Master Thesis

Data Mining for Tweet Sentiment Classification

Author:
R. de Groot
`r.degroot4@students.uu.nl`

Internal Supervisors:
dr. A.J. Feelders
prof. dr. A.P.J.M. Siebes

External Supervisors:
E. Drenthen
G. Ramaker

July 23, 2012

ICA-3238679

Utrecht University
Faculty of Science
Department of Information and Computing Sciences

Abstract

The goal of this master thesis is to classify short Twitter messages with respect to their sentiment using data mining techniques. Twitter messages, or tweets, are limited to 140 characters. This limitation makes it more difficult for people to express their sentiment and as a consequence, the classification of the sentiment will be more difficult as well. The sentiment can refer to two different types: emotions and opinions. This research is solely focused on the sentiment of opinions. These opinions can be divided into three classes: positive, neutral and negative. The tweets are then classified with an algorithm to one of those three classes.

Known supervised learning algorithms as support vector machines and naive Bayes are used to create a prediction model. Before the prediction model can be created, the data has to be pre-processed from text to a fixed-length feature vector. The features consist of sentiment-words and frequently occurring words that are predictive for the sentiment. The learned model is then applied to a test set to validate the model.

The data that is considered in this research is based on two datasets, one from Sananalytics and a self-built Twitter dataset. When the models are applied and tested in-sample the results were quite acceptable. However out-of-sample, with cross-validation the results were disappointing. The sparsity in the dataset seems to cause the issue that the features in the training set does not cover the data in the test set well.

Keywords: sentiment analysis, data analysis, text mining, classification, big data, social media, twitter, complex event processing

Acknowledgment

Foremost, I would like to sincerely thank my supervisors, who continuously supported me during my research. In the first place, my internal supervisor Ad Feelders, for the productive sessions we had. Those sessions lead to new insights, which I could directly apply in my research. During the writing of my thesis I appreciated the scientific feedback of Ad Feelders, especially when it was difficult to put things on paper. I am very grateful that he spend a lot of time in reviewing my thesis.

I also want to thank my external supervisors Edo Drenthen and Gijs Ramaker. The weekly meeting with Edo Drenthen helped me to get on track, especially at the start. The feedback of Edo Drenthen on all delivered products was very insightful and useful. Also thanks to Gijs Ramaker for the more general guidance and feedback on my research.

Last but not least, I want to thank Avanade for allowing me to do my graduation project at their company. It was a period where I learned a lot, and get involved with the practical application in business perspective. And thanks to the Avanadi, the people of Avanade, who helped me with several issues and made me feel comfortable during my project.

Contents

Chapter 1

Introduction

In the last couple of years the social medium Twitter has become more and more popular. Since Twitter is the most used microblogging website with about 500 million users and 340 million tweets a day, it is an interesting source of information. The messages, or in Twitter terms the tweets, are a way to share interests publicly or among a defined group. Twitter distinguishes itself from other social media by the limited message size. The maximum size of 140 characters restricts users in their writing. Twitter is therefore challenging their users to express their view in one or two key sentences.

Because Twitter is widely adopted through all strata, it can be seen as a good reflection of what is happening around the world. Among all that happens, the latest trends are most interesting for companies. The latest trends can be analyzed and when identified, reacted to. From a marketing point of view, these latest trends can be used to respond with appropriate activities, like product advertisements. Analyzing tweets can therefore be a goldmine for companies to create an advantage to competitors.

One interesting group are tweets expressing sentiments about products, brands or services. These messages contain an opinion about a specific subject. The sentiment of this opinion can be classified in different categories. An obvious example of three categories are the categories positive, neutral and negative. These categories have been studied a lot in the literature. The categories can even be split further into a model with more sentiment levels. The whole process of identifying and extracting subjective information from raw data is known as sentiment analysis. The sentiment analysis research field is closely related to the field of natural language processing. Natural language processing tries to close the gap between human and machine, by extracting useful information from natural language messages. In this research, the extraction of the sentiment from a tweet is studied.

In data analysis, algorithms have been developed which can be used to analyze data, with the goal to extract useful information. Some widely used classification algorithms from the literature, such as naive Bayes and support vector machines, will be applied and compared. In this research, these algorithms are used to assign a sentiment (positive, neutral or negative) to a tweet. The number of tweets that have to be processed is too high and complex for normal data adapters, therefore a complex event processing engine is required for processing the data. Complex event processing deals with processing many events that happen in real-time. Complex event processing platforms allow one to monitor data from multiple sources for meaningful patterns, trends and exceptions. It could be used to perform low-latency analytics on the events and to trigger response actions. CEP caters to the intended functionality of the target system by classifying tweets in near real-time.

This research will focus on the application of sentiment analysis to Twitter and comparing the performance of different classification algorithms on this problem. The main question of this thesis is: "How can Twitter messages be accurately classified with respect to their sentiment?". To answer this question, a sentiment analysis tool is implemented, which provides a framework for testing the quality of the algorithms. This tool provide away to query the sentiment about popular mobile phones and their brands. Mobile phones are an interesting topic to focus on, because predominantly young audience catalyzing the probability for Twitter usage. Also the fact that mobile phones are fast changing products, boosts the amount of tweets about it.

1.1 Context

This master thesis is part of the study computing science, a 2-year research master of the university of Utrecht. The master thesis is the final project leading to graduation. Writing a thesis can be done internally at the university or externally at a company. In the last period of my academical education I realized that I missed some real-life experience working within a professional organization. Therefore I have chosen to conduct my research and thesis delivery at Avanade. Avanade offered the opportunity to get involved with a company and with new technologies. Together with Avanade, I composed a challenging graduation project about the "trending topic" Twitter. This internship project connects my focus area of my master, "data analysis" with a scientific challenging assignment.

Avanade is a multinational IT consulting and system integrator company, which develops Microsoft business and enterprise software. Because Avanade focuses solely on Microsoft-enabled solutions, the project will largely consist of Microsoft tooling. Microsoft will be used where there is value-add. For scientific calculