokens from step 1. Moreover, a token weighting procedure is applied using the frequency counts from step 2. After a token is weighted, it is not any more referred to as a token but as a **feature**. Hence, for the remainder of this thesis, a feature represents a token which is extracted from the large set of on-line news articles and weighted according to a given weighting method based on the token frequency counts.

There are several methods to determine the weight for each token. The most basic weighting method is *'Feature Frequency'* (FF). FF simply uses the frequency of a token in an article as the weight for a token. For example, given the token set $\{mad,\ happy\}$, the sentence *'I am not mad, but happy, very happy'* is represented as the vector $\begin{bmatrix}1 & 2\end{bmatrix}$.

A more complex feature weighting procedure is the *'Term Frequency and Inverse Document Frequency'* (TF-IDF) weighting method. This method uses two scores, i.e. the term frequency score and the inverse document frequency score. The term frequency score is calculated by taking the frequency of a token in an article. The inverse document frequency score is calculated by the logarithm of dividing the total number of articles by the number of articles in which the token occurs. When multiplying these two scores, a value is obtained that is high for features that occur frequently in a small number of news articles, and is low for features that occur often in many news articles.

O'Keefe and Koprinska (2009) have shown that using *'Feature Presence'* (FP) gives the best results in sentiment analysis. With FP, the weight of a token is simply given by a binary variable which is 1 if the token occurs in an article and 0 otherwise. The sentence from the previous example would be represented as the vector $\begin{bmatrix}1 & 1\end{bmatrix}^T$ when using FP, because both tokens are present in this sentence. An additional advantage of FP as weighting method is that a binary data set is obtained, which does not suffer scaling problems. The issue of data scaling is further explained in Chapter 4.4.1. FP is used as the feature extraction weighting method in the remainder of this thesis, because of the conclusions of O'Keefe and Koprinska (2009) and the advantages of a binary data set.

3.3 Feature selection

In text classification, a good feature selection method is very useful for improving the classification task in terms of prediction quality and solution time (Berry and Kogan, 2010). In sentiment analysis, features selection implies selecting features based on the amount of contained sentiment. Features that hardly contain sentiment can be removed from the feature set to significantly reduce the size of the data set and to improve the speed of the machine learning algorithms. In this section four different feature selection methods are discussed. In Chapter 7 the performance of the four different feature selection methods is measured and compared.

3.3.1 Document Frequency selection

Document Frequency (DF) selection is the simplest feature selection method and is a must for many text analysis problems (Berry and Kogan, 2010). DF selects features within an unsupervised setting, i.e. without taking into account the sentiment label of an article. One common approach is by first removing the most common Dutch words, e.g. the Dutch words 'de', 'het' ('the') and 'een' ('a'). After that, all features with a very high document frequency are removed from the data set as these features are not likely to help in differentiating articles. Similar, all features with a very low document frequency are removed as they can make the classification worse. The latter can happen when a feature, e.g. the word 'hippopotamus', only occurs once in the set of all news articles, e.g. once in a negative article. Then this feature is likely to get a negative sentiment impact by most machine learning algorithms, even though most people do not consider the word hippopotamus as negative. Also, the sentiment of future news articles containing this feature is affected negatively due to this word. Hence, it is desirable to filter out all features with a very low or very high document frequency.

3.3.2 Information Gain

The Information Gain (IG) method selects features based on the distribution of a feature's frequency over the different sentiment labels. This is a supervised feature selection procedure, because it takes into account the sentiment label of the articles. IG calculates the amount of information gained about the classification of an article by using the presence or absence of a feature in a message. The assumption is that the higher the information gain value of a feature, the better that feature is to predict the sentiment of a news article.

The result of IG is a list of features that distribute very differently between the sentiments. For example, the word 'happy' is likely to occur much more in articles with a positive sentiment than in articles with a negative sentiment. Therefore, the word 'happy' is a good feature to use to recognize positive articles and much information is 'gained' about the sentiment of an article. Hence, IG is likely to select this feature.

The IG score of the feature f about the classification c is calculated by

$$IG(f,c) = \sum_{c' \in \{c,\ \bar{c}\}} \sum_{f' \in \{f, \bar{f}\}} p(f',\ c') log \left\{ \frac{p(f',\ c')}{p(f')p(c')} \right\},$$

where

- p(c) = probability that a news article has (sentiment) classification c,
- p(f) = probability that a feature f occurs in a message,
- p(f, c) = probability of f & c occurring,

- $\bar{c}$ represents the complement of classification c, i.e. p($\bar{c}$) is the probability of not classification c occurring,
- $\bar{f}$ represents the complement of feature f, i.e. any feature but not feature f.

Note that all probabilities can be estimated by using the frequency of features and sentiment labels in the data set. A high IG score of a feature f given a classification c means that the feature f contains much information about the classification c of an article. Vice versa, a low IG score implies that the feature f contains few information about that classification c of an article. Therefore, it is desirable to only select the features with the highest IG scores. This significantly reduces the number of features, while keeping the features that contain a lot of sentiment and thus provide much information about the classification of an article.

The number of features to select is still an issue though. Similarly as in the DF feature selection method, keeping features that do not contain much sentiment can make the performance of the classification algorithm worse. Vice versa, removing too many features implies that features that contain much sentiment are removed, making it more difficult for the algorithm to learn recognize sentiments. Also, the less features, the shorter the time the classification algorithm needs to find a good classifier function. Therefore, there is a trade-off between the number of features and the performance of the classification algorithm.

It is assumed that the 'optimal' number of features, i.e. the number of features that gives the highest classification performance, lies somewhere in the middle (Yang and Pedersen, 1997). To solve this issue, Chapter 7 compares the results of selecting different numbers of features. Comparing the performance of the classification algorithm using the different configurations provided more insight in the 'optimal' number of features to select.

3.3.3 χ^2 statistic

Similar as the IG method, the χ^2 statistic is a supervised feature selection method. The χ^2 statistic is used to measure the independence between a feature and a classification (Yang and Pedersen, 1997). The χ^2 statistic value of the feature f and the classification c is calculated by

$$\chi^2(f,c) = \frac{N \times \left[p(f,c) \times p(\bar{f},\bar{c}) - p(f,\bar{c}) \times p(\bar{f},c)\right]^2}{p(f) \times p(\bar{f}) \times p(c) \times p(\bar{c})}$$

and

$$\chi^2(f) = \frac{1}{|C|}\sum_{i=1}^{|C|} \chi^2(f,c_i),$$

where the notation is identical to the notation in the IG method. A high value for the $\chi^2(f)$ statistic implies a high dependence of the feature f to one of the classifications. Therefore, it is desirable to only select the features with the highest χ^2 statistic values. However, similar as with IG, the 'optimal' number of features to select can not be calculated directly. Therefore, Chapter 7 compares the classification algorithm performance of selecting different numbers of features.

3.3.4 Random Projection

Random Projection (RP) is an unsupervised dimensionality reduction method and is used in several applications where the high-dimensionality of the data leads to computational problems (Bingham and Mannila, 2001). RP can also be used for feature selection in text classification problems, because text analysis problems also quickly become high-dimensional due to the large number of features. To reduce the dimensionality of a matrix, RP projects the matrix to a lower-dimensional space, while trying to contain as much of the variation in the matrix as possible. The result is that there is a trade-off between the accuracy of the classification algorithm and the size of the problem. The more random projections are applied, the lower the dimensionality and the smaller the problem size will be at the cost of a lower accuracy. A smaller problem size results in less computational work and thus the solving time of the classification algorithms significantly reduces. Moreover, less memory is needed to store the problem and relevant calculations. Li et al. (2006) were also able to improve RP by applying sparse random projections. As a consequence, the resulting lower-dimensional matrix is sparse and the computational effort and memory needed for RP was significantly reduced.

A particular useful property of RP is that the *Johnson-Lindenstraus lemma* makes it possible to control the decrease in accuracy (Johnson and Lindenstrauss, 1984). The lemma states that it is possible to embed any n row matrix in $O(\log n\ \epsilon^{-2})$ columns while the Euclidean distances between the rows are not changed more than a factor $1 \pm \epsilon$. This is a useful property, because many classification algorithms work with the Euclidean distances between points. Also, using this lemma, it is possible to calculate the minimum number of columns, i.e. features, needed to obtain an embedding with an accuracy of ϵ.

Table 3.1 provides an example of this lemma for the numeric feature representation of the on-line news articles data set. From this table it follows that, as the value for ϵ gets smaller, drastically more features are needed to ensure a random projection embedding satisfying accuracy ϵ. It is important to note though that this lemma only provides a general rule on the loss of accuracy. It could still be possible to achieve the same (or less) loss in accuracy when using less than the minimum number of samples and features provided by this lemma. However, there is no guarantee.

It is important to note that the sentiment the features contain is not important when applying RP. In contrast to IG and the χ^2 statistic, RP does not look whether a feature contains few or much sentiment. Therefore, RP does not eliminate the possi-

	Number of news articles (n)		
ϵ	100	1.000	10.000
0.01	370.886	556.329	741.772
0.1	3.947	5.920	7.894
0.5	221	331	442

Table 3.1: The minimum number of features needed to ensure that a random projection embedding changes the original Euclidean distances between the articles not more than a factor $(1 \pm \epsilon)$. The minimum number of features is given for various combinations of the parameters n and ϵ.

ble classification performance problems of using features that contain few sentiment. It might be wise to apply RP *after* filtering out features that do not contain much sentiment using the IG or χ^2 statistic feature selection method. Lastly, note that the sparsity and binarity of the numeric feature representation of the news articles data set is lost when using RP. The data set is still sparse, but the data set might suffer scaling problems due to not being binary. Chapter 4.4.1 discusses the necessity of scaling the data.

3.4 Final notes on feature extraction and selection

As a last remark on this chapter it is stated that in any text classification application the feature extraction and feature selection method have a very big impact on the final results (Berry and Kogan, 2010). For example, a bad feature selection method implies that features that contain a lot of sentiment are being removed. This can lead to the classification algorithm failing to find the correct sentiment of an article, making future sentiment predictions inaccurate.

Moreover, feature extraction and selection requires a multidisciplinary approach as it combines techniques from different fields such as statistics and linguistics. As the focus of this thesis is not developing a new feature extraction or selection method, the methods discussed in this chapter are used for the remainder of this thesis.

To be more precise, FP is used as feature extraction weighting method in the remainder of this thesis, because FP is the only discussed feature extraction method that transforms the on-line news articles data set in a sparse and binary matrix. Chapter 4.4.1 discusses why sparsity and binarity of the data is desirable. Additionally, Chapter 7 compares the four discussed feature selection methods in terms of classification performance.

4. Support Vector Machine

Support vector machine (SVM) is nowadays a state-of-the-art machine learning technique and is used in many real-life classification problems (Cristianini and Shawe-Taylor, 2000). For example, SVMs are often used for all kinds of recognition problems such as handwriting, speech, faces and objects in images. SVMs are also used for weather and financial time series forecasting and for less obvious studies such as studying the effects of different health indicators on the probability of diseases (Kecman, 2001). This chapter starts with a detailed introduction on SVMs, followed by an extensive analysis on the different methods to model the SVM learning problem as an OR optimization problem. The chapter is concluded by stating several remarks on 'training' SVMs.

4.1 Introduction on SVMs

Performing a sentiment analysis is basically solving a text categorization problem. That is, given a set of sentiments, each text needs to be categorized as one of the possible sentiments. Therefore, the sentiment analysis problem is a classification problem where the goal is to classify every text as a particular class, e.g. a sentiment. The challenge of most classification problems is to find a well-performing classification rule, i.e. a function that takes any news article as input and returns the correct sentiment the article contains.

From the previous chapter it follows that, using the numerical feature representation, every on-line news article can be represented as a point in a D-dimensional space, where D is the number of features. For example, consider a data set consisting of fourteen news articles ($N = 14$) with two features ($D = 2$) and two possible sentiment classifications. Also, assume that the first seven articles have an opposite sentiment than the last seven. The news articles can now easily be plotted in a 2-dimensional space as is done in Figure 4.1. Using this figure, it can also be seen that any line that divides the sample points of both sentiments can be used as a classification rule, because all points on one side of the line are classified as one sentiment, whereas all points on the other side are classified as the opposite sentiment.

Note that this line corresponds to a hyperplane when the dimension, and thus the number of features, is greater than 2. Because both sides of the hyperplane represent a different classification, this hyperplane is often denoted as a *separating hyperplane*, a *decision boundary*, a *classifier* or as a *classification rule*. Any new sample point is classified based on which side of the separating hyperplane the point lies. Therefore, a separating hyperplane is very useful for quickly making predictions about the class of new samples.

A SVM tries to find the *best* separating hyperplane, i.e. the best hyperplane that separates the points of two classes. The question is how to define the best separating hyperplane. In most cases, there are multiple hyperplanes possible that separate the two classes. Figure 4.1 gives an illustration of multiple separating hyperplanes. The problem of finding the best separating hyperplane was first discussed by Vapnik in 1965 (Vapnik and Kotz, 1982) and his work is nowadays considered as the starting of SVMs. Cortes and Vapnik (1995) provided an improved version of the original research. They stated that the best separating hyperplane is the hyperplane that *maximizes the margin* between the two classes. Figure 4.2 and Figure 4.3 provide graphical examples of separating hyperplanes that maximize the margin between two classes.

A sentiment analysis algorithm to predict the sentiment of any on-line news article using SVMs is created in two steps. In the first step, a SVM is *trained* by finding the best separating hyperplane for a given set of samples, i.e. on-line news articles. This set of samples is called the *training set*. The separating hyperplane is defined by $H = \{x : w^T x = b\}$ or by $H = \{x : w^T x + b = 0\}$ [1]. Training a SVM is thus done by finding the values for the vector w and the scale parameter b such that the resulting separating hyperplane maximizes the margin between two classes.

In the second step, the trained SVM is used to make predictions about the sentiment of new samples that were not in the training set. The sentiment predictions of new sample points are made by using the best separating hyperplane from the first step. This is generalized for any sample point x by stating that if $w^T x + b \geq 0$, the article is classified as class '+1', e.g. positive. Otherwise, if $w^T x + b < 0$, the article is classified as class '-1', e.g. negative. The predicted sentiment classification of article i is denoted by $\hat{y_i}$ and is calculated by $\hat{y_i} = \text{sign}\{w^T x_i + b\}$. A separate set of samples that were not in the training set is often used for testing the performance of the SVM. Therefore, this set of samples is called the *testing set*. Chapter 6 explains the importance of separate training and testing sets.

4.2 Training the SVM using a linear classifier

Training a SVM, i.e. finding the best separating hyperplane, is done by using a wide range of OR optimization techniques. By the end of this chapter, the problem of finding the best separating hyperplane is formulated as an optimization problem. However, first of all, a clear definition about the margin between two classes is needed.

[1] It does not matter whether $+b$ or $-b$ is used, because there is no restriction on the sign of b.

4.2.1 Defining the margin

Consider a data set consisting of samples that are classified as only two different possible classifications, i.e. a positive classification with label +1 and a negative classification with label -1. Also, assume that the sample points of this data set are *strictly linear separable*. Strictly linear separable means that there always exists at least one separating hyperplane that exactly separates the sample points of the two different classes and no sample points are lying on this hyperplane.

Figure 4.1 provides an example of a strictly linear separable data set with three possible separating hyperplanes, i.e. $H1$, $H2$ and $H3$. In this figure, all three hyperplanes perfectly separate the sample points of both classifications. However, problems may occur when using hyperplanes $H1$ and $H2$ to make predictions about the classification of any new sample point. For example, consider a new sample point with classification 'sentiment 2' (circle) that lies at coordinate $\begin{bmatrix}3 & 1\end{bmatrix}$. Note that this new sample point lies very close to the existing sample points of the 'sentiment 2' class, and therefore this new sample point intuitively also belongs to the 'sentiment 2' class. However, when using $H1$ as separating hyperplane, this new sample point falls on the other side of the hyperplane and is therefore classified as 'sentiment 1'. Using $H1$ as separating hyperplane thus gives inaccurate classification predictions. A similar example can also be created for $H2$. Therefore, Cortes and Vapnik (1995) defined the best separating hyperplane to be the hyperplane that maximizes the distance between the separating hyperplane and the nearest sample points of both classes. By this definition, the hyperplane $H3$ in the example in Figure 4.1 is the best separating hyperplane.

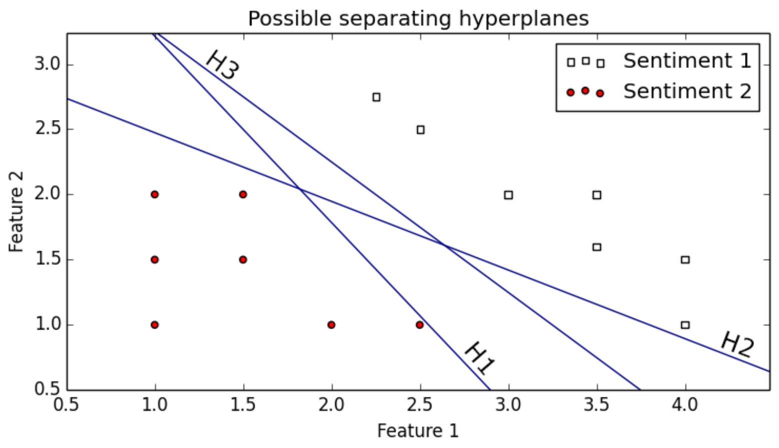


Figure 4.1: Example of how two classes (square and circle) can be separated in three different ways using three different hyperplanes. The question is how to find the 'best' separating hyperplane.

Assume the sample point that is nearest to the separating hyperplane $H_{w,b}$ is denoted by x. The Euclidean distance between $H_{w,b}$ and x can be seen as the *margin*

M for which it holds that no sample points lie within this distance to $H_{w,b}$. Also, the value of the margin M provides a minimum bound on the distance between the points of both classes that are nearest to each other. The best separating hyperplane $H3$ of the example in Figure 4.1 and its margin M are also given in Figure 4.2. In addition, Figure 4.3 provides a similar example.

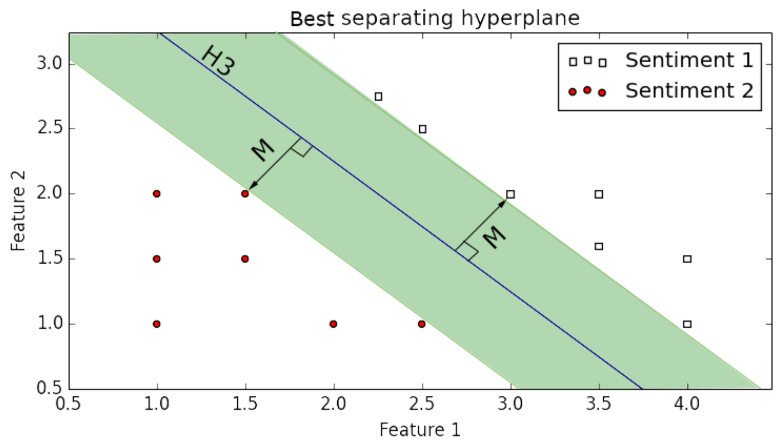


Figure 4.2: The best separating hyperplane $H3$ of the example in Figure 4.1 and its margin M.

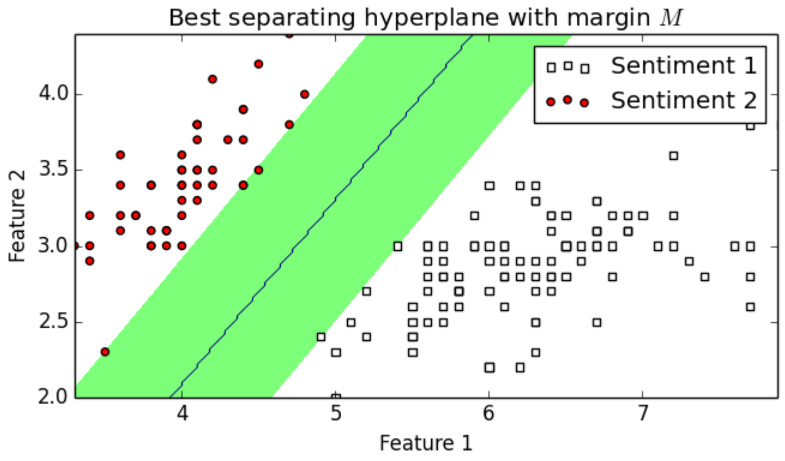


Figure 4.3: A graphical example of a separating hyperplane for which the margin around the separating hyperplane is maximized. The accentuated area is the margin M around the separating hyperplane.

4.2.2 The hard-margin SVM optimization problem

The size of the margin M is dependent on the separating hyperplane $H_{w,b}$ and the margin is therefore denoted by $M = M(w,\ b)$. The Euclidean distance between $H_{w,b}$ and any point x is denoted by $D_{w,b}(x)$. It follows that the size of the margin M is given by

$$M = M(w, b) = \inf_{x \in X} D_{w,b}(x), \tag{4.1}$$

where X is the matrix containing all sample points x_i. In the sentiment analysis problem, the matrix X thus contains the numeric feature vector representation of all N on-line news articles. The best separating hyperplane is the separating hyperplane that maximizes the margin (Cortes and Vapnik, 1995). Finding the best hyperplane can then be formulated as an optimization problem as follows

$$\max_{w,b,\ x \in X} \inf D_{w,b}(x). \tag{4.2}$$

Solving the model in Equation (4.2) gives the best separating hyperplane. However, the solution is not unique as there are multiple best separating hyperplanes with the same value for the margin M. For example, consider the separating hyperplane $-2x_1 - 2x_2 + 8 = 0$. This hyperplane can be multiplied by any constant to obtain the **same** separating hyperplane, but then given by different parameters w and b. Therefore, the separating hyperplanes are *scale invariant*. To ensure only one unique best separating hyperplane is obtained, a *scaling constraint* is necessary and is given by

$$|w^T x_n + b| = 1, \tag{4.3}$$

where x_n is the point that is nearest to the separating hyperplane. Note that it is always possible to make the scaling constraint hold without affecting the separating hyperplane, because the parameters w and b can be scaled such that the scaling constraint holds. Additionally, because of the scale invariance property, the scaling constraint does not affect the resulting best separating hyperplane.

Also note that the vector w is always perpendicular to the hyperplane. For example, take any point y' and y'' on the separating hyperplane. Then it holds that

$$w^T y' + b = 0 \text{ and } w^T y'' + b = 0 \ \Rightarrow w^T(y' - y'') = 0.$$

The perpendicular property is used to calculate the distance between any point x and the separating hyperplane. Consider the sample point x and a point y on the separating hyperplane. It was shown that the vector w is always perpendicular to the separating hyperplane for any point y on the separating hyperplane. However, the vector $x-y$ is not always perpendicular to the separating hyperplane. Therefore, the Euclidean distance between the points x and y is not always equal to the Euclidean distance between the separating hyperplane and the sample point x. A projection

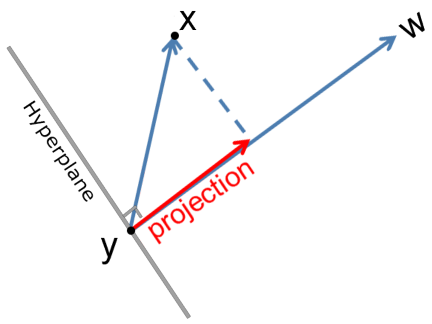


Figure 4.4: Projection of the vector $x - y$ on the vector w.

of the vector $x - y$ onto the vector w is used to obtain the distance between the separating hyperplane and any sample point x in terms of w. Figure 4.4 gives a graphical illustration of this projection. Using basic geometry and algebra rules, the projection of $x - y$ on the perpendicular vector w is calculated by

$$\text{projection} = \frac{w^T}{||w||}(x - y), \tag{4.4}$$

where $\frac{w}{||w||}$ is the unit vector of w. The projection in Equation (4.4) is used to calculate the distance of any point x to the separating hyperplane $H_{w,b}$ by

$$D_{w,b}(x) = \left|\frac{w^T}{||w||}(x - y)\right| = \frac{1}{||w||}\left|w^T(x - y)\right| = \frac{1}{||w||}\left|w^T x + b\right|.$$

Note that by definition of x_n and strict linear separability it also holds that the distance of any other point x_i is at least greater than or equal to 1. So $D_{w,b}(x_i) = |w^T x_i + b| \geq 1$ for all $i = 1, ..., N$. Therefore, the scaling constraint in Equation (4.3) is rewritten to

$$\min_{i=1,...,N} |w^T x_i + b| = 1$$

and the margin in Equation (4.1) is simplified as follows

$$\begin{aligned} M = \inf_{x \in X} D_{w,b}(x) = \inf_{x \in X} \frac{1}{||w||}\left|w^T x + b\right| = \frac{1}{||w||} \min_{i=1,..,N} \left|w^T x_i + b\right| \\ = \frac{1}{||w||}. \end{aligned}$$

The optimization problem in Equation (4.2) can then be rewritten to

$$\begin{aligned} &\max \frac{1}{||w||} \\ &\text{ s.t. } \min_{i=1,...,N} |w^T x_i + b| = 1 \\ &\qquad w \in \mathbb{R}^D,\ b \in \mathbb{R}. \end{aligned} \tag{4.5}$$

Note that the scaling constraint also puts a bound on the objective function. Without the scaling constraint, the vector w can be set such that the objective goes to zero.

Using the assumption that the data set is strictly linear separable, it follows that all points are classified correctly by the best separating hyperplane. Assume all sample points with classification label $y_i = +1$ lie on the side of the separating hyperplane for which it holds that $w^T x_i + b \geq 0$. Vice versa, all points with classification label $y_i = -1$ lie on the side of the separating hyperplane for which it holds that $w^T x_i + b < 0$. Then, it holds that

$$|w^T x_i + b| = y_i(w^T x_i + b). \tag{4.6}$$

Also, note that maximizing $\frac{1}{||w||}$ is equivalent to

$$\max_w \frac{1}{||w||} \Leftrightarrow \min_w ||w|| \Leftrightarrow \min_w ||w||^2 \Leftrightarrow \min_w \frac{1}{2}||w||^2 \Leftrightarrow \min_w \frac{1}{2} w^T w \tag{4.7}$$

Hence, using Equations (4.6) and (4.7) the optimization problem in (4.5) is rewritten to

$$\begin{aligned} &\min \frac{1}{2} w^T w && (4.8)\\ &\text{s.t. } \min_{i=1,\ldots,N} \; y_i(w^T x_i + b) = 1. \end{aligned}$$

Note that the constraint in the optimization problem in (4.8) is almost linear. Removing the minimum function in the constraint makes the constraint linear. Therefore, the following equivalent optimization problem is given

$$\begin{aligned} &\min \frac{1}{2} w^T w && \text{(HM-P)}\\ &\text{s.t. } y_i(w^T x_i + b) \geq 1 && \forall i = 1,..,N.\\ &\quad w \in \mathbb{R}^D, \; b \in \mathbb{R}, \end{aligned}$$

To show that (HM-P) and (4.8) are equivalent, note that by Equations (4.3) and (4.6) it holds that $y_n(w^T x_n + b) = 1$. For all other x_i where $i \neq n$, the distance to the separating hyperplane is greater than or equal to the distance of x_n to the hyperplane, so $y_i(w^T x_i + b) \geq 1$ for all $i = 1, ..., N$. Suppose (HM-P) gives a solution where $y_i(w^T x_i + b) > 1$ for all $i = 1, ..N$, then the scaling constraint of Equation (4.3) does not hold. Also, such a solution can not be the global minimum, because by scaling down w and b it is always possible to find a better solution such that $y_i(w^T x_i + b) = 1$ for at least one i. Hence, (4.8) is equivalent to (HM-P). Moreover, the latter is called the primal form of the **hard-margin SVM** optimization problem, where the data set has only two possible classifications and is strictly linear separable.

4.2.3 The soft-margin SVM optimization problem

The model in (HM-P) was developed for the ideal scenario where the two possible classifications in a data set are strictly linear separable. However, as in practice this is not often the case, the model in (HM-P) is not valid, because it no longer holds that the best separating hyperplane correctly separates all sample points. Therefore, the constraint $y_i(w^T x_i + b) \geq 1$ for all $i = 1, .., N$ no longer holds, because for example a misclassified sample point i would imply that $y_i(w^T x_i + b) < 0$. Figure 4.5 gives an example of a scenario with such misclassifications.

To solve this problem, Cortes and Vapnik (1995) introduced the **soft-margin support vector machine** which deals with the problem of non-separability by introducing an error-variable ξ_i. The soft-margin SVM allows sample points to violate the margin constraint $y_i(w^T x_i + b) \geq 1$ by softening this constraint to $y_i(w^T x_i + b) \geq 1 - \xi_i$, where $\xi_i \in \mathbb{R}$ and $\xi_i \geq 0$. However, data points that violate this constraint and thus the margin are penalized. This is done by adding ξ_i to the objective function, so that sample points are penalized by the proportion of which the point violates the margin. Note that if a data point is correctly classified and lies outside the margin, ξ_i is equal to zero and there is no penalty. The following optimization problem is obtained using the new constraint.

$$\begin{aligned}
\min \ & \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i && \text{(SM-P)} \\
\text{s.t. } & y_i(w^T x_i + b) \geq 1 - \xi_i && \forall i = 1, .., N \\
& \xi_i \geq 0 && \forall i = 1, .., N \\
& w \in \mathbb{R}^D, \ b \in \mathbb{R}, \ \xi \in \mathbb{R}^N, &&
\end{aligned}$$

where C is a parameter that determines how much violations are penalized. As now also the ξ_i's are minimized, it follows that the resulting hyperplane maximizes the margin while simultaneously minimizing the number of sample points that violate this margin. It follows that there is a trade-off between the size of the margin and the number and size of the violations. Moreover, the value of C has a significant impact on this trade-off. The result is that for different values of C, different 'best' separating hyperplanes are obtained, making it difficult to conclude which value of C gives the overall best separating hyperplane. Therefore, it is necessary to have a method to test the performance of a SVM for different values of C. Chapter 6 discusses how the performance of a SVM can be measured.

4.3 Training the SVM using a non-linear classifier

Using a linear classifier is a very restricting method of separating the data set and in practice, this method not always provides good results (Cristianini and Shawe-Taylor, 2000). For example, Figure 4.6 gives an example of a data set that can not be separated accurately using a linear classifier.

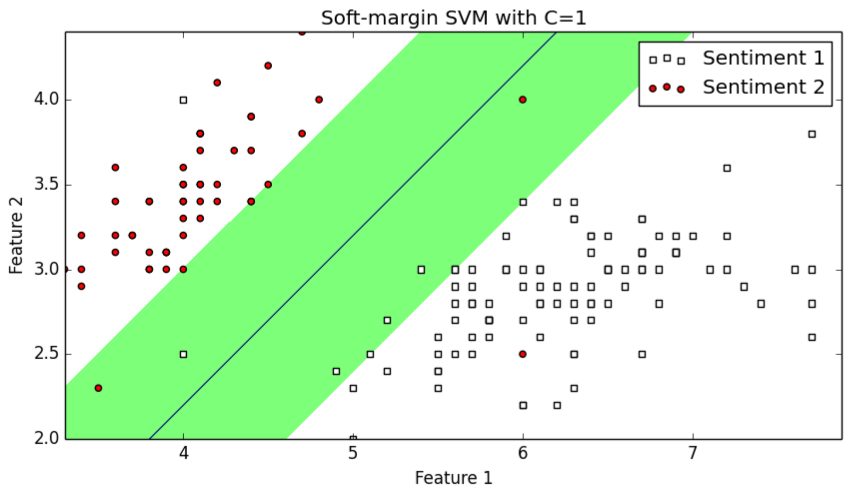


Figure 4.5: Example of sample points that are not strictly linear separable. Any separating hyperplane therefore has sample points that lie on the wrong side of the separating hyperplane or violate the (accentuated) margin M.

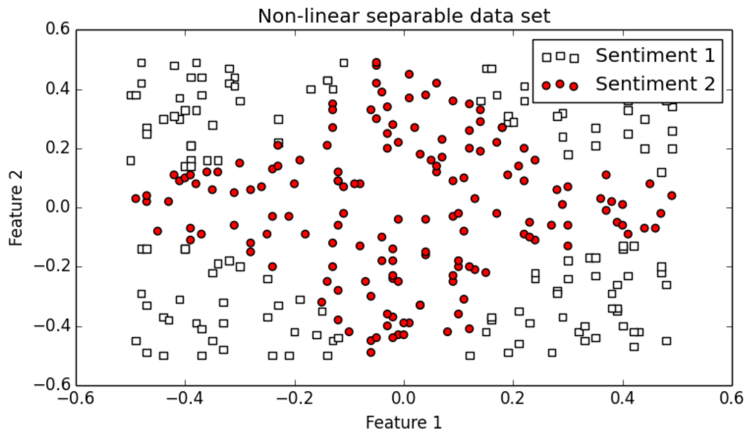


Figure 4.6: Example of a data set that can not be separated accurately using a linear classifier.

Using the soft-margin SVM for the example data set in Figure 4.6 still gives a separating hyperplane. However, due to the linearity of the resulting separating hyperplane, the data is not separated very accurate, i.e. many sample points are misclassified. Figure 4.7 plots the soft-margin SVM decision boundary of the example data set of Figure 4.6. From Figure 4.7 it follows that many of the points are misclassified when a linear classifier is used.

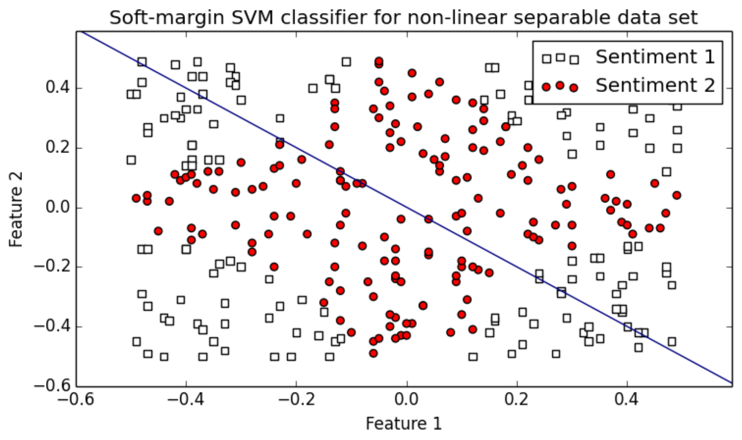


Figure 4.7: Example showing that using the soft-margin SVM with a linear classifier on a not linear separable data set does not always yield accurate results. All points to the left of the decision boundary are classified as 'class 2', whereas all points to the right of the decision boundary are classified as 'class 1'. Note that many sample points lie on the wrong side of the separating hyperplane and are thus misclassified.

4.3.1 Mapping the data to a higher dimension

A method to make non-linear data separable using a linear SVM classifier is by mapping the data to a higher dimension (Aizerman et al., 1964). Figure 4.8 provides an example of a simple mapping function, i.e. $\phi(\begin{bmatrix} x_1 & x_2 \end{bmatrix}) = \begin{bmatrix} x_1 & x_2 & |x_1 x_2| \end{bmatrix})$, that maps a not linear separable 2D dataset to a 3D dimension. In the 3D dimension the data set can then be separated using a linear classifier. In Figure 4.9, the separating hyperplane of the 3D dimension is reduced to a non-linear classifier for the original 2D dimension.

Using a mapping function ϕ, non-linear classifiers can be created for data sets that can not be separated linearly. However, the mapping results in a higher dimensional data set. Therefore, the mapping increases the number of variables of (SM-P), i.e. more w_i's are necessary. In the example in Figure 4.9, the dimension of the matrix X went from $\mathbb{R}^{300\times 2} \to \mathbb{R}^{300\times 3}$. However, suppose $X \in \mathbb{R}^{N\times D}$ and a polynomial mapping of degree 2 is needed in order to obtain linear separability in the higher dimension. The polynomial mapping of degree 2 is given by

$$\phi(x) = \left[x_1^2, \; \ldots \; x_D^2, \; x_1x_2, \; x_1x_3, \; \ldots, \; x_{D-1}x_{D-2}, \; x_Dx_1, \; \ldots, \; \; x_Dx_{D-2}, \; x_Dx_{D-1}\right]. \tag{4.9}$$

The mapping in Equation 4.9 implies calculating the square of all features and the cross product of all features. This mapping would imply that $\phi(x) : \mathbb{R}^D \mapsto \mathbb{R}^{\frac{1}{2}D(D+1)}$. If, for example, $N = 300$ and $D = 100$, this polynomial mapping would imply that the matrix $X : \mathbb{R}^{300\times 100} \mapsto \mathbb{R}^{300\times 5.050}$ and as a result $w : \mathbb{R}^{100} \mapsto \mathbb{R}^{5.050}$. Text analysis

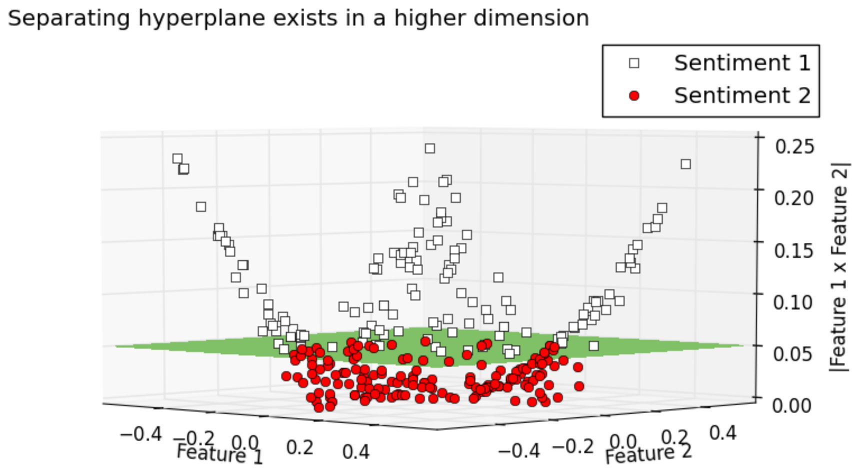


Figure 4.8: Example of how mapping data points to a higher dimension enables them to be linearly separated.

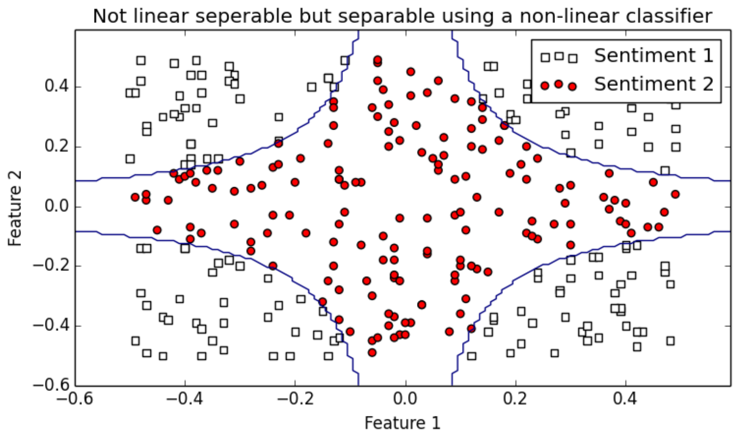


Figure 4.9: Non-linear decision boundary for the example data set in Figure 4.6

often implies using a large number of features, because there are lots of possible words that contain sentiment. Therefore, the value of D is often very large and using the polynomial mapping thus results in a very large matrix X that can be very time and memory expensive to compute. Additionally, the mapping results in a larger number of variables and thus increases the problem size significantly. In general, the larger the problem size, the more time and memory expensive it is to find the optimal solution of the problem. Especially for quadratic optimization problems such as the support vector machine model. Therefore, using this polynomial mapping to achieve linear separability is in practice not useful for sentiment analysis.

A method to create a mapping ϕ without calculating the entire mapping to the higher dimensional space is known as the *kernel trick* (Boser et al., 1992). The trick is that calculating the inner products between the N samples generates the same meaningful information as mapping all examples to a higher dimensional space. Therefore, the kernel trick makes the mapping independent of the size of a sample, i.e. the value D. Using the kernel trick, it is possible to map $X : \mathbb{R}^{N\times D} \mapsto \mathbb{R}^{N\times N}$. As a result, the dimension of the variable w only increases slightly, i.e. $w : \mathbb{R}^{D} \mapsto \mathbb{R}^{N}$. In the previous polynomial mapping example, this would imply $X : \mathbb{R}^{300\times 100} \mapsto \mathbb{R}^{300\times 300}$ and $w : \mathbb{R}^{100} \mapsto \mathbb{R}^{300}$. This is a big difference compared to the polynomial mapping function ϕ in Equation (4.9) that mapped X to $\mathbb{R}^{300\times 5.050}$ and increased w to $\mathbb{R}^{5.050}$. The kernel trick is discussed in Chapter 4.3.3. However, to benefit from the advantage of the kernel trick, (SM-P) is first reduced to the Lagrangian dual formulation.

4.3.2 The dual representation of the SVM

Cortes and Vapnik (1995) showed that (SM-P) can be rewritten using the Lagrangian dual optimization technique (Boyd and Vandenberghe, 2009). First note that the generalized Lagrangian of (SM-P) is defined by

$$\mathcal{L}(w,b,\xi,\alpha,\beta) \ = \frac{1}{2}w^T w + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[y_i(w^T x_i + b) - 1 + \xi_i\right] - \sum_{i=1}^{N}\beta_i\xi_i, \tag{4.10}$$

where $\alpha, \beta \in \mathbb{R}^N$ are Lagrange multipliers. The Lagrangian duality theory states that

$$d^* = \max_{\alpha\geq 0,\beta\geq 0}\min_{w,b,\xi}\mathcal{L}(w,b,\xi,\alpha,\beta) \leq \min_{w,b,\xi}\max_{\alpha\geq 0,\beta\geq 0}\mathcal{L}(w,b,\xi,\alpha,\beta) = p^*, \tag{4.11}$$

where p^*=(SM-P). Moreover, when strong duality holds, it holds that $d^* = p^*$. Therefore, in order to use the dual formulation d^* to solve (SM-P), it is necessary to have strong duality. Note that the following two conditions are sufficient for strong duality to hold (Boyd and Vandenberghe, 2009):

1. the primal problem needs to be convex w.r.t. the variables w, b, ξ.
2. Slater's condition needs to hold.

The constraints of (SM-P) are affine and thus convex w.r.t. w, b and ξ. The objective function of (SM-P) is convex because it is an Euclidean squared norm plus an affine function. Therefore, (SM-P) is a convex problem and condition 1 for strong duality is satisfied. Slater's condition holds if there exists an x $\in$ $\mathbf{relint}(\mathcal{D})$ such that

$$f_i(x) < 0, i = 1,...,N \ , \qquad\qquad Ax = b,$$

where $\mathcal{D} = \bigcap_{i=0}^{m} dom(f_m)$ (Boyd and Vandenberghe, 2009). Therefore, Slater's condition holds if a feasible point is found such that all inequality constraints strictly hold. Take any w and b. Set $\xi_i = \max\left\{2 - y_i(w^T x_i + b), 1\right\}$. Then

$$-y_i(w^T x_i + b) + 1 - \xi_i \leq -1 < 0 \qquad \text{and} \qquad -\xi_i \leq 1 < 0 \text{ for all } i.$$

Both inequalities hold strictly, so Slater's condition holds. Therefore, the two sufficient conditions for strong duality hold for (SM-P). As the problem is differentiable and there is strong duality, it holds that the KKT conditions are necessary and sufficient for global optimality (Boyd and Vandenberghe, 2009). This is useful when solving the SVM dual optimization problem, because any solution that satisfies all KKT conditions is guaranteed to be the global optimum. Chapter 5 uses this property to find the global solution of the optimization problem.

From the first order conditions it follows that at the minimum of $\mathcal{L}$ w.r.t. w, b and ξ it holds that

$$\nabla_w \mathcal{L} = w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0 \Leftrightarrow w = \sum_{i=1}^{N} \alpha_i y_i x_i, \tag{4.12}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^{N} \alpha_i y_i = 0, \tag{4.13}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \Leftrightarrow \alpha_i + \beta_i = C. \tag{4.14}$$

First note that the objective function in Equation (4.10) can be rewritten to

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i \left[y_i(w^T x_i + b) - 1 + \xi_i\right] - \sum_{i=1}^{N} \beta_i \xi_i \\
&= \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i \left[y_i(w^T x_i + b) - 1\right] - \sum_{i=1}^{N} \alpha_i \xi_i - \sum_{i=1}^{N} \beta_i \xi_i \\
&= \frac{1}{2} w^T w + (C - \alpha_i - \beta_i) \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i \left[y_i(w^T x_i + b) - 1\right].
\end{aligned} \tag{4.15}$$

Substituting (4.14) in (4.15) gives

$$\mathcal{L} = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i \left[y_i(w^T x_i + b) - 1\right]. \tag{4.16}$$

Additionally, substituting (4.13) in (4.16) gives

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}w^Tw - \sum_{i=1}^{N}\alpha_i\left[y_i(w^Tx_i+b)-1\right] \\ &= \frac{1}{2}w^Tw + \sum_{i=1}^{N}\alpha_i - b\sum_{i=1}^{N}\alpha_i y_i - \sum_{i=1}^{N}\alpha_i y_i w^T x_i \\ &= \frac{1}{2}w^Tw + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\alpha_i y_i w^T x_i. \qquad (4.17)\end{aligned}$$

Finally, substituting (4.12) in (4.17) gives

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}w^Tw + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\alpha_i y_i w^T x_i \\ &= \frac{1}{2}\left(\sum_{i=1}^{N}\alpha_i y_i x_i^T\right)\left(\sum_{i=1}^{N}\alpha_i y_i x_i\right) + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\alpha_i y_i\left(\sum_{i=1}^{N}\alpha_i y_i x_i^T\right)x_i \\ &= \sum_{i=1}^{N}\alpha_i + \frac{1}{2}\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j x_i^T x_j\right) - \left(\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j x_i^T x_j\right) \\ &= \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j x_i^T x_j\right). \qquad (4.18)\end{aligned}$$

The result is a Lagrangian function that only depends on α. Moreover, from Equation (4.14) and $\beta_i \geq 0$, it follows that the value of C is an upper bound on the value of α. Therefore, the Lagrangian dual optimization problem d^* is given by

$$\begin{aligned}d^* &= \max_{\alpha\geq 0}\ \mathcal{L} \\ &\quad \text{s.t. } \alpha_i \leq C \quad \forall i = 1,..,N \\ &\quad \sum_{i=1}^{N}\alpha_i y_i = 0 \quad \forall i = 1,..,N \\ &= \max_{\alpha}\ \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j x_i^T x_j\right) \\ &\quad \text{s.t. } 0 \leq \alpha_i \leq C \quad \forall i = 1,..,N \\ &\quad \sum_{i=1}^{N}\alpha_i y_i = 0 \quad \forall i = 1,..,N\end{aligned}$$

$\Leftrightarrow$

$$\begin{aligned} d^* = \min_{\alpha} \ & \frac{1}{2}\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j x_i^T x_j\right) - \sum_{i=1}^{N}\alpha_i \\ \text{s.t. } & 0 \leq \alpha_i \leq C \qquad \forall i = 1,..,N \\ & \sum_{i=1}^{N}\alpha_i y_i = 0 \qquad \forall i = 1,..,N \end{aligned}$$

$\Leftrightarrow$

$$\begin{aligned} d^* = \min_{\alpha} \ & \frac{1}{2}\alpha^T \begin{bmatrix} y_1y_1x_1^Tx_1 & y_1y_2x_1^Tx_2 & \dots & y_1y_Nx_1^Tx_N \\ y_2y_1x_2^Tx_1 & y_2y_2x_2^Tx_2 & \dots & y_2y_Nx_2^Tx_N \\ \vdots & \vdots & \vdots & \vdots \\ y_Ny_1x_N^Tx_1 & y_Ny_2x_N^Tx_2 & \dots & y_Ny_Nx_N^Tx_N \end{bmatrix} \alpha - (\mathbf{1})^T\alpha \\ \text{s.t. } & 0 \leq \alpha \leq C \\ & y^T\alpha = 0. \end{aligned}$$

Define $Q = \begin{bmatrix} y_1y_1x_1^Tx_1 & y_1y_2x_1^Tx_2 & \dots & y_1y_Nx_1^Tx_N \\ y_2y_1x_2^Tx_1 & y_2y_2x_2^Tx_2 & \dots & y_2y_Nx_2^Tx_N \\ \vdots & \vdots & \vdots & \vdots \\ y_Ny_1x_N^Tx_1 & y_Ny_2x_N^Tx_2 & \dots & y_Ny_Nx_N^Tx_N \end{bmatrix}$ and e as a vector of ones of length N. Note that Q consists solely of the parameters x and y. Therefore, Q is considered as input in the model. Hence, the Lagrangian dual problem of the soft-margin SVM is given by

$$\begin{aligned} d^* = \min_{\alpha} \ & \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ \text{s.t. } & 0 \leq \alpha \leq C \\ & y^T\alpha = 0. \end{aligned} \tag{SM-D}$$

Solving (SM-D) gives the optimal vector α^*. With the help of α^*, the best separating hyperplane is constructed. First, using Equation (4.12), w^* is calculated by

$$w^* = w_{X,y}(\alpha^*) = \sum_{i=1}^{N}\alpha_i^* y_i x_i. \tag{4.19}$$

The value of b^* is found using the complementary slackness conditions of the KKT-conditions. The complementary slackness conditions are given by

$$\alpha_i\left(1 - \xi_i - y_i(w^Tx_i + b)\right) = 0 \qquad \forall i = 1,...,N, \tag{4.20}$$

$$\beta_i\xi_i = 0 \qquad \forall i = 1,...,N. \tag{4.21}$$

Take any point i for which $0 < \alpha_i < C$. Then from Equation (4.20) it follows that

$$1 - \xi_i - y_i(w^T x_i + b) = 0 \Leftrightarrow 1 - \xi_i = y_i(w^T x_i + b). \tag{4.22}$$

From Equation (4.14) it follows that $\beta_i > 0$ and thus from Equation (4.21) it follows that $\xi_i = 0$. Hence, from (4.22) it follows that $y_i(w^T x_i + b) = 1$ and $b = 1 - y_i w^T x_i$. Therefore,

$$b^* = b_{X,y}(\alpha^*) = 1 - y_i \left(\sum_{i=1}^{N} \alpha_i^* y_i x_i \right)^T x_i \tag{4.23}$$

The soft-margin SVM primal problem (SM-P) and the soft-margin SVM dual problem (SM-D) are both linear constrained quadratic optimization problems. However, the two problems differ significantly in problem size. Consider a data set of N samples and D features. The primal formulation consists of $N + D + 1$ variables and $2N$ constraints. The dual formulation consists of N variables and $N + 1$ constraints. It follows that the problem size of the dual problem is smaller, especially when the number of features, i.e. D, is high. Additionally, it turns out that the dual formulation is advantageous when mapping non-linear separable data to a higher dimension in order to achieve separability. This advantage is discussed in Chapter 4.3.3.

4.3.3 Applying the kernel trick

Chapter 4.3.1 explained that non-linear classifiers for not linearly separable data sets can be created by mapping data to a higher dimension. In (SM-D), the sample points x only appear in the matrix Q. Therefore, the mappings $x \to \phi(x)$ are implemented by changing Q to

$$Q = \begin{bmatrix} y_1 y_1 \phi(x_1)^T \phi(x_1) & y_1 y_2 \phi(x_1)^T \phi(x_2) & \dots & y_1 y_N \phi(x_1)^T \phi(x_N) \\ y_2 y_1 \phi(x_2)^T \phi(x_1) & y_2 y_2 \phi(x_2)^T \phi(x_2) & \dots & y_2 y_N \phi(x_2)^T \phi(x_N) \\ \vdots & \vdots & \vdots & \vdots \\ y_N y_1 \phi(x_N)^T \phi(x_1) & y_N y_2 \phi(x_N)^T \phi(x_2) & \dots & y_N y_N \phi(x_N)^T \phi(x_N) \end{bmatrix}.$$

Chapter 4.3.1 showed that the mapping $\phi(x)$ can significantly increase the dimension of the vector x. Therefore, calculating the inner products of $\phi(x_i)$ and $\phi(x_j)$ for all i and j in the matrix Q also significantly becomes more demanding in terms of calculations and memory as the number of samples N or the dimensionality of the mapping increases.

The kernel trick avoids the need to fully calculate all mappings and inner products (Boser et al., 1992). The trick is that kernel functions K exist such that $K(x, z) = \phi(x)^T \phi(z)$, making it possible to rewrite Q to

$$Q = \begin{bmatrix} y_1 y_1 K(x_1, x_2) & y_1 y_2 K(x_1, x_2) & \dots & y_1 y_N K(x_1, x_N) \\ y_2 y_1 K(x_2, x_2) & y_2 y_2 K(x_2, x_2) & \dots & y_2 y_N K(x_2, x_N) \\ \vdots & \vdots & \vdots & \vdots \\ y_N y_1 K(x_N, x_2) & y_N y_2 K(x_N, x_2) & \dots & y_N y_N K(x_N, x_N) \end{bmatrix}.$$

Note that the function $K(x,z)$ can be created by taking the inner product of the higher-dimensional mappings $\phi(x)$ and $\phi(z)$. However, proper kernel functions K enable $K(x,z)$ to be calculated much faster using the dimensional-space of the input vectors x. For example, consider the mapping function

$$\phi(x) = \begin{bmatrix} x_1x_1 & \dots & x_1x_D & x_2x_1 & \dots & x_{D-1}x_D & x_Dx_1 & \dots & x_Dx_D \end{bmatrix}^T.$$

Note that ϕ can be computed using $\mathcal{O}(D^2)$ calculations. It follows that $K(x,z) = \phi(x)^T\phi(z)$ requires $\mathcal{O}(D^2)$ calculations. However, the kernel function $K(x,z)$ is equivalent to

$$\begin{aligned} K(x,z) = \phi(x)^T\phi(z) &= \sum_{i,j=1}^{n}(x_ix_j)(z_iz_j) = \left(\sum_{i=1}^{n} x_iz_i\right)\left(\sum_{i=1}^{n} x_iz_i\right) \\ &= (x^Tz)^2. \end{aligned}$$

Note that $(x^Tz)^2$ can be computed using $\mathcal{O}(D)$ calculations. So, for this example, using $K(x,z) = (x^Tz)^2$ is significantly faster then calculating the inner product of the higher-dimensional mappings $\phi(x)$ and $\phi(z)$. Similarly, other mapping functions ϕ can be reduced to a much less computational expensive kernel function K. Well-known common kernel functions in machine learning are the

i) Linear kernel:
$K(x,z) = x^Tz$,

ii) Polynomial kernel of degree d with constant c:
$K(x,z) = (x^Tz + c)^d$,

iii) Radial Basis Function (RBF) or Gaussian kernel with sensitivity parameter σ:
$K(x,z) = \exp\left(-\frac{||x-z||^2}{2\sigma^2}\right) = \exp\left(-\gamma||x-z||^2\right)$, where $\gamma = \frac{1}{2\sigma^2}$.

Note that the linear kernel does not perform any mapping and thus gives a linear classifier. Also, the RBF kernel is a special type of kernel, because the underlying mapping ϕ maps the input x to an infinite-dimensional space (Cristianini and Shawe-Taylor, 2000). Moreover, note that the RBF kernel function K requires more complex mathematical operations, i.e. norm, exponent, division and square, than the other kernels. Therefore, the RBF kernel is often slower than the other kernels, because calculating the full matrix Q implies applying the more complex RBF kernel function N^2 times. However, the RBF kernel often provides the best results in practice and is therefore a widely-used kernel (Hsu et al., 2003).

4.3.4 Support vectors

The *support vectors* of a SVM optimization problem are the sample points that define the separating hyperplane (Cortes and Vapnik, 1995). By this support vector definition, all non-support vector sample points can be removed from a data set without affecting the best separating hyperplane obtained by the SVM.

Recall that the best separating hyperplane is the separating hyperplane that maximizes the margin. Chapter 4.2.1 showed that for a linear separable data set, the margin around the separating hyperplane is defined by the sample points of both classifications that lie nearest to that separating hyperplane. As a result, the support vectors of a linear separable data set are the points that lie on the boundary of the margin around the separating hyperplane. Figure 4.10 shows the support vectors of the strictly linear separable data set from the example in Figure 4.3. Figure 4.10 also shows that removing all non-support vector sample points will not affect the best separating hyperplane.

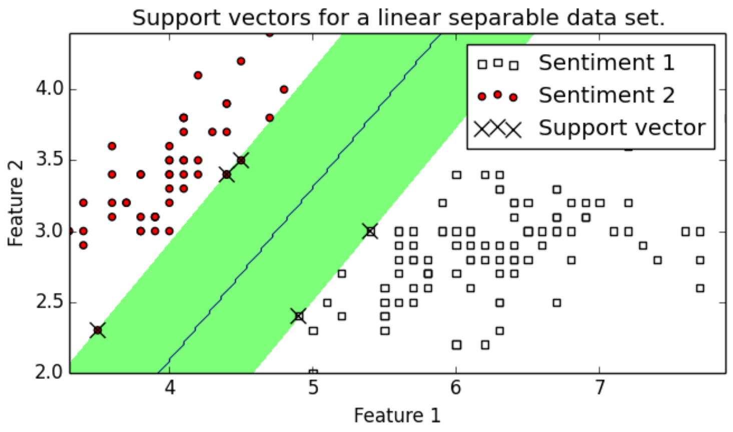


Figure 4.10: Support vectors for the linear separable data set of the example in Figure 4.3.

For a not linear separable data set the support vectors are not only the sample points that lie on the margin. Chapter 4.2.3 showed that the best separating hyperplane for a not linear separable data set can be obtained by using the soft-margin SVM optimization problem (SM-P). The soft-margin SVM allows sample points to violate the margin by introducing the penalty variable ξ_i. However, all sample points that violate this margin also affect the best separating hyperplane. Therefore, sample points violating the margin are also support vectors. Figure 4.11 shows the support vectors of the not linear separable data set from the example in Figure 4.5.

After mapping a not linear separable data set to a higher dimension to achieve linear separability, the best separating hyperplane is again found using (SM-P). Therefore, the support vectors are all sample points that violate or lie on the boundary of the margin. Figure 4.12 shows the higher dimensional support vectors of the not linear separable example in Figure 4.6. To obtain the higher dimensional support vectors of the last example required a full mapping of the original data set to a higher dimension in which the data set is linearly separable. Chapter 4.2.1 however discussed that this mapping is very time and memory-expensive to compute and

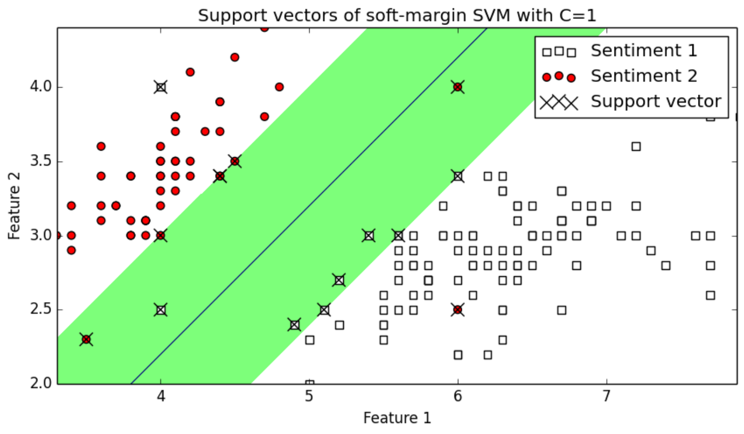


Figure 4.11: Support vectors of the not linear separable data set of the example in Figure 4.5

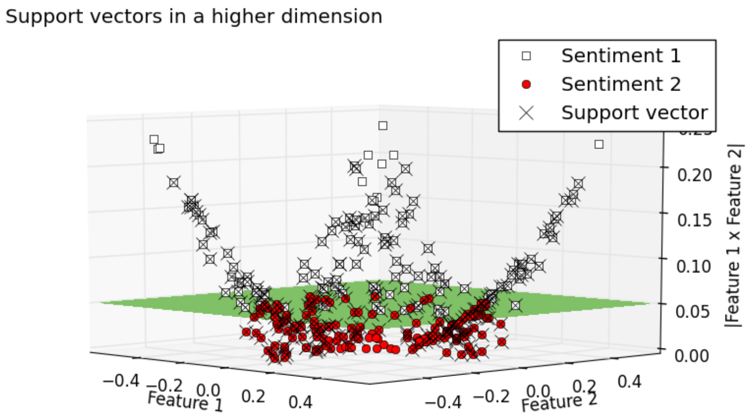


Figure 4.12: Support vectors of the not linear separable example in Figure 4.5, however, mapped to an higher dimension where it is linear separable.

significantly increases the SVM problem size. The kernel trick avoided the need to calculate the full mapping. Figure 4.13 shows the support vectors, obtained using the kernel trick, of a non-linear classifier for the not linear separable example in Figure 4.6.

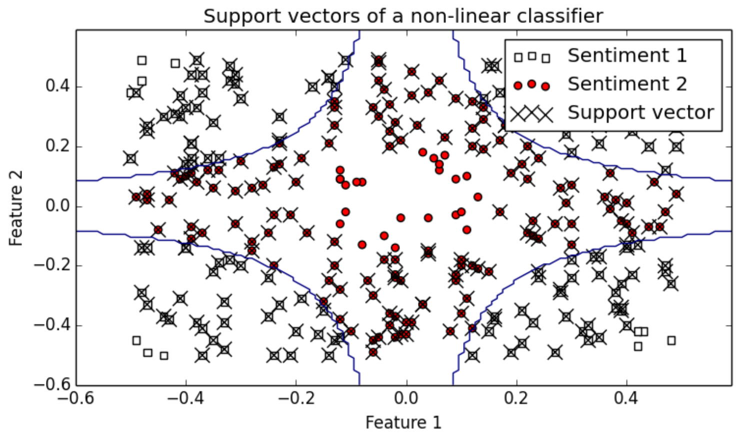


Figure 4.13: Support vectors of a non-linear classifier for the not linear separable example in Figure 4.5.

Note that (SM-D) provides an additional easier method to find the support vectors. From Equations (4.19) and (4.23) it follows that the best separating hyperplane is defined by the sum over all weighted sample vectors, where the weights are the values for α and y. Therefore, all samples for which $\alpha_i^* = 0$ do not contribute to the best separating hyperplane and can thus be removed. Hence, all sample points for which $\alpha_i^* > 0$ are support vectors of the problem. It follows that, to store the best separating hyperplane found by (SM-D), only the support vectors and their corresponding α_i^*'s need to be stored. Also, Chapter 5.2 shows that using the value of α to determine whether a sample point is a support vector makes it easier to find the global optimal solution of (SM-D).

4.4 Remarks on SVM training

Some important remarks on training a SVM first need to be stated, before proceeding on how (SM-D) can be solved efficiently.

4.4.1 Feature scaling

First of all, before training a SVM to find the best separating hyperplane, it is important to apply feature scaling (Hsu et al., 2003). Feature scaling prevents that feature value outliers may dominate other very low-valued features. Also, feature scaling prevents numerical issues during the calculation of large numbers. Preventing numerical issues is especially useful when calculating kernel values. For example, kernel functions often calculate the inner product of two vectors. When these vectors get larger or contain higher numbers, numerical problems may occur when calculating inner products. In general, scaling and normalization techniques are applied such that all feature values are standard normally distributed or are in the range of $[0, 1]$. If feature values are not properly scaled, one of the following two possible scaling techniques is often applied:

1. **Min-Max scaling:** scale all feature values to the range of $[0, 1]$ by applying

$$X = \frac{X - min(X)}{max(X) - min(X)}.$$

2. **Standardization:** scale all feature values to be standard normally distributed by applying

$$X = \frac{X - \bar{X}}{S_X}.$$

Chapter 3.2 stated that, in text analysis, the matrix X is often very sparse. However, the two scaling techniques above result in dense matrices and thus the sparsity of the matrix X is lost. The dense matrices result in higher memory-requirements and longer computational times, e.g. for calculating inner products. Chapter 3.2 also showed that some feature extraction weighting methods result in feature values that are already properly scaled. That is, the chapter showed that FP as weighting method resulted in all feature values being binary, i.e. 0 or 1. Using FP as feature extraction weighting method is advantageous, because it avoids the need of feature scaling and thus preserves the sparsity of X. Therefore, in the remainder of this thesis, FP is used as the feature extraction weighting method.

4.4.2 Choosing penalty and kernel parameters

The second remark is about choosing the best values of the penalty parameter C and the kernel parameters such as the type of kernel and corresponding kernel parameters $(\gamma, c, d, ...)$. Recall that the value C in the soft-margin SVM optimization problem penalizes sample points proportional to the violation of the margin. Note that a high value of C severely penalizes violations of this margin. Therefore, a high value of C implies that the best separating hyperplane is the hyperplane that correctly classifies all sample points used to train the SVM. However, this introduces the problem of overfitting as shown in Figure 6.1 and discussed in Chapter 6.1. A low value of C hardly punishes data points that violate the margin. Therefore, a low value of C results in bad separating hyperplanes with lots of samples points that violate the margin or are misclassified. Hence, the value of C has a large impact on the performance of the SVM and therefore has to be chosen carefully.

Similarly, the choice of kernel also has a big impact on the performance of a SVM. From the previous sections it follows that using a linear kernel on a not linear separable data set results in many sample points being misclassified. However, using a RBF kernel on a data set that is linear separable is a waste of computation time, because the RBF kernel involves more complex calculations. Ideally, the type of kernel is chosen based on prior knowledge of the data set. If there is not such knowledge, the RBF kernel is commonly used because of it is good capabilities to catch non-linear relationships (Cristianini and Shawe-Taylor, 2000).

After a kernel is chosen, some kernels require additional parameters to be set. For example, the polynomial kernel requires setting the values of c and d, whereas the RBF kernel requires setting the value of γ. Similarly as choosing the value of C, the kernel parameters have a very big impact on the result. Failing to choose proper kernel parameter values results in kernels failing to map the original not linear separable data to a higher dimension in where it is linear separable. The result is an inaccurate separating hyperplane with many sample points that are misclassified. Moreover, an inaccurate separating hyperplane also implies that classification predictions of new sample points are inaccurate.

One of the best methods to determine the best values of the penalty and kernel parameters is to perform a *grid search* (Hsu et al., 2003). A grid search consists of iterating over a range of possible values for all parameters. For each combination of parameter values, a SVM is trained and its performance is measured. The combination of parameter values that gives the highest SVM performance is considered the best. For example, in the case it is known that a RBF kernel is necessary, performing a grid search implies training and testing a SVM for a large number of different (C, γ) combinations.

Although smarter methods to determine the optimal parameter values exist, the simple grid search method is still very effective, i.e. easy to implement, almost equal in performance and not much slower (Hsu et al., 2003). Moreover, additional searching techniques can be applied on the grid search method to improve the grid search performance. For example, the grid search can first be restricted to iterating over a rough range of parameter values. After good rough parameter values are found, the grid search can then be intensified around promising rough parameter values.

5. Solving the SVM training problem

Chapter 4 discussed several methods to model the SVM learning problem as an optimization problem. The optimization problem (SM-D) was considered to be the best formulation. However, it was not mentioned how (SM-D) can be solved efficiently, i.e. how to obtain the vector α^* that gives the best separating hyperplane. This chapter discusses possible approaches to efficiently solve the SVM problem. Also, the pros and cons of each approach are evaluated.

5.1 Quadratic Programming

The optimization problem (SM-D) consists of a quadratic objective function subject to linear constraints. Therefore, the most intuitive approach to solve these problems is using Quadratic Programming (QP). Quadratric Programming is a thoroughly studied method of optimization and multiple algorithms to solve QP problems are available, e.g. interior point or conjugate gradient algorithms. Moreover, well-known solvers such as C-Plex, GUROBI and MOSEK have built-in algorithms to solve QP problems. Therefore, the pro of using a QP approach is that the SVM optimization problem is easy to implement.

However, in general, QP is a very slow and computational- and memory-expensive method that does not scale well with the number of samples in the data set. Most QP algorithms require calculating the Hessian of the quadratic objective function and this Hessian calculation often leads to problems. For example, consider the convex quadratic formula $f(x) = x^T Q x$ where $x \in \mathbb{R}^N$ and Q is a $N \times N$ matrix consisting of random values. Calculating the Hessian of f implies first calculating N partial derivatives, followed by calculating N^2 second derivatives. Suppose $N = 1.000$, then calculating the Hessian of f requires 1.000.000 Hessian values to be calculated. Therefore, evaluating the Hessian of a quadratic formula quickly becomes more computational expensive as the dimension of the input variable increases. Moreover, because all values of the Hessian need to be stored in the memory too, the problem also quickly becomes more memory-expensive. Therefore, the QP approach for solving SVMs is in practice only applicable on relatively small data sets. Otherwise, time or memory issues can be expected.

5.2 Sequential Minimal Optimization

Platt et al. (1998) developed the Sequential Minimal Optimization (SMO) algorithm to overcome the computational difficulties of solving a very large QP problem. SMO breaks the large QP problem of (SM-D) into very small QP subproblems, i.e. QP problems consisting of only two variables. Using the smaller QP subproblems, the global solution for the large QP problem is found iteratively. The small QP problems consisting of only two variables can be solved fast and analytically. Moreover, the small QP problems are very memory-efficient making the SMO method much better scalable with the number of samples than the QP approach of Chapter 5.1.

To find the global solution of the large QP problem given by (SM-D), the SMO algorithm iteratively optimizes the variables of (SM-D), i.e. all α_i's, until all KKT-conditions of (SM-D) are satisfied. For a positive definite QP minimization problem, the KKT-conditions ensure a solution is optimal (Chinneck, 2006). To make (SM-D) a positive definite QP problem, the matrix Q in (SM-D) must be positive definite, i.e. the kernel function K must satisfy Mercers conditions (Cortes and Vapnik, 1995). Note that popular kernels such as the linear, polynomial and RBF kernel all satisfy Mercer conditions. Moreover, the KKT-conditions for (SM-D) can be reduced to a particular simple version given by

$$\begin{aligned} \alpha_i = 0 &\Rightarrow y_i f(x_i) \geq 1, \\ 0 < \alpha_i < C &\Rightarrow y_i f(x_i) = 1, \\ \alpha_i = C &\Rightarrow y_i f(x_i) \leq 1, \end{aligned}$$

where $f(x_i) = w^T x_i + b$.

At every iteration of the algorithm, SMO chooses two Lagrange multipliers α_i (called the working set), where one of the multipliers currently violates the KKT-conditions. The two Lagrange multipliers are jointly optimized such that they both satisfy the KKT-conditions and the improvement in objective function is maximized. Then the objective function and the number of KKT-violations are updated and a new working set is selected. Therefore, at every iteration, SMO reduces the number of Lagrange multipliers that violate the KKT-conditions and improves the objective function. The SMO algorithm stops when there are no more Lagrange multipliers that violate the KKT-conditions. At this point, the found α_i's satisfy all KKT-conditions and the globally optimal solution α^* is obtained.

Platt et al. (1998) showed how the small QP problems consisting of two Lagrange multipliers can be solved analytically. An important step in the algorithm is choosing the two Lagrange multipliers to optimize in an iteration, i.e. the working set to select. Note that the working set selection determines the speed of the convergence to the optimal solution and thus also the solving time of the algorithm. Platt et al. (1998) therefore also proposed a heuristic to determine the best working set. This heuristic first iterates over all sample points in the data set until a sample point is found that violates the KKT conditions. This sample point is used as the first Lagrange

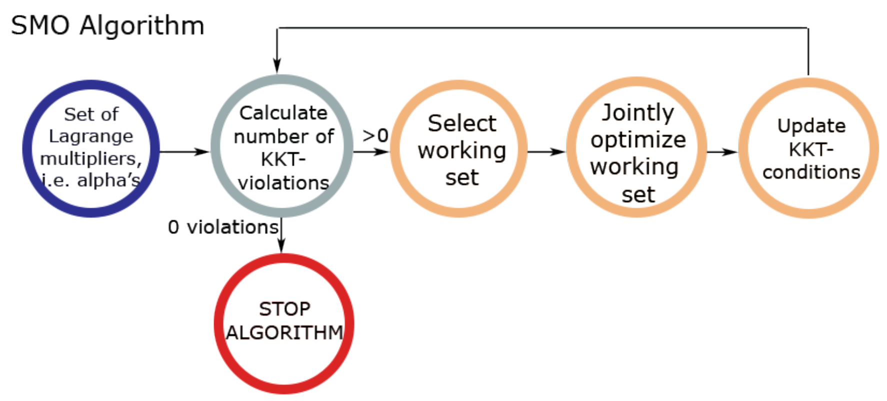

SMO Algorithm
Set of Lagrange multipliers, i.e. alpha's
Calculate number of KKT-violations
>0
Select working set
Jointly optimize working set
Update KKT-conditions
0 violations
STOP ALGORITHM

Figure 5.1: An illustration of the processes in the SMO algorithm.

multiplier in the working set. The second multiplier is then chosen such that the improvement in objective function for that iteration is likely to be maximized.

The SMO method is considered as a big breakthrough in SVM theory, because it suddenly became possible to efficiently solve large-scale SVMs without any optimization software. Many papers therefore also give Platt et al. (1998) a large part of credit for the increased popularity of SVMs. Although the working set selection heuristic proposed by Platt et al. (1998) provided good results, several improvements or alternative working set selection methods have been proposed (Fan et al. 2005, Glasmachers and Igel 2006). Nowadays, many commercial software packages such as MatLab and Scikit-Learn solve the SVM optimization problem using the SMO algorithm of Platt et al. (1998) in combination with the working set selection method of Fan et al. (2005).

5.3 Second Order Cone Programming

Debnath et al. (2005) proposed a Second Order Cone Programming (SOCP) method to solve the SVM problem. The SOCP method formulates the large QP problem of the dual soft-margin SVM into a new, also large, SOCP problem consisting of only linear and second-order cone constraints. Using a chunking decomposition method on the kernel matrix, the problem is split into multiple small SOCP problems that can be solved more efficiently. Similar as SMO, the solving time of the SOCP method depends on the decomposition method. The convergence of the decomposition method is determined by the working set selection method, i.e. the method that creates the SOCP subproblems. In contrary to the SMO method, the SOCP working set selection method is not restricted to using only two sample points. Debnath et al. (2005) slightly adapted the working set selection approach of Hsu and Lin (2002b) to deal with multiple sample points as the working set.

Although Debnath et al. (2005) claim to be faster than the SMO algorithm, there are some remarks that need to be made. For one, the SOCP method was compared against the SMO algorithm and the working set selection method developed by Platt et al. (1998). As mentioned in the previous section, several improvements have been proposed for this SMO working set selection method. Therefore, it is uncertain whether the SOCP method is still faster than the SMO algorithm in combination with the improved working set selection method by Fan et al. (2005).

Secondly, the SOCP method uses a Cholesky factorization on the kernel matrix to reformulate the large QP problem into a SOCP problem. Because the dual SVM problem is a convex QP and the matrix Q is positive semi-definite, it is possible to decompose the matrix $Q = GG^T$ using Cholesky factorization. However, some software packages such as MatLab still have problems when trying to calculate this decomposition. The problems are mainly caused by numerical precision errors. Additional steps are necessary to find the proper decomposed matrix G. Therefore, the SOCP method is more subject to numerical precision problems than the SMO method.

Lastly, the chunking decomposition method used in the SOCP method is based on the notion that the best separating hyperplane does not change if all non-support vectors are removed, i.e. removing all rows for which $\alpha_i = 0$. This reduces the large SOCP problem into smaller SOCP subproblems. All SOCP subproblems are solved to find all of the non-zero α_i's. The non-zero α_i's of multiple subproblems are then combined, split again in subproblems and solved to find the new non-zero α_i's of all subproblems. This process is repeated until all of the the non-zero α_i's of the large SOCP problem are found. The last step thus solves the large SOCP problem and provides the global optimal solution.

Note that this chunking method performs the best when most of the α_i's are equal to zero, because that implies a lot of samples can be removed. The chunking method thus performs the best when there are few support vectors. If the number of support vectors is almost equal to the number of samples in the data set, the problem that is solved at the last iteration of the algorithm is almost as large as the full and large SOCP problem. A high number of support vectors thus makes the chunking decomposition method almost useless. Simple preliminary calculations show that the sentiment analysis problem contains many support vectors, because of the high number of features. Each feature contains sentiment and is thus very likely to affect the best separating hyperplane. Because of the high number of support vectors, it is doubtful whether the SOCP method indeed gives better results than the SMO method when performing a sentiment analysis.

6. Measuring the performance of a SVM

A method to measure the performance of a SVM is required to compare the results of different SVMs or different SVM configurations such as different penalty and kernel parameter values. Moreover, a performance indication helps decide which SVM or SVM configuration provides the best sentiment predictions of new sample points, i.e. new on-line news articles. It is important to have an unbiased performance indication. Otherwise it might happen, for example, that a SVM achieves a very high performance for one particular data set, but a very low performance for different data sets. The latter example must be avoided to obtain reliable sentiment predictions of new samples. This chapter discusses two scoring functions as a method to measure the performance of a SVM. However, first, cross-validation is proposed as a method to obtain an unbiased performance indication.

6.1 Cross-validation to prevent overfitting

Assume that the performance of a SVM is determined by the SVMs accuracy with respect to classification predictions. Kohavi et al. (1995) defined the accuracy of a classifier as the probability of correctly classifying a randomly selected sample. This definition implies that the accuracy of a SVM is given by the probability that any sample from the data set is correctly classified. However, this induces a very large bias, because a SVM classifier is obtained by learning from the data set. Therefore, SVMs are often able to achieve an 100% accuracy on the data set by *overfitting* it. Overfitting occurs when a SVM interprets the noise in a data set as part of the relationship between the dependent and independent variables. The dashed line in the example in Figure 6.1 shows how a very accurate classifier for any data set can be obtained by overfitting it. The solid line in this figure provides a more intuitive classifier for this example, i.e. labeling all samples left of the line as '-1' and all samples right of the line as '+1'.

The problem of overfitting a data set is that it results in inaccurate predictions of new sample points, i.e. sample points outside the data set. For example, consider a new sample point in Figure 6.1 at coordinate $\begin{bmatrix}4.5 & 3\end{bmatrix}$ with classification -1 . As it lies very close to all other samples with classification -1, it intuitively makes sense that the new sample point also has classification -1. However, the overfitted classifier (dashed line) classifies this new sample point as classification $+1$. The more intuitive classifier (solid line) classifies the new sample point as the correct classification though. The example shows that an overfitted classifier results in a low classification accuracy on samples outside the data set. Therefore, overfitting makes it difficult to obtain an unbiased, low-variance estimate of the accuracy of a SVM.

Note that an overfitted classifier is the result of choosing wrong SVM penalty and kernel parameters. Especially the penalty parameter C has a big impact on overfitting. Chapter 4.2.3 showed that the penalty parameter C proportionally penalizes sample points that violate the margin or lie on the wrong side of the classifier. It follows that as the value of C increases, misclassified sample points are penalized more. Therefore, the best separating hyperplane is adjusted such that less sample points are misclassified and thus less penalties are incurred. The result is that the best separating hyperplane is scrambling to correctly classify all samples and weird counter-intuitive classifiers such as the dashed line in Figure 6.1 are obtained.

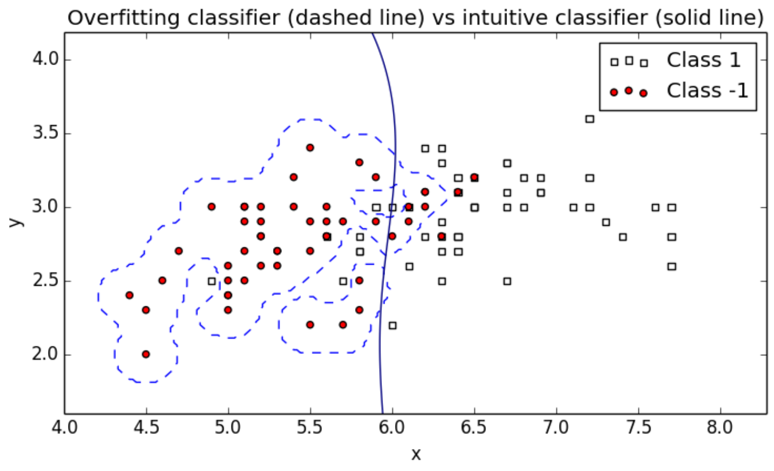


Figure 6.1: An example of a SVM that overfits the data (dashed line). Note that the solid line would provide a more intuitive decision boundary in this scenario.

Kohavi et al. (1995) proposed using *stratified ten-fold cross-validation* to obtain the best unbiased, low-variance estimation of a classifiers accuracy, even in case of overfitting. Define $T = \{1, 2, ..., k\}$. In regular k-fold cross-validation, the data set $\mathcal{D}$ is randomly divided into k subsets of equal size, denoted by $\mathcal{D}_1, ..., \mathcal{D}_k$. For each $t \in T$, a SVM is trained using the training data set given by $\mathcal{D} \setminus \mathcal{D}_t$ and then tested on the test data set given by $\mathcal{D}_t$. Testing is done by calculating the predicted classification for each sample in the test data set. The percentage of correct classifications of the test set $\mathcal{D}_t$ is called the accuracy of $\mathcal{D}_t$. After training and testing all k SVMs, an accuracy score is obtained for each subset $\mathcal{D}_i$. The cross-validation accuracy is given by the average over all subset accuracies.

The difference between regular k-fold cross validation and stratified k-fold cross-validation is that stratified requires that all subsets contain approximately the same distribution of classifications as the full data set. For example, if the full data set contains 60% positive and 40% negative classifications, then stratified implies each subset also needs to contain 60% positive and 40% negative classifications.

Note that in case a SVM is heavily overfitted, it follows that a low accuracy is obtained when using the SVM to predict the classifications of the samples in the test set. Moreover, to prevent that an overfitted SVM accidentally gives a high accuracy on the test set too, the whole process is performed $k = 10$ times to average out these incidental outliers. In the end, an almost unbiased low-variance estimation of the classifiers accuracy is obtained (Kohavi et al., 1995).

In this section, the problem of overfitting and the necessity of cross-validation was illustrated using accuracy as a scoring function to measure the performance of a SVM. However, the problem of overfitting and the necessity of cross-validation also hold for other performance measure functions, i.e. cross-validation is necessary to obtain an unbiased estimation of any performance indicator function.

6.2 F1 score

A common performance indicator in many classification problems is accuracy. However, Powers (2011) showed that accuracy is not a complete performance measure and other scoring functions provide better insights into the performance of a SVM. The F1-score and the Precision-Recall Area Under The Curve (PR-AUC) score are often seen as better SVM performance indicators.

The F1 and the PR-AUC score are both calculated using a *confusion matrix*. The confusion matrix compares the real classification of samples against the predicted classification and shows the results in a small matrix. Table 6.1 provides an example of a confusion matrix. In this example, 130 out of 200 samples are classified correctly, whereas 70 out of 200 samples are misclassified. Moreover, the confusion matrix in Table 6.1 shows that slightly more of the misclassifications are caused by negative samples that are predicted as the classification positive. To generalize the notion of

		Predicted class		
		Positive	Negative	Total
Actual class	Positive	70	30	100
	Negative	40	60	100
	Total	110	90	200

Table 6.1: *An example of a confusion matrix given two classes, i.e. positive and negative. The values in the table represent the number of samples that met two criteria, i.e. the actual class and the predicted class.*

a confusion matrix, positive samples that are predicted as positive are called true positives (T_P). Also, positive samples that are predicted as negative are called false negatives (F_N). Vice versa, there are also true negatives (T_N) and false positives (F_P). Table 6.2 provides the general form of a confusion matrix.

Using a confusion matrix, two new performance scoring functions can be computed, i.e. the *precision* and *recall* score. Powers (2011) defined the precision of the classification Ω as the number of samples that have predicted and (real) labeled classification Ω divided by the number of predicted samples that have classification Ω. For example, consider the confusion matrix in Table 6.1, i.e. a test set containing 200 samples where 100 samples are labeled as positive. Assume 110 samples are predicted as positive, however, only 70 samples out of these 110 positive predictions are also labeled as positive. Then the precision of the classification positive is given by $\frac{70}{110} = 0.636$. Note that precision can also be seen as classification accuracy using the accuracy definition of Kohavi et al. (1995). In terms of the confusion matrix, the precision of the class 'positive' is given by

$$\text{Precision}_{pos} = P_{pos} = \frac{T_P}{T_P + F_P}. \tag{6.1}$$

Powers (2011) defined the recall of the classification Ω as the ability to correctly identify all samples with classification label Ω. For example, consider again the confusion matrix in Table 6.1, i.e. a test set containing 200 samples where 100 samples are labeled as positive. Assume 110 samples are predicted as positive, however, only 70 samples out of these 110 positive predictions are also labeled as positive. Then the recall of the classification positive is given by $\frac{70}{100} = 0.7$. In terms of the confusion matrix, the recall of the class 'positive' is given by

$$\text{Recall}_{pos} = R_{pos} = \frac{T_P}{T_P + F_N}. \tag{6.2}$$

Similarly, the precision and the recall score of other classifications, e.g. 'negative', is calculated. To obtain an precision or recall score over all classifications, the classification scores are weighted according to their sample distribution in the data set.

Using the precision and the recall score, the F1 score is defined as the harmonic mean of the precision and the recall score (Powers, 2011). Therefore, the F1 score of a classification is given by

$$\text{F1}_{class} = 2\frac{P_{class} \times R_{class}}{P_{class} + R_{class}}. \tag{6.3}$$

The F1 score is a performance indication based on a weighted average between the precision and recall score. In order to obtain a high F1 score, the precision and the

		Predicted class	
		Positive	Negative
Actual class	Positive	True Positives (T_P)	False Negatives (F_N)
	Negative	False Positives (F_P)	True Negatives (T_N)

Table 6.2: *A more generalized confusion matrix in terms of true and false positives or negatives.*

recall score need to be jointly optimized. However, typically there is a conflicting relationship between the precision and recall scores (Raghavan et al., 1989). That is, if one score improves, the other score is likely to decrease. For example, the recall score of a classification can be improved by increasing the number of samples that are predicted as that classification. However, it follows that eventually this leads to more wrong predictions being made, which decreases the precision score. Because of the conflicting nature between the precision and recall score and the harmonic mean average, it follows that the F1 score leads to good performance on both precision and recall simultaneously. Taking precision or recall as the only performance indicators can result in extremely high performance on one indicator, but a very poor performance on the other.

6.3 PR-AUC score

A different commonly accepted performance measurement indicator is the *Precision-Recall Area Under the Curve* (PR-AUC) score (Davis and Goadrich, 2006). The PR-AUC score is calculated by plotting the Precision-Recall curve and calculating the area under the curve. Because the maximum values for precision and recall are equal to 1, the maximum area under the PR curve can also be at most 1.

Multiple precision-recall pairs are calculated for different thresholds to obtain the PR-curve. These thresholds are different cut-off values for the classifier. For example, in the binary classification case, the threshold to determine a samples predicted classification is zero, i.e. $w^T x \geq 0$ (see Chapter 4). However, using different thresholds leads to different classification predictions and thus different precision-recall pairs. The precision-recall pairs are plotted and a curve is obtained from these points using an unique PR interpolation technique (Davis and Goadrich, 2006). Moreover, an 'area under the curve' score is obtained using a simple integral.

Sokolova et al. (2006) stated that the PR-AUC score is particularly useful in applications with imbalanced data sets, i.e. data sets containing much more samples of a particular classification. Imbalance is often the case in experimental sciences such as medical sciences. For example, a study of the effects of different health indicators on a particular disease often contains much more people that do not have the disease. The on-line news articles data set used in this thesis only contains a small imbalance. The small imbalance is solved by removing some samples of the dominant class. The result is a balanced on-line news articles data set. Also, the PR-AUC score is focussed on classification problems containing only two possible classifications. If more classifications are possible, the precision-recall pairs need to be plotted in a higher dimensional space and difficulties occur when trying to calculate the PR-curve or the area under the curve. Therefore, the F1 score is used in this thesis as the performance indicator of a SVM.

7. Binary sentiment analysis

A binary sentiment analysis is performed in this Chapter. That is, a SVM that classifies the sentiment of on-line news articles is trained, tested and evaluated. Binary sentiment classification implies only two possible sentiment classifications are considered, i.e. positive or negative. However, this requires a method to translate the emotion probability vectors of all on-line news articles collected in Chapter 3.1 to binary sentiment labels. Moreover, the best-performing parameters of the SVM such as the penalty and kernel need to be found using a grid search approach and evaluated using the F1 score as performance indicator. After the best SVM parameters are found, the results of the binary sentiment analysis are analyzed and conclusions are made.

The SVM is trained using the SMO algorithm from Chapter 5.2. SMO is incorporated in major mathematical software packages such as MatLab and Scikit-Learn. A software package to solve SVMs using SMO is preferred above coding the SMO algorithm from scratch, because the software packages apply techniques that drastically improve the performance of the SMO algorithm in terms of computation time and memory size. For example, the software packages apply techniques to efficiently store large-scale kernel matrices in the memory, efficiently calculate gradient calculations for the working set selection or kernel function calculations and use parallel computations where possible. The Greenhouse Group prefers open-source software above commercial software such as MatLab due to cross-platform and application implementation issues and money or licensing constraints. Therefore, Scikit-Learn (Pedregosa et al., 2011) was chosen as software package, because it is a well-known and commonly-used open-source machine learning library for Python.

The choice for Scikit-Learn resulted in Python as the programming language. Moreover, because Scikit-Learn is only a library of machine learning functions, Python coding was still essential. For example, Python coding was needed to collect and store the on-line news articles from the web and MySQL databases, to extract and select features from the news articles or to perform a grid search to find the best SVM parameters. The result of this thesis was a deliverable in the form of a Python application. The deliverable handled all necessary tasks for collecting news articles and training, testing and evaluating a SVM. Most importantly, the deliverable was able to make predictions about the sentiment of any new on-line news article.

7.1 Obtaining ‘simple’ sentiment labels

Chapter 3.1 described how all on-line news articles in the data set were classified by readers as one out of six possible emotions, i.e. fascinating, funny, heart-warming, irritating, frightening or depressing. The result was a vector of different emotion probabilities for each news article. However, because the goal is to analyze sentiments, a method is necessary to convert the emotion probabilities to a particular sentiment, i.e. positive or negative.

An intuitive method to convert the emotions into sentiments is by dividing the emotions into positive and negative emotions. Three of the six emotions tend to be positive, i.e. fascinating, funny, heart-warming. The other three emotions tend to be negative, i.e. irritating, frightening and depressing. Therefore, an intuitive and simple method to obtain sentiment labels from the emotion probabilities is by applying a majority-rule principle. A news article is labeled as positive if the sum of the positive emotion probabilities is larger than the sum of the negative emotion probabilities. Vice versa, a news article is labeled negative if the sum of the negative emotion probabilities is larger than the sum of the positive emotion probabilities. Equation 7.1 shows the resulting ‘simple labeling method’.

$$\text{SimpleSentimentLabel}_i = \begin{cases} \text{Positive} & \text{if} \quad \begin{array}{ll} \mathbb{P}(Fascinating)_i & \\ +\mathbb{P}(Funny)_i & \\ +\mathbb{P}(Heartwarming)_i & \geq 50, \end{array} \\ \text{Negative} & \text{if} \quad \begin{array}{ll} \mathbb{P}(Irritating)_i & \\ +\mathbb{P}(Depressing)_i & \\ +\mathbb{P}(Frightening)_i & > 50 \end{array} \end{cases} \tag{7.1}$$

7.2 Creating a data set

The full set of news articles with corresponding emotion probabilities collected in Chapter 3.1 consisted of more than 80.000 various news articles, e.g. articles regarding domestic and foreign news, articles describing sports events and articles describing showbizz-related news. It can be argued that the type of news category also contributes to the sentiment of a news article. For example, foreign news articles often describe negative events from abroad. Therefore, foreign news could be on average more negative than other news categories.

The difference in sentiment between news categories might make it beneficial to train a SVM for each different news category. However, this implies that the predicting power of a SVM is limited to the news category the SVM was trained for. Therefore, it is necessary to know in advance what news category an article belongs to before it is possible to obtain an accurate sentiment prediction. Moreover, additional problems might occur such as not having enough news articles for one news category to properly train a SVM for that news category. Therefore, a general SVM that combines all news categories is considered in this thesis. Also, by combining all categories, the prediction results of one news category might benefit from the found verbal and contextual sentiment relationships in a different news category. However, it is assumed that not all news categories are useful for training the general SVM, because of possible sentiment bias in a news category. Two measures are taken to reduce the bias.

The first measure to reduce the bias is eliminating all articles from the data set that have less than ten votes. This measure is taken to prevent users from 'trolling' and voting a totally different non-fitting emotion on purpose. Because of the anonymity on the internet, trolling is a frequently seen habit. Moreover, using the Weak Law of Large Numbers (Feller, 1971), a higher number of votes imply that the emotion probabilities lie closer to their expected mean values. Therefore, more votes reduces the risk of incidental outliers in the emotion probabilities. The limit is set at ten votes, because this drastically reduces the variance of the emotion probabilities. Moreover, the probability of several people 'trolling' the voting system and thus significantly affecting the emotion probabilities is considered very small. Also, using the limit of ten votes, lots of on-line news articles can still remain in the data set.

The second measure to reduce the bias is eliminating all sports related articles from the data set. An analysis upon the data set showed that the emotion voting with respect to sports articles is very diverse, especially articles about soccer. This diversity can be explained because people are supporting a sports team or are fan of an individual sports player. So, if their team loses they tend to vote negative, whereas people supporting their rivals team or player tend to vote positive. The result is that it is difficult to obtain a unbiased sentiment label for sport articles. Therefore, all sports articles are left out of the data set.

25.099 articles remain in the data set after applying the above two measures. The huge reduction is mainly caused by filtering out the sports articles. For the grid searching part of the binary sentiment analysis, a data set consisting of 5.000 randomly selected articles was created, where half of the articles are labeled as positive and the other half are labeled as negative using the simple sentiment labeling method in Equation (7.1). Note that the number was set at 5.000, because this number allows a SVM to be trained in a reasonable amount of time, i.e. in the order of 10 minutes given that the number of features is less than 40.000. Although the nature of the sentiment analysis problem does not necessarily require such a time constraint, it is very useful for the grid searching part where many SVM parameter combinations are tried using a 10-fold cross validation scheme. Therefore, trying 5 possible combinations of SVM parameters implies training 50 SVMs. After the best-performing SVM parameters for 5.000 articles have been found, a SVM can be trained using more articles. It is assumed that the SVM parameters that performed good for 5.000 articles also perform good for SVMs trained on a higher number of articles. Another 500 articles were randomly selected and taken out of the data set as well. These 500 articles are used to create a manually labeled test set in Chapter

9. The manually labeled test set is used to measure the performance of the labeling method.

7.3 Grid searching

The penalty and kernel parameters of the SVM are optimized using a grid search (see Chapter 4.4.2). To start simple, the following parameters are considered (Hsu et al., 2003).

$$\begin{aligned} C &= \{0.1,\ 0.5,\ 1\}, \\ \text{Kernels} &= \{\text{Linear},\ \text{RBF}\}, \\ \text{RBF}_\gamma &= \{0.0001,\ 0.0005,\ 0.001,\ 0.005\}. \end{aligned}$$

Chapter 3.3 also discussed four possible feature selection methods. Each feature selection method also has one parameter that directly affects the number of features that are selected. Therefore, the grid search is extended with the following feature selection methods.

$$\text{Feature Selection method} = \{\text{None},\ DF,\ IG,\ \chi^2,\ RP\},$$

where the possible corresponding parameters for each feature selection method are given by

$$\begin{aligned} \text{Params}_{DF} &= \text{Min. feature document frequency (\% of documents)} \\ &= \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}, \\ \text{Params}_{IG} &= \text{Percentage (\%) of features with highest IG score} \\ &= \{5, 10, 20, 30, 40, 50, 70, 90\}, \\ \text{Params}_{\chi^2} &= \text{Percentage (\%) of features with highest } \chi^2 \text{ statistic score} \\ &= \{5, 10, 20, 30, 40, 50, 70, 90\}, \\ \text{Params}_{RP} &= \text{Percentage (\%) of features remaining after random projections} \\ &= \{10, 20, 30, 50, 70, 80, 90\}. \end{aligned}$$

The grid search consists of training a SVM using 10-fold cross-validation for each combination of the possible penalty, kernel and feature selection parameters. The possible parameters imply that 4.650 SVMs need to be trained to fully conduct the grid search using 10-fold cross-validation. Given the assumption that training a SVM using 5.000 news articles having less than 40.000 features can be solved in the order of 10 minutes, the full grid search would take at most 775 hours. Because of feature selection, some of the SVMs are small enough in terms of memory to be processed in parallel though. Parallel SVM training significantly reduces the total grid search time. However, a maximum of 5 SVMs can be processed in parallel due to computational reasons.

Even with parallel SVM training, a smarter grid search is necessary to avoid waiting very long for the results. It can be argued that it makes sense to always apply DF as a feature selection method. Without DF, the set of 5.000 articles would contain more than 1.000.000 features. Moreover, most of these features only occur once in the 5.000 news articles. Training a SVM on 5.000 articles having more than one million features leads to very time expensive kernel computations, i.e. 25 million inner products of two vectors having 1 million values. Also, troubles occur when storing a matrix of 5.000 rows and 1.000.000 columns in the memory. When a minimum feature document frequency of 2 is used, the number of features in the data set is already reduced to approximately 85.000. This is a much better usable number of features that does not lead to memory problems. Therefore, DF is a must-have feature selection method.

However, it is not known yet what a good-performing minimum feature document frequency bound is. Therefore, the first part of the grid search is restricted to search for the best *DF* feature selection parameter, i.e. the best minimum feature document frequency bound. This restriction implies 1.050 SVMs need to be trained and tested. Given the approximate upper bound of 10 minutes to train one SVM having less than 40.000 features, this *DF* grid search is done in at most 175 hours. However, the problem size quickly becomes small enough to allow parallel training of 5 SVMS. Moreover, the higher the minimum feature document frequency bound, the less features are selected and the smaller the problem size becomes. A smaller problem size significantly reduces the necessary SVM training time. The parallel training and the gradually decreasing SVM training time makes it possible to perform the full *DF* grid search in approximately 12 hours, e.g. one night. The results of the *DF* grid search are shown in Figure 7.1. The results are also presented in Table A.1 in the Appendix.

Minimum feature document frequency bound	None (0.00%)	3 (0.05%)	5 (0.1%)	10 (0.2%)
Number of features	1.041.822	40.559	19.641	8.569
SVM training time (min.)	Out of 8GB memory error	11.9	6.1	2.7

Minimum feature document frequency bound	15 (0.3%)	20 (0.4%)	25 (0.5%)	30 (0.6%)
Number of features	5.465	4.012	3.162	2.611
SVM training time (min.)	1.6	1.2	1	0.8

Table 7.1: The number of features selected and the SVM training time for different minimum feature document frequency bounds. Note that these values hold for the binary sentiment analysis data set consisting of 5.000 randomly selected news articles.

Using the results of the partial grid search for the best DF bound, the following conclusions were made.

1. **The RBF kernel outperforms the linear kernel.** It is evident that the RBF kernel provides an higher SVM performance for every combination of the proposed parameters. Chapter 4.3.3 stated that the RBF kernel is a more computational expensive kernel than the linear kernel. However, this partial grid search shows that the extra time needed for the RBF kernel is worth the increase in performance.

2. **The RBF kernel performs less as γ increases.** Although the differences are only small, it can be seen that in most cases the performance of a SVM decreases as the value of γ increases. Especially when $C = 1$, this effect is noticeable.

3. **The highest performance scores are obtained when the DF threshold K is below 0.4 %.** It is difficult to state what DF bound performs the best, because the differences are very small. However, the highest performance scores in this grid search, i.e. 0.695, were obtained for multiple bounds below 0.4%. Recall that the higher the DF bound, the smaller the problem size and the faster the SVM is trained. A small problem size is therefore preferred, because it significantly reduces the time necessary to perform the grid search for the additional feature selection methods or to try additional combinations of the kernel and penalty parameters. Therefore, it is concluded that the bound $K = 0.4\%$ performs the best for the binary sentiment analysis.

The main result from the partial DF grid search is that the minimum feature document frequency is set at 0.4%, i.e. a feature has to occur in at least 20 different documents given a training data set of 5.000 articles. Applying this bound on the training data set of 5.000 articles results in only 4.012 features that are selected, a significant reduction of the original number of features. The reduction in features also allows 5 SVMs to be trained in parallel within 2 minutes. In the second part of the grid search, other feature selection methods are evaluated using a DF bound of 0.4%. The option of the linear kernel is no longer considered in the second part of the grid search, because the RBF kernel clearly outperformed the linear kernel in the first part of the grid search. Therefore, the remainder of the grid search consists of iterating over the following parameters.

$$\begin{aligned} C &= \{0.1, 0.5, 1\}, \\ \text{RBF}_\gamma &= \{0.0001, 0.0005, 0.001, 0.005\}, \\ \text{Feature Selection method} &= \{IG, \chi^2, RP\}, \end{aligned}$$

where the possible corresponding parameters for each feature selection method are

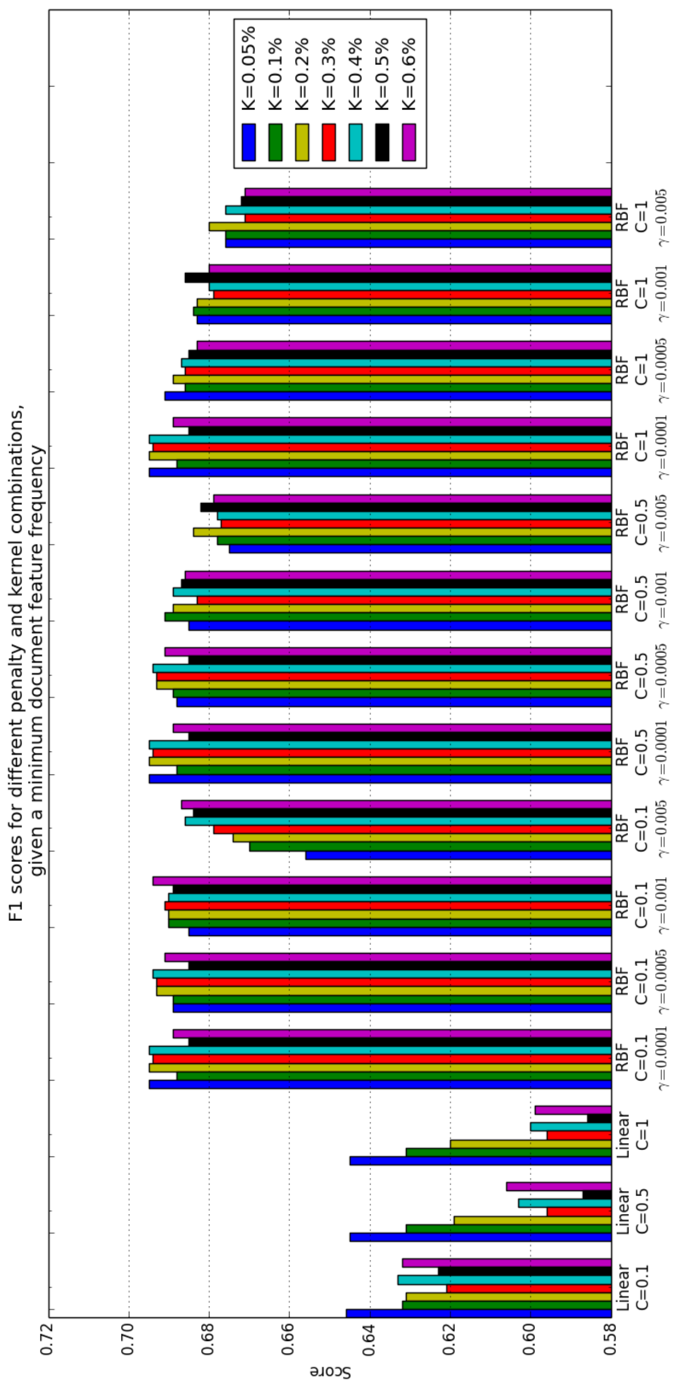


Figure 7.1: Results of the grid search for the best bound of the minimum feature document frequency bound. The search was done for various minimum bounds and different SVM parameters.

given by

$$\begin{aligned}
\text{Params}_{IG} &= \text{Percentage (\%) of features with highest IG score} \\
&= \{5, 10, 20, 30, 40, 50, 70, 90\}, \\
\text{Params}_{\chi^2} &= \text{Percentage (\%) of features with highest } \chi^2 \text{ statistic score} \\
&= \{5, 10, 20, 30, 40, 50, 70, 90\}, \\
\text{Params}_{RP} &= \text{Percentage (\%) of features remaining after random projections} \\
&= \{10, 20, 30, 50, 70, 80, 90\}.
\end{aligned}$$

A full grid search over all the above parameters implies training 2.760 SVMs. Considering the fact that 5 SVMs can be trained in parallel in at most 2 minutes, the full grid search can be done in at most 18 hours. However, all three feature selection methods and corresponding parameters reduce the problem size and thus also reduce the training time of the SVM. Therefore, the grid search can be done in much less than 18 hours. Figures A.1, A.2 and A.3 in Appendix A.1 show the results of the grid search for the three different feature selection methods. The results are also presented in a tabular format in Table A.2, A.3 and A.4 in Appendix A.1. The following conclusions were made using these results.

1. **χ^2 feature selection provides the best performance.** The χ^2 method improves the best SVM performance score to 0.715. This is a small improvement compared to the score of 0.695 with only DF as feature selection method. RP is not able to obtain an increase in the SVM performance score. IG is also able to obtain an higher SVM performance score, but IG is much less consistent than the χ^2 method, i.e. the difference in SVM performance between various IG bounds is much larger than for the χ^2 method. Therefore, the performance of the χ^2 method is much less dependent on the chosen parameter value which is an useful property.

2. **The best performing penalty and kernel parameters are C=1 and γ=0.005.** Although the results show that the values for C and γ have less impact on the results than the method of feature selection, the highest performance score of 0.715 is obtained using χ^2 feature selection with $C = 1$ and $\gamma = 0.005$.

3. **χ^2 feature selection obtains the best results when the 40% highest χ^2-scoring features are selected.** The best selection percentage depends on the penalty and kernel parameters. However, for all possible parameters, the values between 20 and 50% often perform well. When using less than 20% or more than 50% the results tend to decrease, especially for some penalty and kernel combinations. Note that the best score was obtained when using the 40% highest χ^2-scoring features.

4. **Random projection deteriorates the performance.** For all possible values for the kernel and penalty parameter, the SVM performance when using RP

is worse than when not applying RP. This deterioration makes sense, because RP is an unsupervised feature selection method and is based on trading in a small amount of accuracy in return for reducing the dimension of the problem. Therefore, although the dimension of the problem is significantly reduced, RP gives a lower SVM performance.

5. **Random projection is of more use when the number of features is higher.** Although the SVM performance with RP is worse than without RP, the difference in performance is only small when the random projections do not reduce the number of features to less than 30% of the original number. Therefore, RP is useful when the number of features is high, because then a significant increase in speed is obtained in return for a slightly worse performance. However, for this sentiment analysis, the DF and χ^2 feature selection methods reduce the problem size to such a small size that this increased speed is not necessary. Therefore, RP is considered as not useful for this sentiment analysis.

7.4 Results binary sentiment analysis

Note that the best-performing parameters in the grid search were $C = 1$ in combination with the RBF kernel with $\gamma = 0.005$. Also, DF feature selection was applied such that a feature had to occur in at least 0.4% of the documents and. After the DF filtering, only the 40% of the features with the highest χ^2 score were selected. The full binary sentiment analysis is conducted with these best-performing SVM parameters. The full binary sentiment analysis is performed by significantly increasing the number of articles with which to train and test the SVM. Normally, the entire grid search operation has to be repeated when increasing the number of training samples, because the additional samples might change the feature-sentiment relationships that were found for the data set consisting of 5.000 samples. Therefore, it could be that the additional samples lead to different best-performing SVM parameters. However, recall that this grid search is a very time-consuming operation that takes significantly more time when the number of articles increases. Therefore, it is assumed that the best-performing parameters for 5.000 articles also perform well when the number of articles in the training set is increased. This assumption eliminates the need for a time consuming grid search.

Figure 7.2 shows that this assumption seems to hold. The figure shows that as the number of articles in the training set increases, the SVM performance decreases slightly. Several possible reasons for this effect can be given. For one, the best-performing parameters for the grid search using 5.000 articles simply might not be the best-performing parameters when using 22.000 articles. However, recall that it takes too long to perform a full grid search on 22.000 articles, and therefore, the assumption was made that the best-performing parameters for 5.000 articles also perform well on more articles.

Another possible reason is that the randomly selected set of 5.000 articles could be easier to classify, i.e. the articles might contain more sentiment which makes it easier to identify the sentiment. For example, Figure 7.3 provides an example of a SVM trained on a data set consisting of articles that contain few sentiment versus a data set consisting of articles that were voted as a highly positive or highly negative emotion and are thus assumed to contain more sentiment. The figure shows that the data set containing more sentiment resulted in a significantly higher SVM performance. A larger randomly selected data set is more likely to prevent this effect, because the probability of only selecting the articles that contain much sentiment decreases as the number of selected articles increases.

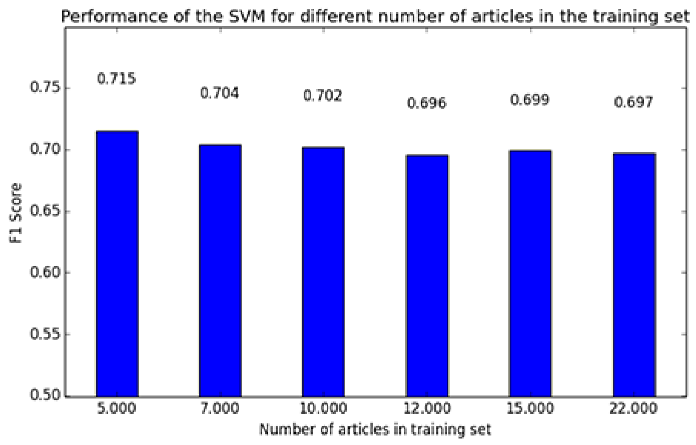


Figure 7.2: Results of the grid search for different number of training samples, given DF=0.4%, $C = 1$ and RBF kernel with $\gamma = 0.005$.

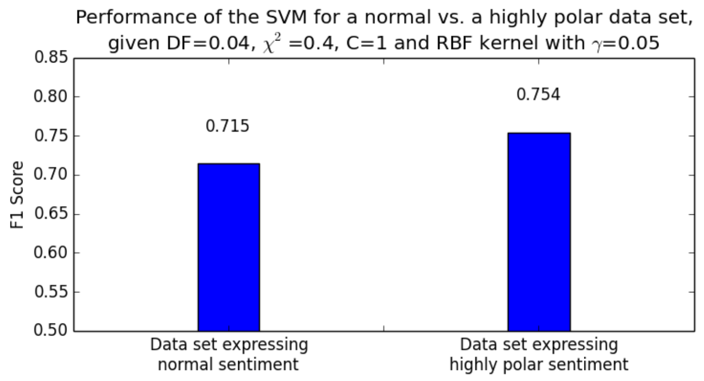


Figure 7.3: Performance of a SVM trained on two different data sets. The first data set consists of news articles containing few sentiment. The second data set consists of news articles that were voted as a highly positive or highly negative emotion and are therefore assumed to contain more sentiment. It can be seen that the second data set gives a significantly better SVM performance.

7.5 Conclusions binary sentiment analysis

The results of the binary sentiment analysis lead to three main conclusions. For one, the performance of the binary sentiment SVM is approximately 0.70. Considering that randomly predicting the sentiment yields an average performance of 0.50, this is not that big of an increase, because it still implies predicting approximately 1 out of 3 articles as the wrong sentiment. More importantly, because wrong predictions are predictions of a totally opposite sentiment, this really limits the range of possible practical applications. For example, an application that analyzes the sentiment of a brand in on-line news articles is difficult when 1 out of 3 articles are predicted as the wrong sentiment. It could be that, due these misclassifications, the sentiment about a brand is falsely interpreted as positive or negative. The latter can then lead to wrong decision making.

Secondly, the binary sentiment analysis shows that articles that contain much sentiment can be classified much better (see Figure 7.3). However, the full data set appears to have a lot of articles that do not contain that much sentiment. Therefore, the full data set is harder to classify as either positive or negative and thus the SVM performance decreases. To improve the results, it is necessary to improve the sentiment labels or the sentiment predictions. These improvements can be realized by either improving the SVM to better deal with the articles that contain few sentiment (Chapter 8) or by improving the method to obtain sentiment labels from the emotion probabilities (Chapter 9).

Lastly, from the grid search on 5.000 articles it followed that the DF and χ^2 feature selection methods provided the best results. Moreover, particular values for the kernel parameters, penalty parameter and DF and χ^2 feature selection methods were found. It was assumed that these parameters would also perform well when using more articles to train and test the SVM. Although the performance decreased slightly when using more articles, the differences were not big enough to state that it was a significant decrease (Figure 7.2). Therefore, this assumption seems to hold and is therefore repeatedly used in the remainder of this thesis.

8. Multi-sentiment analysis

In this chapter, the binary sentiment analysis is extended to a multi-sentiment analysis by introducing the neutral sentiment. The first section discusses why the neutral sentiment is a necessary sentiment to reduce the number of 'heavy' misclassifications, i.e. a prediction of positive while the real sentiment is negative, or vice versa. However, because SVMs are designed to solve binary classification problems, the multi-sentiment analysis cannot be solved using the previously used SVM technique. The second section addresses this problem and discusses three possible heuristics to solve the multi-sentiment analysis problem using SVMs. The chapter concludes with the results of the multi-sentiment analysis.

8.1 Introducing the notion of neutral

The binary sentiment analysis in Chapter 7 only considered two sentiments, i.e. positive and negative. As as result, all news articles were classified as either positive or negative and all features tended to have either a positive or a negative impact on the sentiment. Considering only the positive and the negative sentiment is however a rough simplification of sentiments, because not all news articles or features are positive or negative. For example, a feature such as the phrase 'it is time' does not contain much sentiment and can be either positive or negative. Therefore, it could be considered as a new sentiment classification, i.e. neutral. Despite the presence of a neutral sentiment, the neutral classification is often ignored in sentiment analysis problems, because of the following two assumptions (Koppel and Schler, 2006):

- More can be learned from documents containing a clearly-defined positive or negative sentiment than from documents containing a neutral sentiment.
- When solving the binary positive/negative sentiment analysis problem, neutral articles will lie more to the boundary of the classifier. Therefore, the binary sentiment analysis problem also solves the multi-sentiment analysis problem, i.e. the problem considering the positive, negative and neutral sentiment.

Koppel and Schler (2006) showed that both assumptions do not hold and overall results can be improved when neutral is taken into consideration as a separate sentiment classification. Moreover, the impact of 'misclassifications' can be reduced.

In the binary sentiment analysis, every misclassification implied that a totally opposite sentiment was predicted, i.e. a positive prediction for a negative sample or vice versa. In the remainder of this thesis, these type of misclassifications are called *heavy misclassifications*. With the additional neutral classification misclassifications can also imply that a positive or negative article is classified as neutral. Depending on the desired application of the sentiment analysis problem, classifying a positive or negative sample wrongly as neutral is intuitively less wrong than a heavy misclassification. Moreover, even humans do not always agree whether something is neutral or positive/negative. As the misclassifications in the binary sentiment problem are all heavy misclassifications, it makes sense to extend the binary sentiment problem to a multi-sentiment problem containing neutral as additional sentiment classification to reduce the number of heavy misclassifications.

8.2 Extending the simple sentiment labeling method

Chapter 7.1 discussed a simple method to obtain a sentiment label from the emotion probabilities of an article. However, because of the introduction of the neutral classification, this method is extended to

$$\text{SimpleSentimentLabel}_i = \begin{cases} \text{Positive} & \text{if PositiveRate}_i \geq 75, \\ \text{Negative} & \text{if NegativeRate}_i \geq 75, \\ \text{Neutral} & \text{otherwise,} \end{cases} \tag{8.1}$$

where i represents news article i, PositiveRate$_i$ is given by the sum of the probabilities for the three positive emotions (fascinating, funny and heartwarming) of article i and NegativeRate$_i$ is given by the sum of the probabilities for the three negative emotions (irritating, frightening, depressing) of article i. Using the 'extended simple labeling method' given by Equation (8.1), roughly 50% of the articles in the data set is labeled as neutral. Moreover, only very distinctive articles are labeled as positive or negative. It is expected that this labeling method significantly improves the precision of positive and negative predictions.

8.3 Heuristics to solve the multi-class SVM

The multi-sentiment problem leads to new difficulties in training the machine. Because SVMs are designed to solve binary classification problems, it is not possible to use a single SVM to solve the multi-sentiment problem with three possible sentiment classifications. However, by using heuristics and combining the results of multiple SVMs, it is possible to solve the multi-sentiment problem using SVMs. In this section, three heuristics are discussed and evaluated.

8.3.1 One-versus-all

One of the earliest and most common heuristics to solve the multi-class SVM problem is the 'one-versus-all' approach (Hsu and Lin, 2002a). For each possible classification $c \in C$, a binary SVM is trained with one class being all samples with as label classification c and the other class being all samples whose label is not equal to classification c. Mathematically, the first class gets label $y_i = 1$ and the other class gets label $y_i = -1$. The result is a SVM with only two classes and that can be solved using the standard SVM training techniques such as SMO. Moreover, every SVM results in a separating hyperplane. It follows that, after training $|C|$ binary SVMs, also $|C|$ separating hyperplanes are obtained, given by $w_c^T x + b_c = 0$. Figure 8.1 gives an example of a 3-class problem for which three separating hyperplanes are obtained.

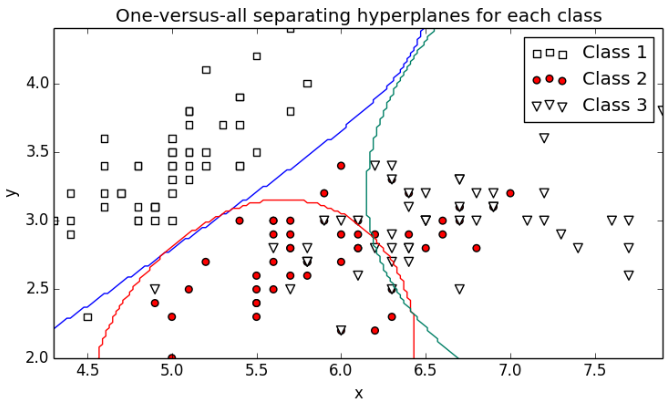


Figure 8.1: Example of three separating hyperplanes found using the one-versus-all heuristic. The used data set is the Iris machine learning data set (Fisher, 1936).

Similarly as in the binary classification problem, the $|C|$ separating hyperplanes are used to make predictions. Note that the distance between a sample point x and the separating hyperplane belonging to class c tells how likely that sample point is to belong to the class c. Also note that this distance is given by $w_c^T x + b_c$. The more positive this distance, the better this points fits in class c. Vice versa, the more negative this distance, the more it fits to the 'rest' class, i.e. all classes except class c. Therefore, the distance $w_c^T x + b_c$ can be seen as a score for class c. The classification c that gives the highest score is selected as the predicted sentiment classification. Figure 8.3 gives an illustration of this prediction process.

For example, assume it is desired to know the prediction of the new sample point $x = \begin{bmatrix} 5 & 3.5 \end{bmatrix}$ for the example in Figure 8.1. From Figure 8.1 it is evident that this new sample point belongs to class 1. Using the generalized prediction approach as given in Figure 8.3, first the respective sentiment scores need to be calculated. The sentiment scores for the point x are given by the distance of x to the three separating

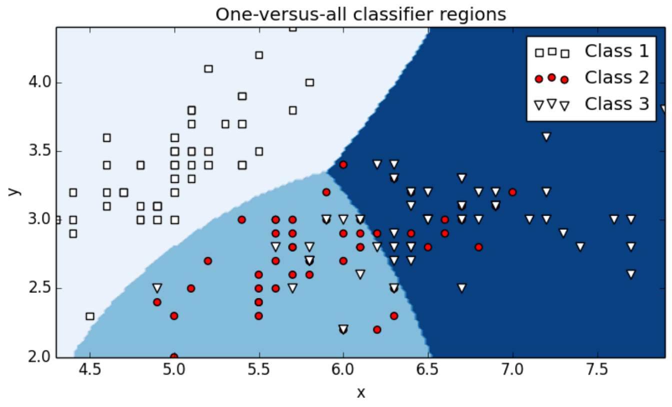


Figure 8.2: The resulting three one-versus-all classification regions for the example in Figure 8.1.

hyperplanes, i.e. $\begin{bmatrix}1.86 & -1.65 & -1.45\end{bmatrix}$. Because the score for the first class is the highest, the predicted classification for point x is 1. Figure 8.2 shows the classifier regions for the example in Figure 8.1 using the one-versus-all heuristic.

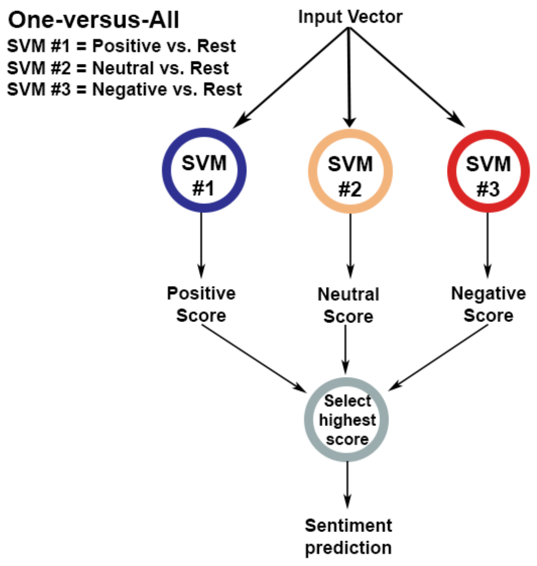


Figure 8.3: Illustration of the prediction proces of the one-vs-all SVM approach.

8.3.2 One-versus-one

Another common heuristic to solve the multi-class SVM problem is the *'one-versus-one'* heuristic (Hsu and Lin, 2002a). The one-versus-one heuristic compares each possible classification $c \in C$ against each other possible classification $c \in C$, i.e. SVMs are trained for all possible pairs of classifications in C. The one-versus-one heuristic results in $\frac{1}{2}K(K-1)$ binary classification problems, where $K = |C|$. These binary classification problems can be solved using the standard SVM training techniques such as SMO. It follows that the one-versus-one heuristic also results in $\frac{1}{2}K(K-1)$ separating hyperplanes, where each hyperplane is functioning as a classifier between two classifications. Figure 8.4 gives an example of a 3-class problem for which three separating hyperplanes are obtained using the one-versus-one heuristic.

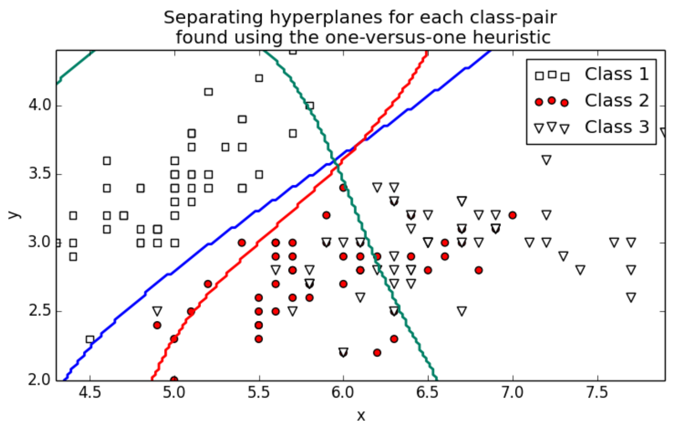


Figure 8.4: Example of three separating hyperplanes found using the one-versus-one heuristic. Note that each separating hyperplanes focusses on separating a classification pair, e.g. class 1 versus class 2 or class 2 versus class 3. The used example data set is the Iris machine learning data set (Fisher, 1936).

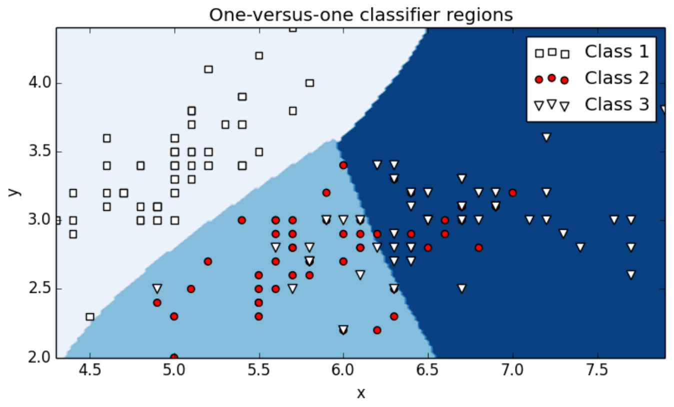


Figure 8.5: The resulting three one-versus-one classification regions for the example in Figure 8.4.

The one-versus-one separating hyperplanes in combination with a majority voting method are used to predict the classification of any new sample point x. First, the predicted classification of the point x for all the $\frac{1}{2}K(K-1)$ separating hyperplanes is calculated. Note that the predicted classification for point x by the separating hyperplane that separates class i and $j \in C$ is given by $y_{(i,j)} = sign\left\{w_{(i,j)}^T x + b_{(i,j)}\right\}$. If $y_{(i,j)} = +1$, the predicted classification of x is class i. Else, if $y_{(i,j)} = -1$, the predicted classification of x is class j. The sample point x is classified as the class that is predicted by the majority of the separating hyperplanes. For example, consider a new sample point $x = \begin{bmatrix} 5 & 3.5 \end{bmatrix}$ in the example in Figure 8.4. The three separating hyperplanes, respectively separating class 1 and 2, class 1 and 3 and class 2 and 3, predict the classifications $\begin{bmatrix} \text{class 1} & \text{class 1} & \text{class 2} \end{bmatrix}$ for the point x. The majority of the separating hyperplanes predict class 1 for the point x. Therefore, the point x is classified as class 1. Figure 8.5 shows the classifier regions for the example in Figure 8.4 using the one-versus-one heuristic.

In case of a tie, the distances to the separating hyperplanes are used as decisive factor. Note that the distance between a sample point and a separating hyperplane that separates class i and class j provides information on how likely the sample point belongs to class i or class j. For example, if the sample point lies very far on the side of the separating hyperplane that classifies a point as class i, this large distance tells that this sample point is very unlikely to belong to class j. Vice versa, a sample point that lies very close to the separating hyperplane is not very convincingly classified as one of the two classes, because a small change in the separating hyperplane or the sample point could lead to a different classification. Therefore, it is assumed that the higher the distance, the higher the probability that the predicted classification is correct. This assumption makes distances a good measure to solve ties in the majority voting method. Figure 8.6 provides a generalized prediction approach for the one-versus-one heuristic.

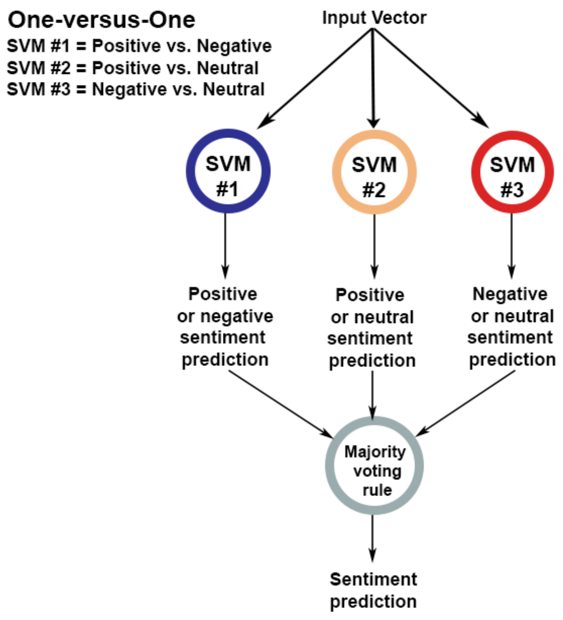

One-versus-One
SVM #1 = Positive vs. Negative
SVM #2 = Positive vs. Neutral
SVM #3 = Negative vs. Neutral
Input Vector
SVM #1
SVM #2
SVM #3
Positive or negative sentiment prediction
Positive or neutral sentiment prediction
Negative or neutral sentiment prediction
Majority voting rule
Sentiment prediction

Figure 8.6: Illustration of the prediction proces of the one-versus-one heuristic.

8.3.3 Hierarchical SVM

A different and less often used approach to solve the multi-class problem using SVMs is hierarchical classification (Silla Jr and Freitas, 2011). Hierarchical classification divides the problem in a hierarchical manner. At the top of the hierarchy, similar classes are combined. Therefore, at the top level a rough prediction about the classification can be made. At lower levels of the hierarchy, the similar classes of the top level are divided into groups of less similar classes, up until the final prediction is a group that only consists of one class. When combining SVMs and hierarchical classification, typically top-down tree-like decision models as in Figure 8.7 are obtained, because of the SVMs binary classification ability.

The advantage of the hierarchical classification heuristic is that relatively easy rough classification distinctions can be made at the top of the hierarchy. At lower levels of the hierarchy, SVMs are trained solely on making much more difficult distinctions between similar classes, without other classes affecting the predictions, i.e. without noise of other classifications. For example, a possible hierarchical SVM heuristic for the example in Figure 8.1 is to first make a prediction whether a point belongs to class 1 or class 2/3. If the prediction of a point is class 2/3, a different SVM is used to make a prediction of whether the point belongs to class 2 or class 3. Note that the latter SVM is therefore strictly trained on points that are labeled as class 2 or 3. Therefore, the resulting best separating hyperplane is thus not affected by points of class 1.

The disadvantage of the hierarchical SVM heuristic is that if a prediction at the top of the hierarchy is wrong, the mistake can not be repaired. As a result, if at the top of the hierarchy a wrong rough prediction about a sample is made, the sample is guaranteed to end with the wrong predicted classification. Such a problem is partly solved by adding interrelationships between classifications. Figure 8.8 gives an example of such a hierarchical classification tree with interrelationships between classifications.

The hierarchical SVM heuristic can also be applied on the multi-sentiment problem. One possible approach is to group the classifications that contain a clear sentiment at the top, i.e. the positive and the negative sentiment. This group is denoted by the *polar(ity)* group. At the top level of the hierarchical classification tree, a SVM predicts whether an article is polar or neutral. If an article is polar, a second SVM is used to predict whether the article is positive or negative. Figure 8.9 gives an illustration of the decision tree of such an *'hierarchical polar-neutral SVM'*.

The hierarchical polar-neutral SVM is not the only possible approach to create a hierarchical SVM for the multi-sentiment problem. A different approach is to make a distinction between the positive and negative sentiment at the top level. If an article is predicted as positive at the top, a second SVM checks whether the article tends to be more positive or more neutral. Vice versa, the same is done for negative predictions. Figure 8.10 gives an illustration of the decision tree of such an *'hierarchical positive-negative SVM'*.

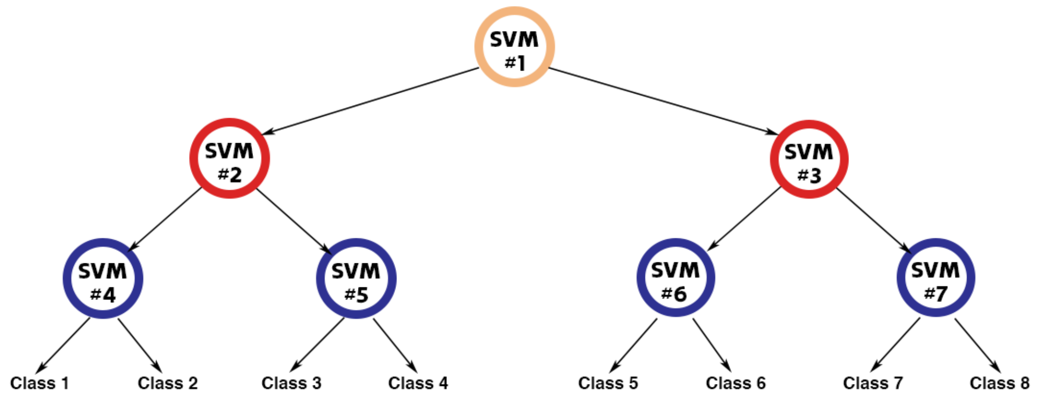


Figure 8.7: A typical tree-like hierarchical SVM structure with no interrelations between nodes.

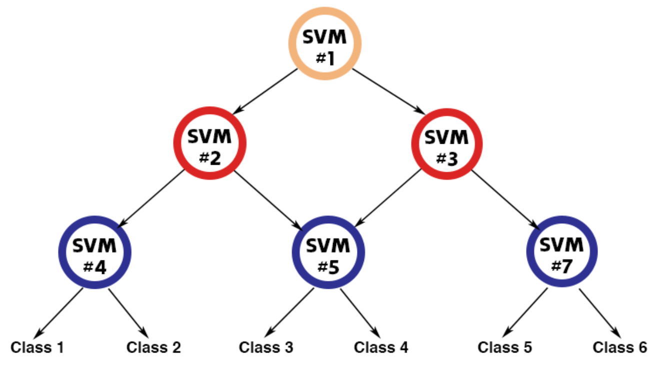


Figure 8.8: A typical hierarchical SVM structure with interrelations between nodes.

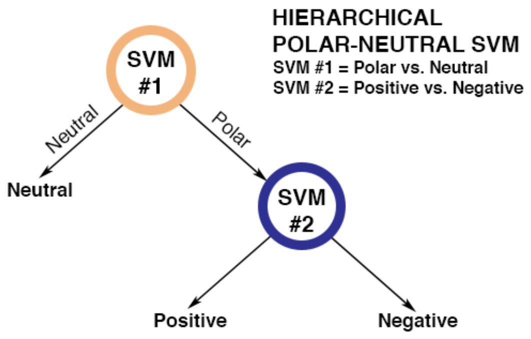


Figure 8.9: A hierarchical SVM approach where the top layer focusses on making a distinction between polar or neutral articles. If an article is polar, the second layer checks whether is positive or negative. This approach is called the 'hierarchical polar-neutral SVM' heuristic.

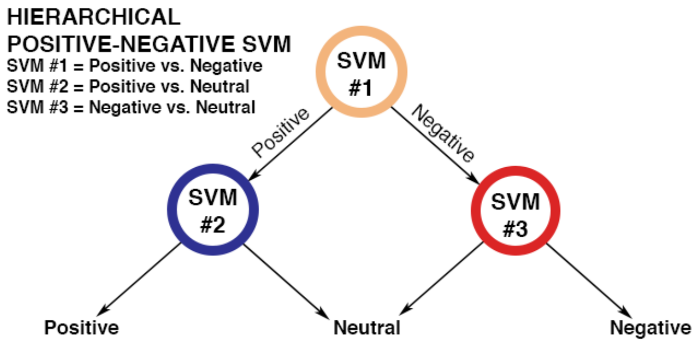


Figure 8.10: A hierarchical SVM approach where the top layer focuses on making a distinction between articles that then to be positive or to be negative. In the second layer it is checked whether an article expresses more polar than neutral sentiment. This approach is called the 'hierarchical positive-negative SVM' heuristic.

8.4 Results

Recall that in order to obtain a good performing SVM, a grid search is often executed to find the best-performing parameters for the kernel, penalty and feature selection methods. In the multi-sentiment case, it is even necessary to perform multiple grid searches, because the one-versus-all, the one-versus-one and the hierarchical SVM methods consist of multiple SVMs. All of these SVMs can be fine-tuned to achieve a better performance.

It is not possible to take the best-performing parameters from the binary sentiment analysis, because the introduction of the neutral sentiment might require different parameter values. For example, it is very likely that neutral and positive articles are more similar than positive and negative articles. Therefore, different kernel parameters might be required to accurately separate the sentiments in a higher-dimensional space. A new grid search is therefore necessary to find the new best-performing SVM parameter values.

Chapter 7.3 already showed that a full grid search is too time-consuming and therefore proposed a faster 'partial' grid search method. This partial grid search method is again applied, because the multi-sentiment problem requires even more grid searches. However, the possible feature selection methods are limited to the χ^2 statistic feature selection method as this method provided the best performance in the binary sentiment analysis. Therefore, the second part of the partial grid search method focuses on finding the best χ^2 statistic parameter. The limitation to the χ^2 statistic feature selection method significantly shortens the second part of the partial grid search method.

Additionally, only 7.500 articles were used for training to speed up the grid search. For each sentiment, 2.500 news articles were randomly selected and added to the training set to obtain a balanced training set. Similar as in the binary sentiment analysis, it is assumed that the best-performing SVM parameters for 7.500 news articles also perform well for SVMs trained with more articles.

Note that individually optimizing the different SVMs within a heuristic does not imply that the overall performance of the heuristic is optimized. To optimize the overall performance of a heuristic, a grid search over all possible configurations of the SVMs in the heuristic needs to be executed. This drastically increases the number of SVMs that need to be trained. For example, consider the one-versus-all heuristic and a grid search method that tries 100 parameter combinations to obtain the best-performing parameters for a single SVM. In this example, $100^3 = 1.000.000$ SVM parameter combinations need to be calculated to obtain the overall best-performing parameters for the one-versus-all heuristic consisting of 3 SVMs. Even when assuming that, in the best case, a SVM can be trained for a single parameter combination in one minute, training 1 million SVMs to grid search over 1 million different SVM parameter combinations takes far too long. Therefore, the grid search for the multi-sentiment analysis is restricted to individually optimizing the SVMs within the heuristics. Additionally, a grid search is executed to find the best-performing joint set of parameters for each heuristic, i.e. using one set of SVM parameters for all SVMs in a heuristic.

Table 8.1 provides the new best-performing parameters for the individual SVMs and the joint SVM parameter heuristics. A full multi-sentiment analysis was executed using the best-performing SVM parameters. Figure 8.11 provides the results of this analysis. It follows that the one-versus-one method slightly outperforms the hierarchical positive-negative method.

One-versus-all SVMs				
SVM	Min. DF-bound (%)	χ^2-bound (%)	C	γ
Positive vs. rest	0.4	0.2	2	0.05
Negative vs. rest	0.4	0.2	2	0.05
Neutral vs. rest	0.4	0.2	2	0.05

One-versus-one and hierarchical SVMs				
SVM	Min. DF-bound (%)	χ^2-bound (%)	C	γ
Positive vs. negative	0.5	0.4	2	0.005
Positive vs. neutral	0.6	0.4	0.5	0.005
Negative vs. neutral	0.6	0.3	0.5	0.005
Polar vs. neutral	0.2	0.2	2	0.01

Joint SVM parameter heuristics				
SVM	Min. DF-bound (%)	χ^2-bound (%)	C	γ
Joint one-versus-all	0.3	0.5	2	0.005
Joint one-versus-one	0.4	0.4	2	0.005
Joint polar-neutral hierarchical	0.3	0.6	2	0.01
Joint pos-neg hierarchical	0.3	0.6	2	0.001

Table 8.1: *The new best-performing SVM parameters for the individual SVMs and joint SVM heuristics obtained using the partial grid search method. Joint implies that all SVMs in the heuristic have identical SVM parameters, e.g. all SVMs in the heuristic having $C = 2$.*

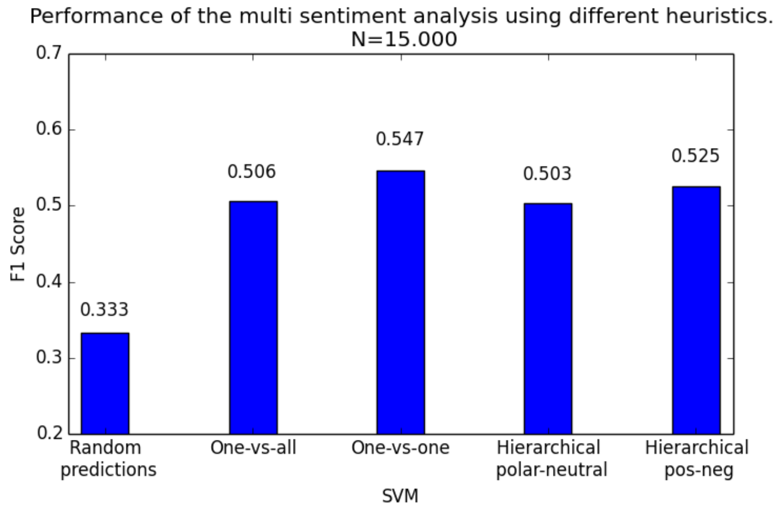


Figure 8.11: A bar chart providing the best performance scores for the different heuristics to solve the multi-class SVM. Note that the 'random predictions column' represents randomly predicting the sentiment of an article.

9. Improving the sentiment labels

The extended simple sentiment labeling method from Chapter 8.2 provided an intuitive, easily implementable and fast method to obtain sentiment labels from emotion probabilities. However, the results of this simple labeling method turn out to very poor when compared to manual sentiment labeling. The first section in this Chapter defines a generalized method to measure the performance of any sentiment labeling method. The second section discusses three new innovative methods to obtain improved sentiment labels by applying a clustering algorithm on the emotion probabilities. The new improved sentiment labeling method is used to update the results of the multi-sentiment analysis. The chapter is concluded by an evaluation and discussion of the new results.

9.1 Performance of a sentiment labeling method

There are several possible methods to obtain sentiment labels from emotion probability vectors. The simple sentiment label method from Chapter 8.2 is only one possible method. For example, a different sentiment labeling method is to label all emotion probability vectors with a low standard deviation as neutral. A low standard deviation makes it difficult to attach the label positive or negative as all emotion probabilities are relatively close to each other. Therefore, the label neutral seems to be the best sentiment label. Another different sentiment labeling method is obtained by adjusting the probability bounds for the sentiment labels in the simple sentiment labeling method of Chapter 8.2, i.e. changing the probability bounds for the positive or negative sentiment label.

In order to compare the different labeling methods, a method to measure the labeling performance is necessary. The F1-score statistic as discussed in Chapter 6 is also well-suited for measuring the labeling performance. The sentiment labels that follow from a sentiment labeling method can be seen as the 'predicted' sentiment labels. However, the 'real' sentiment labels are not known. That is, the overall sentiment readers attach to a news article is not known, only the emotion probabilities. Therefore, it is not possible to generate a confusion matrix from the real sentiment labels and the sentiment labels given by the labeling method. It is not possible to calculate the F1-score directly without the confusion matrix.

The problem of not having the real sentiment labels of news articles is solved by creating a manually labeled test set. Recall from Chapter 7.2 that 500 randomly selected on-line news articles were excluded from the training data set. These 500 news articles were shown to three different users, who labeled every article as one out of the possible three sentiments. Note that the sentiment a user associates with a news article is personal and can therefore differ from person to person. Hence, to obtain a more general overall sentiment label, a majority vote system was used on the three manually obtained sentiment labels to create one sentiment label. The majority-vote sentiment label is considered as the 'real' sentiment label. In the rare case, all three users voted a different sentiment (one positive, one neutral and one negative), the label neutral was assumed to be the majority vote label. For the 500 articles having a real sentiment label, a confusion matrix and F1-score can be calculated to measure the performance of any sentiment labeling method.

Table 9.1 provides the performance statistics of the extended simple labeling method from Chapter 8.2 in terms of precision, recall and F1-score. Note that from the confusion matrix in Table 9.1 it follows that the majority of misclassifications are positive-neutral or negative-neutral misclassifications. Intuitively, these are less wrong than positive-negative misclassifications, because the difference between neutral and a polar sentiment is very personal. This also makes it difficult for the machine to distinguish between the neutral and polar sentiment. Recall that Chapter 8.1 denoted positive-negative misclassifications as 'heavy' misclassifications.

		Labeled class			
		Positive	Negative	Neutral	Total
Real (manual) class	Positive	47	25	65	137
	Negative	3	109	44	156
	Neutral	27	78	102	207
	Total	77	212	211	500

Sentiment	No. of labels in test set	No. of labels by labeling method	Precision	Recall	F1 score
Positive	77	137	0.610	0.343	0.439
Negative	212	156	0.514	0.699	0.592
Neutral	211	207	0.483	0.493	0.488
(Weighted) total	500	500	0.516	0.557	0.507

Table 9.1: *The confusion matrix and general performance statistics for the extended simple labeling method (see Chapter 8.2).*

9.2 Cons of the simple sentiment labeling method

The simple sentiment labeling method is easy to implement and intuitively easy to understand. However, there are some issues with the underlying assumptions. For one, it assumes that every emotion represents one particular polar sentiment, i.e. positive (fascinating, funny, heartwarming) or negative (irritating, frightening and depressing). Secondly, it assumes that the emotions all contain the same amount of sentiment, for example, the positive emotions are all equally positive.

An analysis of the manually labeled test set and the full news articles data set shows that these two assumptions do not hold. For example, the manually labeled test set contains 35 samples that have a 'funny' emotion probability larger than 50%. Out of those 35 samples, only 5 were manually labeled as positive, 10 as negative and 20 as neutral. Therefore, the emotion 'funny' tends to be a neutral emotion when comparing its probability to the manual labels. However, the simple sentiment labeling method considered the emotion 'funny' as a positive sentiment. Therefore, the first assumption fails. Also, a closer look at the 5 positive samples that have a funny probability over 50 % shows that their funny rate is actually very high, i.e. higher than 75 %. So this leads to the assumption that only very high funny probabilities should be considered as positive and thus the second assumption also fails. A better approach could be to use a quadratic relationship on the effect of the emotion funny on the sentiment positive. For lower probabilities, the effect of the emotion funny should be low, whereas higher probabilities should have a higher impact.

Checking if the same problem also occurs for different emotions gives a remarkable result. The as negative considered emotions, i.e. irritating, frightening and depressing, have a really large resemblance with the manually labeled sentiment negative. That is, samples with larger negative emotion probabilities are very likely to also be manually labeled as negative. On the other hand, the as positive considered emotions, i.e. fascinating and heart-warming, tend to suffer the same effect as the emotion funny.

Discussing the issue of translating emotions to sentiments with an expert in the field of Psychology resulted in a referral to the research of Marsha M. Linehan, a professor at the department of Psychology at the University of Washington. Linehan reduces the diversity of emotions to four basic emotions, i.e. *happy, afraid, mad* and *sad* (Linehan, 1993). Each basic emotion also has different gradations. Note that the first basic emotion, happy, can be seen as positive. The other three basic emotions are negative. Using the notion of basic emotions, each emotion can be reduced to a gradation of a basic emotion. For example, the emotion 'terrified' is a high grade of the negative basic emotion afraid. Therefore, the emotion terrified is also very negative. Depending on the gradation of the basic emotion, an emotion can also be neutral. For example, the emotion 'worried' can be seen as a low grade of the basic emotion afraid and is therefore more neutral than negative.

The six emotions of the collected on-line news articles data set can also be mapped to a gradation of the basic emotions. Four of the six emotions are clear gradations of a basic emotion, i.e. irritating→ mad, frightening→afraid, depressing→sad and heart-warming→happy. The other two emotions in the data set, funny and fascinating, are more difficult to map to a single basic emotion, because these two emotions are not *pure* emotions, i.e. the brain is required to understand them. For example, humans consider things as funny, because their brains tells them something is funny. Without the brain, humans are not able to understand the relationship between several objects or events that cause them to find something funny. Similar for the emotion fascinating, the brain is needed to consider something fascinating. For example, the fact that Usain Bolt can run 100 meters in 9.58 seconds is fascinating, because our brain tells us that running that fast is nearly impossible for humans.

Several samples in the data set confirm that, because two of the emotions in the data set are not pure, strange things happen with the sentiment people associate with a news article. For example, a news article titled 'four killed in mortar attack in Somalia' is rated by 33% as funny and by 40% as depressing. The content of the article is clearly very negative though and describes an innocent family with two children being killed in a mortar attack. Using the simple labeling method, this article would be labeled as neutral because of the high probability of the emotion funny. Voting as a reader for the emotion funny in this example seems questionable. Unfortunately the reason why readers vote for funny can not be found out, because of the anonymity of the voting process. However, one possible reason is that people express their madness about a particular subject, for example Somalian piracy or African criminality, through the non-pure emotion funny. In this example, Somalian piracy as the underlying reason implies that the underlying basic emotion is mad and the sentiment of the article is thus negative. On the contrary, different examples can be given where the emotion funny is reduced to the positive basic emotion happy. It follows that the emotion funny or other non-pure emotions are difficult to directly relate to sentiments, as the sentiment of non-pure emotions is largely dependent on the context and other emotion probabilities.

9.3 Semi-supervised clustering

Because the basic assumptions of the simple labeling method do not hold, a new labeling method is required to improve the performance of the labeling. The new method needs to take into account the different non-linear effects of the various emotions on the sentiments. Also, it has to deal with the non-pure emotions that have different effects on the sentiment, depending on the context of the article or the other emotion probabilities.

A clustering algorithm is proposed to capture the complex relationships between the six emotion probabilities and the three sentiments. However, when applying an unsupervised clustering algorithm, the resulting clusters might not relate at all to sentiments, but to other factors such as how well emotions stand out. For example, a sample run of an unsupervised clustering algorithm gave as first cluster all articles where the probability of one of the emotions was very high ($> 80\%$), as second cluster all articles where two or three probabilities were also high but close to each other and as third cluster all other articles. These three clusters can not be directly related to a sentiment, because the problem of assigning an emotion to one sentiment still exists. Applying a supervised clustering algorithm is also difficult, because that would need more than 500 manually labeled samples in order to obtain a good clustering. Therefore, a semi-supervised clustering algorithm is used.

Semi-supervised clustering implies that a subset of the data is labeled, whereas the rest of the data set is not labeled. However, the labeled data provides the basic framework around which the clusters are created. There are several clustering methods that can be used for semi-supervised clustering. A commonly used semi-supervised clustering method is *label propagation* (Zhu and Ghahramani, 2002). The idea of label propagation is that the labeled samples propagate their sentiment label to their neighbors. The propagation is repeated until all samples are labeled.

Zhu and Ghahramani (2002) proposed two different methods to define neighbors. Both methods start by creating a fully-connected graph where each node represents a sample, i.e. an emotion probabilities vector belonging to a news article. The edge for each node i and j is weighted using a weighting function. The first method uses the k-nearest neighbors (kNN) method and the weights are given by

$$w_{ij} = \{1 \text{ if } x_j \in kNN(x_i),\ 0 \text{ otherwise}\},$$

where $kNN(x)$ is the set consisting of the k nodes with the lowest Euclidean distance to x. The second method uses the RBF kernel method and the weights are given by

$$w_{ij} = \exp\left(-\gamma ||x_i - x_j||^2\right),$$

where γ is a sensitivity parameter. Using the RBF kernel results in weights that get significantly larger as the Euclidean distance between x_i and x_j gets smaller. The benefit of the kNN method is that the resulting weighting matrix is a sparse matrix, which is much easier to store in the memory and thus better scalable with the number of samples. The RBF kernel results in a dense and thus less memory-efficient, weighting matrix, but in turn provides a higher clustering performance (Zhu and Ghahramani, 2002).

The label propagation algorithm starts after the weighting matrix has been calculated and the neighbors for each sample are defined and weighted. The algorithm starts by propagating known labels to other nodes. The probability of a label of node x_i being propagated to node x_j is determined by the weights w_{ij}. That is, the probability of a label jumping from x_i to x_j is given by

$$P_{i,j} = \mathbb{P}(x_i \to x_j) = \frac{w_{ij}}{\sum_{k=1}^{N} w_{kj}}.$$

Together all these probabilities form the probabilistic transition matrix P. Additionally, a matrix $Y^t \in \mathbb{R}^{N \times 3}$ is defined, where the ith row represents the sentiment probabilities of node x_i and t represents the iteration number. Note that Y^0 corresponds to all labeled nodes having probability one for their known class label and all unlabeled nodes having probability zero for all three sentiments. At each iteration t of the algorithm, two steps are executed:

Step 1: All nodes propagate their label according to the probabilistic transition matrix, i.e. $Y^t = PY^{t-1}$.

Step 2: All labels of nodes with known labels are reset, i.e. $Y^t_{i,k} = 1$ if x_i has known label k, and zero if x_i has known label not equal to k.

Note that after every iteration, all rows in Y^t are normalized such that $\mathbb{P}(Y^t_i) = 1$. Zhu and Ghahramani (2002) have shown that this algorithm eventually converges to a unique and fixed solution, where every node has one sentiment label with a probability equal to one. The label propagation algorithm was used to obtain new improved sentiment labels. However, there are multiple different methods on how to apply this clustering algorithm. In the remainder of this chapter, three possible methods are discussed.

9.3.1 Clustering on emotion probabilities

The first method is to apply clustering on the six different emotion probabilities. Assume that for each news article, the vector x_i represents the emotion probabilities belonging to article i. Also, a fraction of the samples is labeled using a basic labeling rule that was constructed using the new knowledge of 'basic' emotions. Recall that Chapter 9.2 discussed how emotions are reduced to certain gradations of basic emotions. These basic emotions are then divided into positive and negative basic emotions. Also, it is assumed that if the emotion probabilities are lying relatively close to each other, there is no clear emotion and thus the sentiment tends to be neutral.

A grid search method was executed to find the best probability thresholds for the (semi-)labeling rule. Note that emotions that are very high graded in terms of a basic emotion, such as depressing and frightening, have a lower probability threshold. Also note that the emotion 'funny' was left out because it can be reduced to multiple different basic emotions such as 'happy' and 'anger', both expressing a different sentiment. The grid search method was also used to find the best performing γ-value for the RBF kernel in the label propagation method. Using the grid search to find the best probability thresholds and the value of γ in terms of SVM performance, the best (semi-)labeling rule as given in Equation (9.1) was obtained. The performance of the clustering method and the performance of the SVMs is given in Figure 9.1

and Table 9.2.

$$\text{Label}_i = \begin{cases} \text{Positive} & \text{if } \mathbb{P}(Fascinating)_i \geq 80 \\ & \text{ or } \mathbb{P}(Heartwarming)_i \geq 80, \\ \text{Negative} & \text{if } \mathbb{P}(Irritating)_i \geq 75 \\ & \text{ or } \mathbb{P}(Depressing)_i \geq 60 \\ & \text{ or } \mathbb{P}(Frightening)_i \geq 60, \\ \text{Neutral} & \text{if } \sigma_{\mathbb{P}(emotions_i)} \leq 16, \\ \text{No label} & \text{otherwise.} \end{cases} \quad (9.1)$$

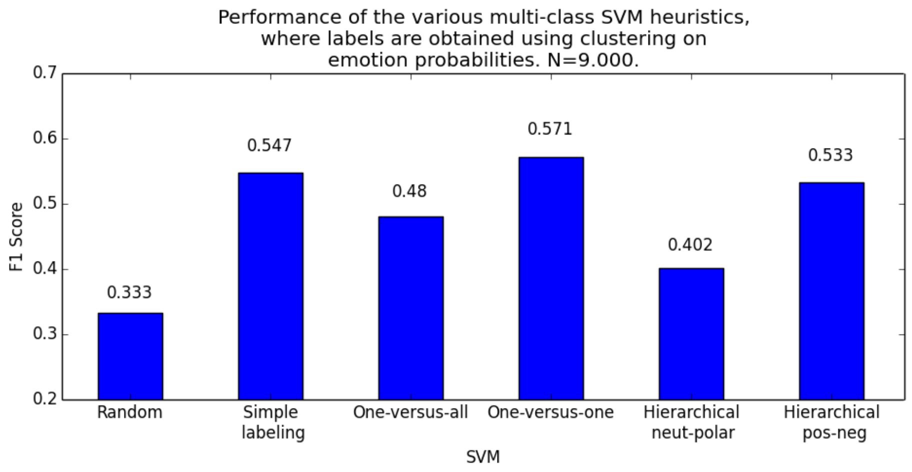


Figure 9.1: Performance of different multi-class SVM heuristics for the clustering on emotion probabilities method.

From the results in Table 9.2 it follows that the labeling performance (0.529) is slightly better than the labeling performance of the extended simple labeling method (0.507). Moreover, Table 9.2 shows that the number of 'heavy' misclassifications is reduced from 5% to only 3%. Additionally, Figure 9.1 shows that the performance of the SVM is improved from 0.547 to 0.571. The combination of a slightly higher sentiment labeling performance, less 'heavy' misclassifications and an increased SVM performance is a desirable outcome, because it results in a more accurate sentiment analysis. Therefore, it can be concluded that clustering improves the overall performance of the sentiment analysis.

9.3.2 Clustering on features

A different clustering approach is to apply clustering on the numeric feature vector representation of the on-line news articles, i.e. to cluster on similar features. The emotion probabilities are still used to construct a rule to label a fraction of the data set. It can be argued that this type of clustering is a form of 'cheating', because it is

		Labeled class			
		Positive	Negative	Neutral	Total
Real (manual) class	Positive	29	4	44	77
	Negative	12	104	96	212
	Neutral	33	47	131	211
	Total	74	155	271	500

Sentiment	No. of labels in test set	No. of labels by clustering	Precision	Recall	F1 score
Positive	77	74	0.392	0.377	0.384
Negative	212	155	0.671	0.491	0.567
Neutral	211	271	0.483	0.621	0.543
(Weighted) total	500	500	0.549	0.528	0.529

Table 9.2: *The confusion matrix and other statistics of the labeling performance when clustering on emotion probabilities.*

very likely that this clustering improves the performance of the SVMs as relatively similar feature vectors are given the same labels. The latter makes it easier for a SVM to obtain a good-performing classifier as similar articles have the same label. On the other hand, because emotion probabilities are not longer taken into account for the clustering part, it also seems very likely that the labeling performance decreases. That is, the obtained sentiment labels are less close to the manually labeled test set. Therefore, the expectation is that there is a trade-off between SVM performance and labeling performance. Similar as before, multiple different basic emotion rules were used to label a fraction of the data set. A grid search method was again executed to find the best probability thresholds for the (semi-)labeling rule and also to find the best performing γ-value for the RBF kernel in the label propagation method. The best performing (semi-)labeling rule is given in Equation (9.2). Figure 9.2 provides the SVM performance of the clustering on features method. Table 9.3 provides the labeling performance.

$$\text{Label}_i = \begin{cases} \text{Positive} & \text{if } \mathbb{P}(Fascinating)_i \quad \geq 75 \\ & \text{ or } \mathbb{P}(Heartwarming)_i \geq 75, \\ \text{Negative} & \text{if } \mathbb{P}(Irritating)_i \quad \geq 75 \\ & \text{ or } \mathbb{P}(Depressing)_i \quad \geq 60 \\ & \text{ or } \mathbb{P}(Frightening)_i \quad \geq 60, \\ \text{Neutral} & \text{if } \sigma_{\mathbb{P}(emotions_i)} \quad \leq 15, \\ \text{No label} & \text{otherwise.} \end{cases} \tag{9.2}$$

From Figure 9.2 it immediately follows that again the one-versus-one multi-class SVM heuristic performs the best. Moreover, the F1 score of 0.600 means a significant increase in SVM performance compared to the simple labeling method score of 0.547.

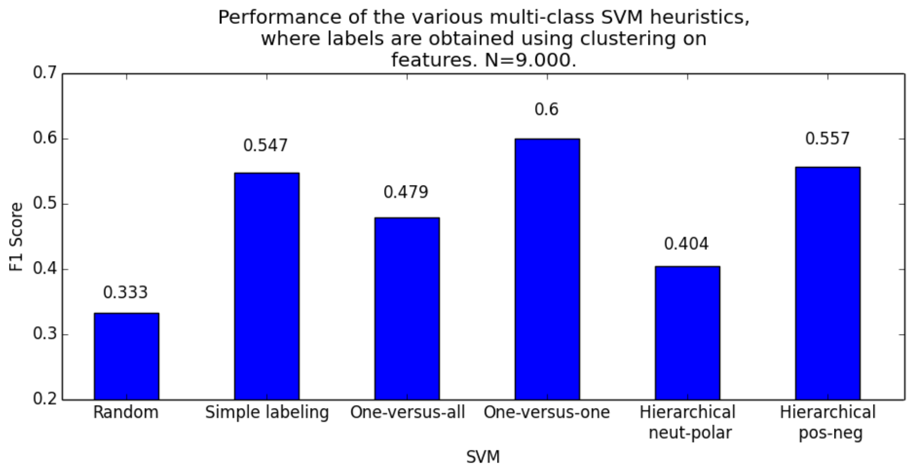


Figure 9.2: Performance of different multi-class SVMs for the clustering on features method.

		Labeled class			
		Positive	Negative	Neutral	Total
Real (manual) class	Positive	6	5	66	77
	Negative	9	65	138	212
	Neutral	11	14	186	211
	Total	26	84	390	500

Sentiment	No. of labels in test set	No. of labels by clustering	Precision	Recall	F1 score
Positive	77	26	0.231	0.078	0.117
Negative	212	84	0.774	0.307	0.439
Neutral	211	390	0.477	0.882	0.619
(Weighted) total	500	500	0.565	0.514	0.465

Table 9.3: *The confusion matrix and other statistics of the labeling performance when clustering on features.*

However, note that the labeling performance of this method is not that good. Table 9.3 shows that the majority of the predictions is set to neutral, allowing for a high neutral recall score and thus a high neutral F1 score. The high neutral F1 score results in a weighted labeling performance F1 score of **0.465**, otherwise it would have been much lower. Another remark is that the clustering on features method does not perform well at distinguishing positive and neutral articles. Almost all positive articles are predicted as neutral. On the other hand, the precision of the negative predictions is very high, so the negative predictions that are made are very reliable. The negative recall score is however very low, so a lot of the negative articles are not identified. That is, two third of them are actually predicted as neutral.

9.3.3 Clustering on features using manually obtained labels

In the previous two methods, the notion of basic emotions and their various gradations was used to construct basic rules to obtain sentiment labels for a fraction of the data set. However, a different approach is using the manually obtained labels from the test set. It is assumed that these labels are very close to the 'real' sentiment classification of an article, because three persons gave their sentiment opinions about those articles. A disadvantage of this method is that only 500 articles were manually labeled, because it is a very time-expensive and costly method. Nonetheless, the hypothesis is that the 500 manually labeled articles might make it possible to obtain good sentiment labels for the remainder of the samples.

Label propagation was again used as the clustering algorithm. Also, the clustering was again done on the feature vectors, because from the previous results, the clustering on features method seems the most promising in terms of SVM performance. A grid search was executed to obtain the best performing value for the RBF kernel parameter γ of the label propagation method. Figure 9.3 provides the SVM performance for the newly obtained sentiment labels.
It can be seen that, compared to all previous results, this clustering method provides the best SVM performance. That is, the one-versus-one method achieves a performance F1 score of 0.626. Moreover, note that in the previous clustering on features method, the hierarchical polar-neutral SVM heuristic performed very poor (0.402). This was caused by difficulties in distinguishing neutral and polar articles, because the majority of the articles were predicted as neutral. However, using this new clustering approach, the hierarchical polar-neutral SVM heuristic performs much better. Therefore, this new clustering approach results in sentiment labels that are easier for the SVM to distinguish between neutral and polar. One of the possible reasons for this is that, in this case, the clustering approach gives much more neutral labels than the previous methods. That is, 64% of the 24.599 articles in the data set are labeled as neutral. Therefore, the articles that are labeled as positive or negative might contain a more clearly defined polar sentiment, making it easier to identify the sentiment and a higher SVM performance is achieved.

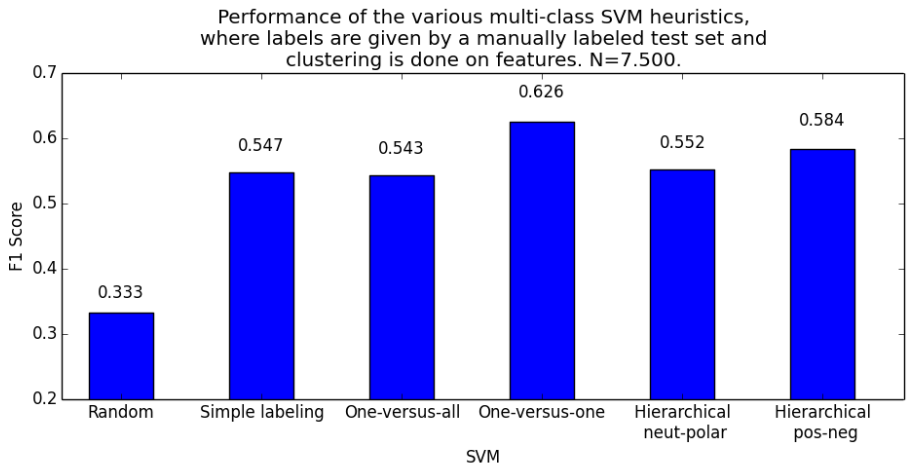


Figure 9.3: Performance of different multi-class SVMs for the clustering on features method.

The drawback of this clustering method is that it is very difficult to accurately measure the labeling performance. Recall that in the previous sections, the labeling performance of the clustering method was measured by comparing the test set labels of the clustering method against the manually obtained test set labels. However, in this case, the manual test set labels were used to create the clustering method. The result is that, when using this clustering method on the test set, all labels are equal to their manually obtained label. As a result, the labeling performance F1 score is equal to 1.00, a perfect fit. Common sense and experience tells that whenever a perfect fit is obtained, there is likely to be overfitting or a bug in the program. In this case, there is a drastic overfit, making it impossible to obtain an unbiased and reliable labeling performance score.

One method to solve this issue would be to create a new second manually labeled test set. The labeling performance of the clustering method can then be calculated using this second manually labeled test set. However, creating such a second test set requires again three persons to manually label 500 articles, a very time-expensive and thus costly procedure. A different method is by applying a stratified 10-fold cross validation procedure on the clustering labeling performance as well. That is, 450 out of the 500 articles in the manual labeled test set are selected and used in the clustering method. The clustering labeling performance is then calculated based on the remaining 50 articles in the manual labeled test set. Stratified 10-fold cross-validation was used to obtain an unbiased estimator (see Chapter 6.1) and a grid search was performed to find the best value of γ. The result is a very poor labeling performance of **0.362** for the clustering on manual sentiment labels method. So, although the SVM performance of the clustering on manual sentiment labels method is higher than the other clustering methods, the labeling performance is very worse and even close to random. It follows that the predicted sentiments using this clustering method are not very similar to the ‘real’ manual obtained sentiment labels. A low similarity between the predicted and the ‘real’ sentiment makes it difficult to obtain reliable conclusions from the SVM sentiment predictions. Therefore, decision making based on the predicted sentiments can be totally wrong.

9.4 Conclusions on clustering

The main conclusion of this chapter is that a clustering approach is able to obtain better sentiment labels and improves the overall quality of the sentiment analysis. That is, the clustering on emotion probabilities method increased the performance of the SVM while simultaneously achieving a slightly higher labeling performance with less ‘heavy’ misclassifications. It was shown that other clustering methods were able to obtain a higher SVM performance, but a lower labeling performance.

Note that both factors, i.e. the SVM and the labeling performance, together define the overall quality of the predicted sentiments of a SVM. For example, a high SVM performance implies that the SVM can very accurately predict the sentiment that was labeled to a news article. However, if the labeling performance was very low, this implies that the sentiment that was labeled to a news article is not very close to the ‘true’ sentiment, i.e. the sentiment humans attach to a news article. Therefore, the overall quality of such a SVM is low.

10. Sentiment analysis in practice

In this chapter, a practical application of the sentiment analysis algorithm is given. As all research for this thesis was done at an on-line marketing company, this practical application was designed to create new insights into on-line marketing strategies. Also, because the company monitors the on-line media campaigns for several top brands in The Netherlands, lots of relevant data was available for analysis.

10.1 Hypothesis

The practical application focuses on researching the relationship between the sentiment of on-line news articles and the performance of on-line advertising campaigns shown next to on-line news articles. Therefore, the following null hypothesis was researched.

Null hypothesis 1 *The sentiment of an on-line news article does not affect the click-through rate of an on-line advertisement shown next to the article.*

The click-through rate (CTR) is a common marketing term and is a method to measure the performance of an online advertising campaign. It is defined by the number of clicks an online advertisement or campaign receives (i.e. the number of clicks) divided by the number of times the advertisement or campaign was shown to an user (i.e. the number of impressions). The result is a score between 0 and 1 representing the ratio of impressions of an advertisement or campaign that resulted in a click on that advertisement or campaign.

$$CTR = \frac{Clicks}{Impressions}.$$

As more users click on an advertisement, more users are redirected to the companies promotional website. Note that the latter results in possible sales or engagement and therefore, more users on a companies promotional website is considered to be better. However, because companies nowadays have to pay per impression of an advertisement, it is desirable to have a large number of impressions converted into clicks and thus to keep the cost per click (CPC) low. Otherwise, the possible sales resulting from clicks do not outweigh the costs of generating clicks. It follows that a higher CTR thus implies a higher advertisement or campaign performance.

The null hypothesis states that the sentiment of an on-line news articles does not affect the CTR of an on-line advertisement shown next to the article. However, it could be for example that a positive news article results in a happier user who is more likely to buy the product that is advertised on an advertisement. Therefore, such a happier user is thus more likely to click on the advertisement. Vice versa, it could also be the case that negative news articles result in depressed users who are more likely to buy products to feel better.

10.2 Results

The hypothesis was researched by analyzing the results of an on-line advertising campaign of a particular major brand in The Netherlands (confidential) on one of the largest Dutch on-line news sites, i.e. NU.nl. Over the course of half a year a particular advertisement was shown to readers of that news site. Note that the content of the shown advertisement was independent of the content of the news articles. However, advertisements can be targeted to a specific group of users, e.g. desktop or mobile users. Therefore, the results of this research might only hold for this specifically targeted group, unless it can be shown that this targeted group is a thorough representation of the Dutch population.

All impressions of the advertisement resulted in lots of clicks on the advertisement. Table 10.1 provides an overview of the impressions, clicks and click through rates grouped by the predicted sentiments of the on-line news articles. Note that all non-news articles of the news website such as index and news overview pages are filtered out of the results.

Predicted sentiment	Number of articles	Number of clicks	Number of impressions	CTR (%)
Positive	338	c	$1.43d$	$0.6172\text{-}\epsilon$
Negative	284	$1.07c$	d	$0.6405\text{-}\epsilon$
Neutral	473	$1.74c$	$2.11d$	$0.6249\text{-}\epsilon$
Total	1.095	$3.81c$	$4.54d$	$0.6259\text{-}\epsilon$

Table 10.1: An overview of the number of articles, clicks, impressions and CTR for each sentiment. The real results are obfuscated, because of confidentiality reasons.

It follows that negative articles result in higher click-through rates. However, it is first tested whether the differences in CTR are statistically significant. By calculating the variances for each sentiment, the t-test scores in Table 10.2 were obtained. Using the t-test scores, it is concluded that the CTR of positive with respect to negative news articles is statistically different at a 95% confidence level. That is, the negative CTR is statistically significant higher than the positive CTR. Therefore, negative articles perform better than positive articles in terms of CTR. and the sentiment of an on-line news article can thus affect the CTR of an advertising campaign shown on the same page. Therefore, the null hypothesis is rejected when comparing the CTR of positive with negative on-line news articles.

Unfortunately, the CTRs of the positive or negative sentiment with respect to the neutral sentiment are not statistically different at a 95% confidence level. Therefore, it can not be concluded with certainty that the CTR of negative articles is always higher than for neutral articles or the CTR for positive articles is always lower than neutral articles. Therefore, the null hypothesis can not be rejected when comparing the CTR of positive or negative news articles with the CTR of neutral articles.

Non-paired t-test	Positive	Negative	Neutral
Positive	-	**-2.21**	-0.97
Negative	**2.21**		1.53
Neutral	0.97	-1.53	

Table 10.2: The unpaired t-test scores for testing whether the CTRs are equal at a 95% confidence level. From the t-test scores it follows that the positive and negative CTRs are not equal with respect to each other. That is, the negative CTR is indeed higher than the positive CTR.

11. Summary and future research topics

The main contribution of this master thesis is a sentiment analysis tool. This tool is able to predict the sentiment of most Dutch on-line news articles, assuming the news articles are grammatically well-structured and syntactically correct. The creation of the sentiment analysis tool consisted of solving four different subproblems, i.e. collecting news articles, obtaining sentiment labels for the collected news articles, processing the news articles to an usable machine learning format and creating a sentiment classification algorithm.

The first subproblem, collecting data, was solved by collecting news articles from multiple Dutch on-line news sites. The benefit of using on-line news sites as data source is that these sites provide a large number of news articles, sometimes dating back to the year 2000. Moreover, the number of news articles on these news sites still significantly increases as more news is published every day. The result after investigating only a few news sites was a data set consisting of more than 2 million on-line news articles.

The second subproblem, obtaining sentiment labels for the collected news articles, was partially solved by using a news site that provided an emotion poll next to most of the news articles. Note that sentiment labels are necessary to let the classification algorithm learn to recognize sentiments. Similar as humans, these algorithms first need to know, for example, how a cat looks like in order to be able to recognize cats. The emotion poll enabled readers of a news article to vote for the emotion the reader attached to that news article. The result of this emotion poll was that more than 80.000 news articles had six different emotion probabilities, e.g. an article could have a 80% probability of being ‘depressing’ and a 20% probability of being ‘irritating’. Multiple methods to translate the emotion probabilities into sentiment labels were discussed in this thesis. A ‘simple’ labeling method was dividing the six emotions into positive and negative emotions. The sentiment of an article was then determined based on the sum of the positive and negative emotion probabilities. However, it turned out that this simple method had some flaws, e.g. an emotion such as funny can be positive and negative depending on the context. A different method to obtain sentiment labels from the emotion probabilities was applying a clustering algorithm on the emotion probabilities. The clustering on emotion probabilities method provided a slightly better labeling *and* SVM performance than the simple labeling method. Two different clustering methods were also able to obtain a higher

SVM performance, but performed worse in terms of labeling performance, i.e. a lower similarity between the obtained cluster sentiment labels and the ‘real’ manual sentiment labels.

The third subproblem, processing data to an usable machine learning format, was solved by applying feature extraction and feature selection. Feature extraction extracts words and phrases from the news articles and translates these words and phrases into weighted numerical features. The ‘feature presence’ weighting method was considered to be the best feature extraction weighting method as it resulted in a sparse and binary data set. The sparsity and binarity of the data set is a very useful property, because it prevents memory and computational issues with the very large size of the data set. Feature extraction led to a very high number of features, e.g. more than 1 million features for a small data set consisting of 5.000 news articles. As most classification algorithms can not deal with a problem of such a large size, feature selection was necessary to reduce the size of the problem. Most feature selection methods do this by removing irrelevant features, i.e. features containing few sentiment. The χ^2 statistic numerically proved to be the best feature selection method, because it gave the highest SVM performance. Moreover, it significantly reduced the number of features, i.e. from more than 1 million features to ± 5000 features considering a data set of 5.000 news articles.

The fourth subproblem, the creation of a sentiment classification algorithm, was solved by applying machine learning techniques. A literature study showed that the support vector machine learning algorithm provides a consistently high performance for most classification problems. The literature study also showed that SVMs perform very good in handling the large dimensionality of sentiment-classification problems. However, the original SVM optimization problem still suffered three problems. For one, the original SVM optimization was still limited to small data sets as it does not scale well with an increasing number of samples in a data set. Secondly, the original SVM optimization problem was limited to a linear separation between classifications, which turned out to give inaccurate results when separating news articles based on their sentiment. Third, the original SVM optimization problem was designed for handling binary classification problems, i.e. only two possible classifications. As the sentiment analysis problem consists of three possible classifications, i.e. positive, negative and neutral, the original SVM optimization problem is thus not suited for solving the sentiment analysis problem.

Several improvements were made to the SVM algorithm to solve the above described three SVM problems. That is, the original SVM optimization problem was improved using the ‘kernel trick’ to provide for better separating between sentiments. It was numerically shown that this kernel trick resulted in a higher SVM performance. Moreover, the original SVM optimization problem was reduced to its Lagrangian dual problem to allow the SVM algorithm to use large-scale data sets without running into time or computational problems. Lastly, four possible heuristics were proposed to solve the 3-class sentiment analysis problem while using the binary classification abilities of SVMs. The one-versus-one heuristic, where a SVM was trained for each possible pair of classifications, proved to give the highest SVM performance.

By combining all four sub-problems, a SVM algorithm was obtained that learns to recognize sentiments of on-line news articles by providing it a large collection of on-line news articles for which the sentiment is already known and attached as a label. The SVM algorithm uses the learned knowledge about sentiments in news articles to predict the sentiment of other on-line news articles for which the sentiment is not yet known. A full sentiment analysis showed that the SVM algorithm was able to give fairly good predictions about the sentiment of on-line news articles. The majority of the wrong predictions turned out to be positive-neutral or negative-neutral misclassifications where one of the polar sentiments is misclassified as neutral or vice versa. Note that the difference between a polar and the neutral sentiment is often very subjective and even humans do not always agree on whether something is, for example, positive or neutral. Therefore, it was assumed that these polar-neutral misclassifications are better than 'heavy' positive-negative misclassifications where a news article is falsely classified as a totally opposite sentiment. Therefore, considering the relatively high number of correct classifications (depending on the used clustering method and multi-class heuristic) and the fact that the majority of the wrong predictions are polar-neutral misclassifications, it was concluded that the final SVM algorithm provides usable and accurate sentiment predictions for on-line news articles.

The final SVM algorithm was used to provide an example of sentiment analysis in practice. The example showed that the sentiment of an on-line news article can affect consumer behavior. That is, users reading a negative on-line news article are more likely to click on an advertisement next to the article than users reading a positive on-line news article. The result of this example may affect the decision making of a company or other advertising strategies. For example, as clicks lead to possible sales, it could be desirable to only show your advertisement next to negative news articles.

11.1 Future research

Although this master thesis covered a wide spectrum of expertises and possible approaches to create a sentiment analysis tool, there are still several possible topics for future research. One interesting topic for future research is *changing the scope* of the sentiment analysis from a document level to a sentence level. The current sentiment analysis tool looks at the entire article to determine the sentiment of a news article. This leads to several problems though. For example, analyzing at the document level implies that all sentences of a news article are important. However, a news article often contains only a few relevant key sentences that contain the sentiment of a news article such as the title or the intro paragraph. Other sentences are used to provide, for example, background on the subject matter and do not really contribute to the sentiment of the article. When analyzing at a sentence level, these key sentences could be identified and be used to determine a 'better' overall sentiment of the article.

Another interesting future research topic is *improving the SVM parameter grid search* as the grid search is the most time-consuming step in the entire process of creating an accurate sentiment analysis tool. Additionally, the SVM parameters largely determine the classification performance of the SVM and thus well-performing SVM parameters are essential. The grid search method discussed in this thesis is based on a 'simple' iteration over all possible parameter values which are multiplied by a factor five at each iteration, e.g. $\gamma = \{0.0001, 0.0005, 0.001, 0.005\}$. Hsu et al. (2003) claimed that this simple grid search approach is not much slower than other SVM parameter estimating heuristics, because the simple grid search approach is very scalable. Every combination of SVM parameters is independent of the other combinations and thus the simple grid search can be performed in parallel for multiple different SVM parameter combinations. Better heuristics for finding well-performing SVM parameters are often faster, but are less scalable as they often use iterative processes which are not independent of each other. Therefore, these better heuristics are harder to perform in parallel and are thus often not faster than a parallel-executed simple grid search. Creating an improved heuristic that is faster than the simple grid search approach *and* can be executed in parallel could significantly reduce the time for the entire grid search operation.

Lastly, a different possible future research topic is developing a *robust model to deal with uncertainty in the sentiment labels.* Recall that the sentiment labels in this master thesis were obtained by applying a clustering algorithm on emotion probabilities. For one, this clustering algorithm therefore introduces a certain degree of uncertainty in the sentiment labels, because not all news articles are necessarily labeled correctly as was shown by the labeling performance indications in Chapter 9. Moreover, the emotion probabilities were obtained by letting readers vote for the emotion they attach to the news articles. This voting process also induces a new kind of uncertainty in the sentiment labels because readers could deliberately vote for a different emotion. For example, the sensitivity of a political news article might make readers vote for a different emotion than they actually attach to an article. The latter also emphasizes the fact that sentiments are subjective, i.e. they can differ from person to person and there is not necessarily one 'correct' sentiment label. Therefore, a more robust model to deal with the uncertainty in the sentiment labels might be useful.

References

A Aizerman, Emmanuel M Braverman, and LI Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25:821–837, 1964.

Shlomo Argamon-Engelson, Moshe Koppel, and Galit Avneri. Style-based text categorization: What newspaper am i reading. In *Proc. of the AAAI Workshop on Text Categorization*, pages 1–4, 1998.

Michael W Berry and Jacob Kogan. *Text mining: applications and theory*. John Wiley & Sons, 2010.

Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM, 2001.

Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2009.

John W Chinneck. Practical optimization: a gentle introduction. *Systems and Computer Engineering), Carleton University, Ottawa. http://www. sce. carleton. ca/faculty/chinneck/po. html*, 2006.

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

Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

Rameswar Debnath, Masakazu Muramatsu, and Haruhisa Takahashi. An efficient support vector machine learning method with second-order cone programming for large-scale problems. *Applied Intelligence*, 23(3):219–239, 2005.

Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. Working set selection using second order information for training support vector machines. *The Journal of Machine Learning Research*, 6:1889–1918, 2005.

William Feller. Law of large numbers for identically distributed variables. *An introduction to probability theory and its applications*, 2:231–234, 1971.

Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

Michael Gamon. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In *Proceedings of the 20th international conference on Computational Linguistics*, page 841. Association for Computational Linguistics, 2004.

Tobias Glasmachers and Christian Igel. Maximum-gain working set selection for svms. *The Journal of Machine Learning Research*, 7:1437–1466, 2006.

Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12, 2009.

Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002a.

Chih-Wei Hsu and Chih-Jen Lin. A simple decomposition method for support vector machines. *Machine Learning*, 46(1-3):291–314, 2002b.

Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification, 2003.

Thorsten Joachims. *Text categorization with support vector machines: Learning with many relevant features*. Springer, 1998.

William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

Jussi Karlgren and Douglass Cutting. Recognizing text genres with simple metrics using discriminant analysis. In *Proceedings of the 15th conference on Computational linguistics-Volume 2*, pages 1071–1075. Association for Computational Linguistics, 1994.

Vojislav Kecman. *Learning and soft computing: support vector machines, neural networks, and fuzzy logic models*. MIT press, 2001.

Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, volume 14, pages 1137–1145, 1995.

Moshe Koppel and Jonathan Schler. The importance of neutral examples for learning sentiment. *Computational Intelligence*, 22(2):100–109, 2006.

Ping Li, Trevor J Hastie, and Kenneth W Church. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 287–296. ACM, 2006.

Marsha Linehan. *Cognitive-behavioral treatment of borderline personality disorder.* Guilford Press, 1993.

James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela H Byers. Big data: The next frontier for innovation, competition, and productivity. 2011.

Prem Melville, Wojciech Gryc, and Richard D Lawrence. Sentiment analysis of blogs by combining lexical knowledge with text classification. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1275–1284. ACM, 2009.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning.* MIT Press, 2012.

Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources. In *EMNLP*, volume 4, pages 412–418, 2004.

Tim O'Keefe and Irena Koprinska. Feature selection and weighting methods in sentiment analysis. *ADCS 2009*, page 67, 2009.

Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*, 2010.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

John Platt et al. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

Vijay Raghavan, Peter Bollmann, and Gwang S Jung. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems (TOIS)*, 7(3):205–229, 1989.

Carlos N Silla Jr and Alex A Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72, 2011.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Conference on Empirical Methods in Natural Language Processing*, 2013.

Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *AI 2006: Advances in Artificial Intelligence*, pages 1015–1021. Springer, 2006.

Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics, 2002.

Vladimir Naumovich Vapnik and Samuel Kotz. *Estimation of dependences based on empirical data*, volume 41. Springer-Verlag New York, 1982.

Jonathan J Webster and Chunyu Kit. Tokenization as the initial phase in nlp. In *Proceedings of the 14th conference on Computational linguistics-Volume 4*, pages 1106–1110. Association for Computational Linguistics, 1992.

Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420, 1997.

Qiang Ye, Ziqiong Zhang, and Rob Law. Sentiment classification of online reviews to travel destinations by supervised machine learning approaches. *Expert Systems with Applications*, 36(3):6527–6535, 2009.

Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.

Notation

C	$C = \{c_1,\ c_2,\ c_3\} = \{\text{Positive},\ \text{Negative},\ \text{Neutral}\}$ denotes the set of possible sentiment classifications of an on-line news article.
D	Denotes the number of features after feature extraction and selection transformation Θ has been applied on all N news articles.
F	$F = \{f_1,\ f_2,\ ...,\ f_D\}$ denotes the set of all D features after the feature extraction and selection transformation Θ has been applied on all N news articles.
M	$M = \{m_1,\ m_2,\ ...,\ m_N\}$ denotes the set of N on-line news articles. The entire collection of all news articles is called the document or the corpus. A single news article m_i is called an article or a sample.
N	Denotes the number of on-line news articles in the used data set.
X	The matrix containing the numeric vector representation x_i of all N articles, i.e. $X = \begin{bmatrix} - & x_1 & - \\ - & x_2 & - \\ & \vdots & \\ - & x_N & - \end{bmatrix} \in \mathbb{R}^{N\times D}$. Note that it holds that $X \in \{0,1\}^{N\times D}$, because FP is used as the feature extraction weighting method for the sentiment analysis in this master thesis.
Ψ	$\hat{y}_i = \Psi(x_i) : \mathbb{R}^D \mapsto \mathbb{R}$ denotes the prediction function of a trained SVM with as input the numerical feature vector x_i corresponding to article m_i. The function returns the predicted sentiment $\hat{y}_i \in C$ for article m_i.
Θ	$x_i = \Theta(m_i) : \mathbb{R}^{\mathbb{C}} \to \mathbb{R}^D$ is the feature extraction and selection transformation with as input on-line news article $m_i \in \mathbb{R}^{\mathbb{C}}$. The function returns the numeric feature vector representation $x_i \in \mathbb{R}^D$ of article m_i.
χ^2	Abbreviation of χ^2-*statistic* feature selection method (Chapter 3.3.3).
x_i	Denotes the numeric vector representing the feature values of all features $f \in F$ for article m_i. The numeric feature vector of article m_i is given by

	$x_i = \begin{bmatrix} x_{f_1} & x_{f_2} & ... & x_{f_D} \end{bmatrix} \in \mathbb{R}^D$, where x_{f_j} represents the numeric feature value of feature f_j.
y_i	$y_i \in C$ denotes the sentiment classification of article m_i.
CTR	Abbreviation of *click-through rate* of an on-line advertisement or on-line advertising campaign.
DF	Abbreviation of *Document Frequency* feature selection method (Chapter 3.3.1).
FE	Abbreviation of *Feature Extraction* (Chapter 3.2).
FP	Abbreviation of *Feature Presence* feature extraction weighting method (Chapter 3.2.3).
FS	Abbreviation of *Feature Selection* (Chapter 3.3).
IG	Abbreviation of *Information Gain* feature selection method (Chapter 3.3.2).
RP	Abbreviation of *Random Projection* feature selection method (Chapter 3.3.4).
SVM	Abbreviation of *Support Vector Machine* learning algorithm (Chapter 4).

A. Tables and figures

A.1 Binary sentiment analysis

A.1.1 Table: grid search DF

Min. doc. frequency bound	Parameters	Kernel				
		Linear	RBF_γ			
	C		0.0001	0.0005	0.001	0.005
3 (0.05%)	0.1	0.646	0.695	0.689	0.685	0.655
	0.5	0.645	0.695	0.688	0.685	0.675
	1	0.645	0.695	0.691	0.683	0.676
5 (0.1%)	0.1	0.632	0.688	0.689	0.690	0.670
	0.5	0.631	0.688	0.689	0.691	0.678
	1	0.631	0.688	0.686	0.684	0.676
10 (0.2%)	0.1	0.630	0.695	0.693	0.690	0.674
	0.5	0.621	0.695	0.693	0.689	0.685
	1	0.621	0.695	0.688	0.683	0.680
15 (0.3%)	0.1	0.621	0.694	0.693	0.691	0.679
	0.5	0.596	0.694	0.693	0.683	0.677
	1	0.596	0.694	0.685	0.679	0.671
20 (0.4%)	0.1	0.633	0.695	0.694	0.690	0.686
	0.5	0.603	0.695	0.694	0.689	0.678
	1	0.600	0.695	0.687	0.680	0.676
25 (0.5%)	0.1	0.623	0.685	0.685	0.689	0.684
	0.5	0.587	0.685	0.685	0.687	0.682
	1	0.586	0.685	0.685	0.686	0.672
30 (0.6%)	0.1	0.632	0.689	0.691	0.694	0.687
	0.5	0.606	0.689	0.691	0.686	0.679
	1	0.599	0.689	0.683	0.680	0.671

Table A.1: Results of the first part of the grid search, i.e. searching for the best minimum feature document frequency bound. Note that the search was done for different minimum bounds and different SVM parameter combinations.

A.1.2 Table: grid search χ^2

% of features with highest χ^2 score	Parameters	Kernel			
		RBF_γ			
	C	0.0001	0.0005	0.001	0.005
5 %	0.1	0.694	0.695	0.695	0.698
	0.5	0.694	0.695	0.694	0.688
	1	0.694	0.693	0.687	0.695
10 %	0.1	0.697	0.697	0.697	0.698
	0.5	0.697	0.697	0.696	0.700
	1	0.697	0.695	0.694	0.702
20 %	0.1	0.701	0.701	0.700	0.702
	0.5	0.701	0.701	0.699	0.708
	1	0.701	0.698	0.700	0.708
30 %	0.1	0.702	0.702	0.704	0.706
	0.5	0.702	0.702	0.700	0.704
	1	0.702	0.700	0.701	0.709
40 %	0.1	0.700	0.700	0.700	0.705
	0.5	0.700	0.700	0.705	0.710
	1	0.700	0.704	0.707	**0.715**
50 %	0.1	0.704	0.704	0.702	0.701
	0.5	0.704	0.704	0.699	0.700
	1	0.704	0.698	0.697	0.700
70 %	0.1	0.700	0.703	0.700	0.695
	0.5	0.700	0.702	0.699	0.695
	1	0.700	0.696	0.696	0.695
90 %	0.1	0.689	0.689	0.691	0.684
	0.5	0.689	0.689	0.684	0.684
	1	0.689	0.684	0.681	0.678

Table A.2: The results of the second part of the grid search, i.e. searching for the best parameters of different feature selection methods. This table shows the F1 performance scores for different minimum χ^2 score bounds and different SVM parameter combinations.

A.1.3 Table: grid search IG

% of features with highest IG score	Parameters	Kernel			
		RBF_γ			
	C	0.0001	0.0005	0.001	0.005
5 %	0.1	0.691	0.691	0.691	0.689
	0.5	0.691	0.691	0.687	0.683
	1	0.691	0.684	0.680	0.690
10 %	0.1	0.659	0.662	0.664	0.684
	0.5	0.659	0.662	0.677	0.696
	1	0.659	0.679	0.688	0.700
20 %	0.1	0.696	0.697	0.698	0.697
	0.5	0.696	0.697	0.692	0.699
	1	0.696	0.690	0.694	0.703
30 %	0.1	0.696	0.697	0.700	0.708
	0.5	0.696	0.697	0.700	0.706
	1	0.696	0.698	0.702	**0.711**
40 %	0.1	0.694	0.694	0.691	0.702
	0.5	0.694	0.694	0.698	0.699
	1	0.694	0.698	0.698	0.706
50 %	0.1	0.676	0.678	0.683	0.697
	0.5	0.676	0.678	0.695	0.700
	1	0.676	0.692	0.696	0.701
70 %	0.1	0.696	0.699	0.699	0.696
	0.5	0.696	0.699	0.696	0.695
	1	0.696	0.695	0.696	0.696
90 %	0.1	0.687	0.686	0.689	0.684
	0.5	0.687	0.686	0.687	0.684
	1	0.687	0.686	0.688	0.681

Table A.3: The results of the second part of the grid search, i.e. searching for the best parameters of different feature selection methods. This table shows the F1 performance scores for different minimum IG score bounds and different SVM parameter combinations.

A.1.4 Table: grid search random projection

% of original features remaining after random projection	Parameters	Kernel			
		RBF_γ			
	C	0.0001	0.0005	0.001	0.005
10 %	0.1	0.554	0.601	0.630	0.642
	0.5	0.555	0.617	0.626	0.618
	1	0.555	0.623	0.628	0.618
20 %	0.1	0.656	0.664	0.673	0.670
	0.5	0.656	0.668	0.664	0.644
	1	0.656	0.661	0.650	0.636
30 %	0.1	0.673	0.676	0.680	0.673
	0.5	0.673	0.676	0.664	0.660
	1	0.673	0.659	0.658	0.656
50 %	0.1	0.686	0.685	0.683	0.680
	0.5	0.686	0.685	0.677	0.669
	1	0.686	0.676	0.671	0.661
70 %	0.1	0.666	0.671	0.677	0.681
	0.5	0.666	0.672	0.683	0.669
	1	0.666	0.681	0.669	0.669
80 %	0.1	0.679	0.679	0.685	0.677
	0.5	0.679	0.677	0.680	0.663
	1	0.679	0.681	0.670	0.658
90 %	0.1	0.678	0.685	0.690	0.682
	0.5	0.678	0.685	0.679	0.675
	1	0.678	0.679	0.674	0.669

Table A.4: The results of the second part of the grid search, i.e. searching for the best parameters of different feature selection methods. This table shows the F1 performance scores for different number of random projections and different SVM parameter combinations.

A.1.5 Figures: grid search χ^2, IG and RP

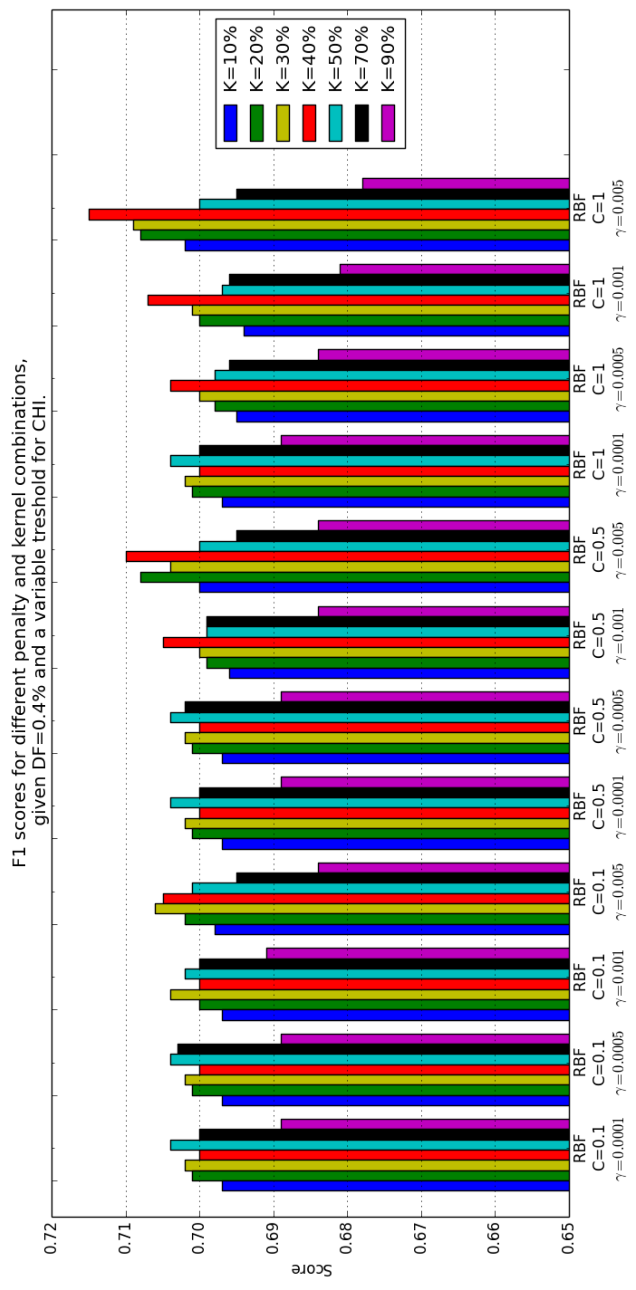


Figure A.1: Results of the grid search for the best χ^2-score bound. The search was performed for various K, i.e. the percentage of features with the highest χ^2-score to select. Note DF=0.4% was used for all calculations.

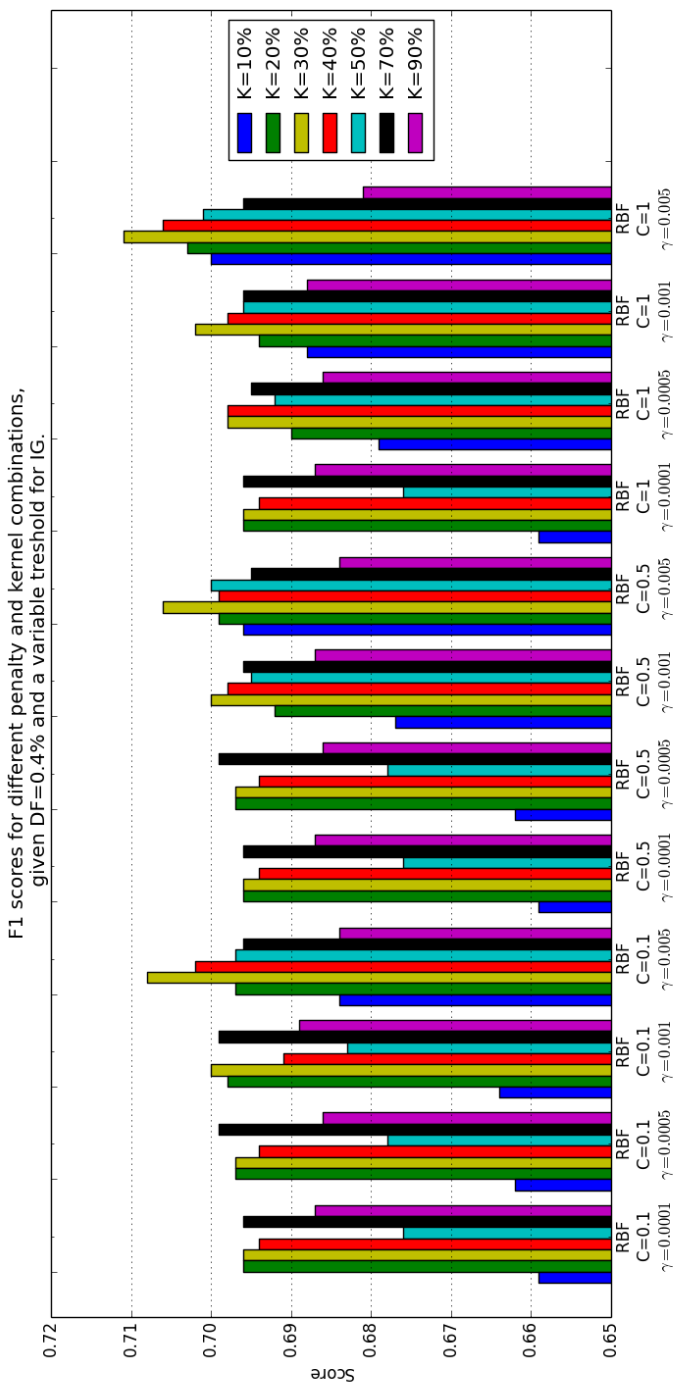


Figure A.2: Results of the grid search for the best IG-score bound. The search was performed for various K, i.e. the percentage of features with the highest IG-score to select. Note DF=0.4% was used for all calculations.

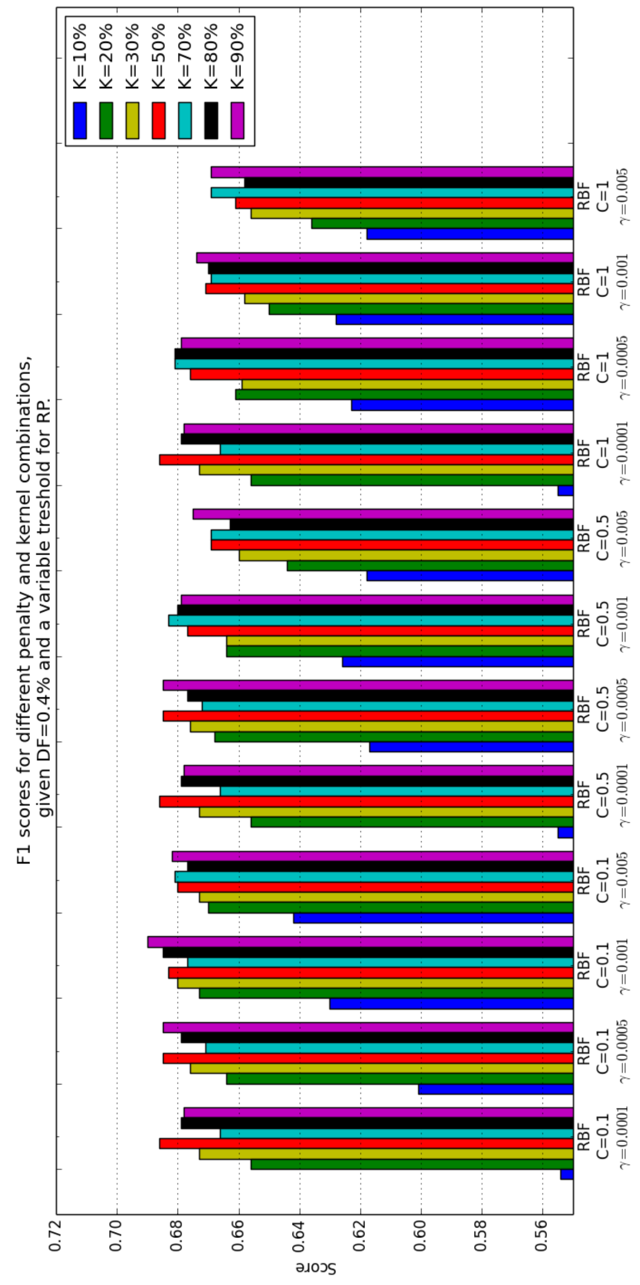


Figure A.3: Results of the grid search for various K, i.e. the percentage of features that remain after random projections are applied. Note that this implies that the number of necessary mappings is given by $(100\% - K) \times 4.012$. Note DF=0.4% was used for all calculations.

A.2 Multi-sentiment analysis

A.2.1 Table: positive vs. rest grid search

Min. doc. frequency bound	Parameters	Kernel		
		RBF_γ		
	C	0.001	0.005	0.01
0.3%	1	0.014	0.402	0.448
	2	0.239	0.491	0.507
0.4%	1	0.010	0.397	0.448
	2	0.218	0.482	**0.508**
0.5%	1	0.009	0.394	0.439
	2	0.204	0.477	0.499
0.6%	1	0.006	0.381	0.428
	2	0.199	0.463	0.496
0.7%	1	0.006	0.385	0.442
	2	0.182	0.475	0.504

Table A.5: The results of the first part of the grid search, i.e. the F1 performance scores for different minimum document feature frequency bounds and different SVM parameter combinations.

% of features with highest χ^2 score	Parameters	Kernel			
		RBF_γ			
	C	0.005	0.01	0.05	0.1
20 %	0.8	0.302	0.410	0.508	0.435
	1	0.351	0.435	0.524	0.474
	2	0.434	0.489	**0.542**	0.507
30 %	0.8	0.335	0.443	0.495	0.304
	1	0.390	0.471	0.517	0.380
	2	0.475	0.514	0.539	0.431
40 %	0.8	0.356	0.451	0.449	0.176
	1	0.396	0.479	0.486	0.260
	2	0.494	0.522	0.517	0.328
50 %	0.8	0.364	0.464	0.390	0.080
	1	0.407	0.487	0.448	0.151
	2	0.500	0.534	0.491	0.227
60 %	0.8	0.369	0.454	0.297	0.026
	1	0.418	0.489	0.379	0.075
	2	0.504	0.531	0.447	0.141

Table A.6: The results of the second part of the grid search, i.e. the F1 performance scores for different minimum χ^2 score bounds and different SVM parameter combinations.

A.2.2 Table: negative vs. rest grid search

Min. doc. frequency bound	Parameters	Kernel		
		RBF_γ		
	C	0.001	0.005	0.01
0.3%	1	0.037	0.376	0.414
	2	0.267	0.459	0.478
0.4%	1	0.038	0.372	0.410
	2	0.253	0.452	**0.496**
0.5%	1	0.027	0.376	0.411
	2	0.249	0.451	0.482
0.6%	1	0.030	0.371	0.416
	2	0.239	0.450	0.480
0.7%	1	0.024	0.376	0.412
	2	0.239	0.440	0.474

Table A.7: The results of the first part of the grid search, i.e. the F1 performance scores for different minimum document feature frequency bounds and different SVM parameter combinations.

% of features with highest χ^2 score	Parameters	Kernel			
		RBF_γ			
	C	0.005	0.01	0.05	0.1
20 %	0.8	0.355	0.427	0.503	0.338
	1	0.388	0.452	0.516	0.425
	2	0.452	0.499	**0.528**	0.485
30 %	0.8	0.370	0.438	0.453	0.195
	1	0.400	0.460	0.492	0.268
	2	0.467	0.504	0.526	0.357
40 %	0.8	0.364	0.435	0.355	0.105
	1	0.398	0.457	0.431	0.163
	2	0.473	0.501	0.488	0.249
50 %	0.8	0.371	0.439	0.293	0.072
	1	0.405	0.466	0.369	0.115
	2	0.479	0.517	0.443	0.173
60 %	0.8	0.369	0.434	0.236	0.034
	1	0.410	0.458	0.309	0.071
	2	0.479	0.511	0.388	0.109

Table A.8: The results of the second part of the grid search. This table shows the F1 performance scores for different minimum χ^2 score bounds and different SVM parameter combinations.

A.2.3 Table: neutral vs. rest grid search

Min. feat. doc. frequency bound	Parameters	Kernel		
		RBF_γ		
	C	0.001	0.005	0.01
0.3%	1	0.032	0.373	0.409
	2	0.262	0.456	0.484
0.4%	1	0.037	0.383	0.419
	2	0.263	0.470	**0.491**
0.5%	1	0.032	0.368	0.412
	2	0.246	0.450	0.475
0.6%	1	0.025	0.375	0.404
	2	0.233	0.446	0.470
0.7%	1	0.025	0.371	0.419
	2	0.230	0.451	0.482

Table A.9: The results of the first part of the grid search. This table shows the F1 performance scores for different minimum document feature frequency bounds and different SVM parameter combinations.

% of features with highest χ^2 score	Parameters	Kernel			
		RBF_γ			
	C	0.005	0.01	0.05	0.1
20 %	0.8	0.358	0.429	0.505	0.342
	1	0.383	0.453	0.520	0.429
	2	0.455	0.492	**0.530**	0.496
30 %	0.8	0.361	0.438	0.456	0.196
	1	0.395	0.459	0.494	0.275
	2	0.464	0.500	0.526	0.356
40 %	0.8	0.376	0.444	0.368	0.108
	1	0.404	0.467	0.437	0.165
	2	0.476	0.517	0.501	0.251
50 %	0.8	0.374	0.443	0.299	0.070
	1	0.412	0.469	0.366	0.113
	2	0.477	0.514	0.444	0.167
60 %	0.8	0.367	0.434	0.242	0.031
	1	0.409	0.467	0.304	0.065
	2	0.478	0.515	0.389	0.114

Table A.10: The results of the second part of the grid search. This table shows the F1 performance scores for different minimum χ^2 score bounds and different SVM parameter combinations.

A.2.4 Table: positive vs. negative grid search

Min. feat. doc. frequency bound	Parameters	Kernel			
		RBF_γ			
	C	0.0005	0.001	0.005	0.01
0.2 %	0.5	0.741	0.750	0.755	0.750
	1	0.750	0.755	**0.761**	0.760
	2	0.756	0.761	0.760	0.758
0.3 %	0.5	0.733	0.741	0.750	0.749
	1	0.744	0.746	0.756	0.757
	2	0.747	0.753	0.754	0.754
0.4 %	0.5	0.737	0.742	0.749	0.747
	1	0.744	0.747	0.754	0.752
	2	0.748	0.749	0.751	0.749
0.5 %	0.5	0.739	0.739	0.754	0.751
	1	0.740	0.747	0.755	0.760
	2	0.748	0.755	0.759	0.760
0.6 %	0.5	0.727	0.737	0.740	0.740
	1	0.736	0.739	0.742	0.740
	2	0.739	0.742	0.738	0.740

Table A.11: The results of the first part of the grid search. This table shows the F1 performance scores for different minimum document feature frequency bounds and different SVM parameter combinations.

% of features with highest χ^2 score	Parameters	Kernel			
		RBF_γ			
	C	0.0005	0.001	0.005	0.01
20 %	0.5	0.747	0.747	0.757	0.757
	1	0.747	0.752	0.758	0.761
	2	0.752	0.755	0.760	0.761
30 %	0.5	0.742	0.745	0.752	0.756
	1	0.744	0.746	0.756	0.759
	2	0.746	0.752	0.759	0.760
40 %	0.5	0.742	0.743	0.756	0.761
	1	0.743	0.750	0.762	0.767
	2	0.749	0.755	**0.767**	0.766
50 %	0.5	0.742	0.743	0.751	0.755
	1	0.740	0.746	0.761	0.760
	2	0.744	0.751	0.759	0.758
60 %	0.5	0.741	0.743	0.758	0.760
	1	0.744	0.747	0.760	0.767
	2	0.747	0.756	0.763	0.765
70 %	0.5	0.740	0.748	0.753	0.756
	1	0.746	0.746	0.758	0.759
	2	0.746	0.751	0.760	0.761

Table A.12: The results of the second part of the grid search. This table shows the F1 performance scores for different minimum χ^2 score bounds and different SVM parameter combinations.

A.2.5 Table: positive vs. neutral grid search

Min. feat. doc. frequency bound	Parameters	Kernel			
		RBF_γ			
	C	0.0005	0.001	0.005	0.01
0.2 %	0.5	0.673	0.662	0.644	0.634
	1	0.657	0.646	0.639	0.635
	2	0.647	0.645	0.633	0.636
0.3 %	0.5	0.674	0.669	0.639	0.641
	1	0.664	0.650	0.642	0.640
	2	0.647	0.639	0.631	0.628
0.4 %	0.5	0.674	0.660	0.636	0.634
	1	0.656	0.650	0.630	0.623
	2	0.647	0.633	0.609	0.605
0.5 %	0.5	0.674	0.663	0.639	0.641
	1	0.661	0.652	0.635	0.635
	2	0.651	0.635	0.626	0.624
0.6 %	0.5	**0.676**	0.662	0.637	0.635
	1	0.659	0.648	0.633	0.633
	2	0.645	0.632	0.628	0.621
0.7 %	0.5	0.674	0.665	0.639	0.639
	1	0.664	0.652	0.634	0.634
	2	0.646	0.636	0.623	0.624

Table A.13: The results of the first part of the grid search. This table shows the F1 performance scores for different minimum document feature frequency bounds and different SVM parameter combinations.

% of features with highest χ^2 score	Parameters	Kernel			
		RBF_γ			
	C	0.0005	0.001	0.005	0.01
20 %	0.5	0.678	0.677	0.669	0.667
	1	0.678	0.674	0.669	0.671
	2	0.673	0.670	0.669	0.670
30 %	0.5	0.677	0.676	0.659	0.666
	1	0.676	0.668	0.667	0.664
	2	0.668	0.659	0.665	0.666
40 %	0.5	**0.680**	0.678	0.658	0.660
	1	0.677	0.663	0.659	0.667
	2	0.663	0.656	0.667	0.663
50 %	0.5	0.679	0.673	0.650	0.654
	1	0.673	0.658	0.655	0.658
	2	0.657	0.650	0.659	0.662
60 %	0.5	0.677	0.670	0.649	0.651
	1	0.670	0.655	0.649	0.654
	2	0.653	0.648	0.649	0.653
70 %	0.5	0.679	0.670	0.655	0.658
	1	0.669	0.657	0.658	0.653
	2	0.656	0.650	0.649	0.650

Table A.14: The results of the second part of the grid search. This table shows the F1 performance scores for different minimum χ^2 score bounds and different SVM parameter combinations.

A.2.6 Table: negative vs. neutral grid search

Min. feat. doc. frequency bound	Parameters	Kernel			
		RBF_γ			
	C	0.0005	0.001	0.005	0.01
0.2 %	0.5	0.682	0.678	0.664	0.649
	1	0.678	0.675	0.653	0.644
	2	0.671	0.664	0.639	0.635
0.3 %	0.5	0.681	0.679	0.662	0.646
	1	0.679	0.667	0.651	0.647
	2	0.666	0.662	0.637	0.628
0.4 %	0.5	0.680	0.681	0.654	0.647
	1	0.679	0.669	0.645	0.642
	2	0.667	0.655	0.634	0.625
0.5 %	0.5	0.682	0.680	0.662	0.654
	1	0.679	0.667	0.652	0.649
	2	0.665	0.658	0.641	0.633
0.6 %	0.5	**0.684**	0.680	0.658	0.646
	1	0.679	0.670	0.644	0.637
	2	0.667	0.653	0.631	0.622
0.7 %	0.5	0.683	0.684	0.658	0.651
	1	0.683	0.669	0.647	0.641
	2	0.669	0.657	0.633	0.629

Table A.15: The results of the first part of the grid search. This table shows the F1 performance scores for different minimum document feature frequency bounds and different SVM parameter combinations.

% of features with highest χ^2 score	Parameters	Kernel			
		RBF_γ			
	C	0.0005	0.001	0.005	0.01
20 %	0.5	0.688	0.690	0.689	0.685
	1	0.691	0.689	0.682	0.684
	2	0.689	0.689	0.684	0.685
30 %	0.5	**0.692**	0.691	0.688	0.688
	1	0.691	0.686	0.686	0.682
	2	0.687	0.688	0.682	0.682
40 %	0.5	0.688	0.689	0.682	0.677
	1	0.689	0.683	0.678	0.674
	2	0.683	0.681	0.675	0.666
50 %	0.5	0.691	0.689	0.679	0.674
	1	0.688	0.687	0.673	0.674
	2	0.685	0.679	0.674	0.668
60 %	0.5	0.687	0.688	0.675	0.675
	1	0.687	0.682	0.675	0.668
	2	0.681	0.678	0.667	0.658
70 %	0.5	0.688	0.687	0.673	0.669
	1	0.686	0.677	0.667	0.664
	2	0.676	0.672	0.658	0.653

Table A.16: The results of the second part of the grid search. This table shows the F1 performance scores for different minimum χ^2 score bounds and different SVM parameter combinations.

A.2.7 Table: polar vs. neutral grid search

Min. feat. doc. frequency bound	Parameters	Kernel			
		RBF_γ			
	C	0.0005	0.001	0.005	0.01
0.2 %	0.5	**0.651**	0.641	0.563	0.543
	1	0.635	0.578	0.575	0.575
	2	0.573	0.570	0.567	0.563
0.3 %	0.5	0.571	0.576	0.539	0.542
	1	0.563	0.538	0.566	0.571
	2	0.530	0.544	0.575	0.575
0.4 %	0.5	0.512	0.521	0.521	0.540
	1	0.510	0.500	0.539	0.546
	2	0.497	0.518	0.542	0.550
0.5 %	0.5	0.486	0.472	0.531	0.542
	1	0.502	0.502	0.555	0.556
	2	0.502	0.535	0.554	0.559
0.6 %	0.5	0.558	0.554	0.543	0.553
	1	0.573	0.546	0.560	0.568
	2	0.543	0.543	0.569	0.573
0.7 %	0.5	0.640	0.627	0.550	0.556
	1	0.626	0.562	0.557	0.562
	2	0.555	0.554	0.554	0.558

Table A.17: The results of the first part of the grid search. This table shows the F1 performance scores for different minimum document feature frequency bounds and different SVM parameter combinations.

% of features with highest χ^2 score	Parameters	Kernel			
		RBF_γ			
	C	0.0005	0.001	0.005	0.01
20 %	0.5	0.656	0.642	0.583	0.626
	1	0.631	0.550	0.639	0.665
	2	0.550	0.580	0.669	**0.690**
30 %	0.5	0.672	0.667	0.603	0.633
	1	0.666	0.602	0.636	0.659
	2	0.601	0.608	0.665	0.681
40 %	0.5	0.670	0.647	0.605	0.631
	1	0.644	0.582	0.639	0.657
	2	0.587	0.608	0.662	0.672
50 %	0.5	0.667	0.661	0.612	0.625
	1	0.657	0.601	0.640	0.654
	2	0.599	0.610	0.664	0.661
60 %	0.5	0.608	0.588	0.598	0.610
	1	0.591	0.565	0.630	0.633
	2	0.561	0.595	0.639	0.639
70 %	0.5	0.666	0.666	0.605	0.606
	1	0.662	0.607	0.613	0.611
	2	0.605	0.606	0.622	0.621

Table A.18: The results of the second part of the grid search. This table shows the F1 performance scores for different minimum χ^2 score bounds and different SVM parameter combinations.

A.2.8 Table: joint one-versus-all grid search

Min. feat. doc. frequency bound	Parameters	Kernel			
		RBF_γ			
	C	0.0005	0.001	0.005	0.01
0.3 %	1	0.495	0.492	0.497	0.485
	2	0.490	0.493	**0.517**	0.504
0.4 %	1	0.494	0.488	0.487	0.482
	2	0.495	0.489	0.510	0.495
0.5 %	1	0.486	0.483	0.484	0.484
	2	0.489	0.492	0.499	0.496
0.6 %	1	0.481	0.471	0.476	0.476
	2	0.479	0.480	0.500	0.498
0.7 %	1	0.479	0.479	0.473	0.476
	2	0.480	0.480	0.498	0.491

Table A.19: The results of the first part of the grid search for the joint one-versus-all heuristic. This table shows the F1 performance scores for different minimum document feature frequency bounds and different SVM parameter combinations.

% of features with highest χ^2 score	Parameters	Kernel			
		RBF_γ			
	C	0.0005	0.001	0.005	0.01
20 %	1	0.479	0.476	0.471	0.479
	1	0.480	0.475	0.480	0.486
	2	0.477	0.474	0.501	0.511
30 %	0.8	0.478	0.475	0.476	0.480
	1	0.482	0.474	0.485	0.496
	2	0.483	0.481	0.515	0.502
40 %	0.8	0.484	0.481	0.484	0.489
	1	0.485	0.481	0.489	0.495
	2	0.481	0.488	0.511	0.499
50 %	0.8	0.489	0.485	0.487	0.479
	1	0.487	0.483	0.489	0.497
	2	0.486	0.491	**0.517**	0.506
60 %	0.8	0.487	0.489	0.483	0.479
	1	0.492	0.483	0.490	0.495
	2	0.491	0.494	0.512	0.506

Table A.20: The results of the second part of the grid search for the joint one-versus-all heuristic. This table shows the F1 performance scores for different minimum χ^2 score bounds and different SVM parameter combinations.

A.2.9 Table: joint one-versus-one grid search

Min. feat. doc. frequency bound	Parameters	Kernel			
		RBF_γ			
	C	0.0005	0.001	0.005	0.01
0.3 %	1	0.466	0.504	0.525	0.525
	2	0.507	0.517	0.516	0.515
0.4 %	1	0.470	0.505	0.524	**0.527**
	2	0.507	0.517	0.519	0.516
0.5 %	1	0.468	0.499	0.517	0.520
	2	0.501	0.511	0.512	0.507
0.6 %	1	0.466	0.498	0.510	0.517
	2	0.502	0.511	0.510	0.505
0.7 %	1	0.466	0.498	0.516	0.516
	2	0.500	0.511	0.510	0.510

Table A.21: The results of the first part of the grid search for the joint one-versus-all heuristic. This table shows the F1 performance scores for different minimum document feature frequency bounds and different SVM parameter combinations.

% of features with highest χ^2 score	Parameters	Kernel			
		RBF_γ			
	C	0.0005	0.001	0.005	0.01
20 %	0.8	0.436	0.489	0.537	0.542
	1	0.456	0.502	0.539	0.548
	2	0.503	0.527	0.548	0.551
30 %	0.8	0.448	0.498	0.544	0.553
	1	0.479	0.511	0.549	0.555
	2	0.513	0.537	0.554	0.557
40 %	0.8	0.451	0.501	0.552	0.555
	1	0.479	0.513	0.553	0.558
	2	0.516	0.540	**0.561**	0.556
50 %	0.8	0.454	0.505	0.548	0.551
	1	0.480	0.516	0.551	0.552
	2	0.518	0.539	0.555	0.557
60 %	0.8	0.461	0.509	0.547	0.553
	1	0.487	0.522	0.549	0.551
	2	0.527	0.541	0.554	0.553

Table A.22: The results of the second part of the grid search for the joint one-versus-all heuristic. This table shows the F1 performance scores for different minimum χ^2 score bounds and different SVM parameter combinations.

A.2.10 Table: joint polar-neutral hierarchical SVM grid search

Min. feat. doc. frequency bound	Parameters	Kernel			
		RBF_γ			
	C	0.0005	0.001	0.005	0.01
0.3 %	1	0.389	0.393	0.410	0.402
	2	0.394	0.414	**0.484**	0.472
0.4 %	1	0.389	0.392	0.408	0.401
	2	0.392	0.407	0.474	0.467
0.5 %	1	0.386	0.391	0.409	0.403
	2	0.393	0.405	0.466	0.460
0.6 %	1	0.387	0.392	0.402	0.401
	2	0.392	0.404	0.456	0.458
0.7 %	1	0.386	0.387	0.406	0.396
	2	0.391	0.400	0.451	0.462

Table A.23: The results of the first part of the grid search for the joint polar-neutral hierarchical heuristic. This table shows the F1 performance scores for different minimum document feature frequency bounds and different SVM parameter combinations.

% of features with highest χ^2 score	Parameters	Kernel			
		RBF_γ			
	C	0.0005	0.001	0.005	0.01
20 %	0.8	0.380	0.385	0.420	0.446
	1	0.384	0.388	0.433	0.458
	2	0.386	0.397	0.475	0.485
30 %	0.8	0.381	0.386	0.420	0.448
	1	0.385	0.389	0.436	0.454
	2	0.388	0.402	0.474	0.487
40 %	0.8	0.384	0.388	0.420	0.432
	1	0.384	0.390	0.431	0.451
	2	0.392	0.406	0.481	0.485
50 %	0.8	0.385	0.390	0.413	0.409
	1	0.385	0.391	0.430	0.438
	2	0.391	0.410	0.482	0.491
60 %	0.8	0.380	0.389	0.407	0.407
	1	0.386	0.391	0.427	0.430
	2	0.393	0.409	0.480	**0.494**

Table A.24: The results of the second part of the grid search for the joint polar-neutral hierarchical heuristic. This table shows the F1 performance scores for different minimum χ^2 score bounds and different SVM parameter combinations.

A.2.11 Table: joint positive-negative hierarchical SVM grid search

Min. feat. doc. frequency bound	Parameters	Kernel			
		RBF_γ			
	C	0.0005	0.001	0.005	0.01
0.3 %	1	0.494	0.512	0.514	0.501
	2	0.515	**0.523**	0.510	0.499
0.4 %	1	0.485	0.506	0.514	0.499
	2	0.514	0.518	0.512	0.502
0.5 %	1	0.492	0.510	0.508	0.500
	2	0.508	0.512	0.511	0.497
0.6 %	1	0.491	0.505	0.505	0.496
	2	0.510	0.513	0.509	0.492
0.7 %	1	0.487	0.512	0.508	0.490
	2	0.502	0.508	0.501	0.493

Table A.25: The results of the first part of the grid search for the joint positive-negative hierarchical heuristic. This table shows the F1 performance scores for different minimum document feature frequency bounds and different SVM parameter combinations.

% of features with highest χ^2 score	Parameters	Kernel			
		RBF_γ			
	C	0.0005	0.001	0.005	0.01
20 %	0.8	0.463	0.495	0.513	0.508
	1	0.480	0.499	0.515	0.513
	2	0.496	0.518	0.519	0.506
30 %	0.8	0.478	0.499	0.520	0.505
	1	0.485	0.504	0.519	0.508
	2	0.503	0.516	0.519	0.508
40 %	0.8	0.479	0.496	0.509	0.502
	1	0.489	0.508	0.515	0.511
	2	0.509	0.519	0.515	0.508
50 %	0.8	0.485	0.505	0.519	0.503
	1	0.491	0.509	0.512	0.504
	2	0.510	0.524	0.518	0.507
60 %	0.8	0.484	0.506	0.511	0.501
	1	0.494	0.508	0.514	0.501
	2	0.513	**0.523**	0.517	0.505

Table A.26: The results of the second part of the grid search for the joint positive-negative hierarchical SVM heuristic. This table shows the F1 performance scores for different minimum χ^2 score bounds and different SVM parameter combinations.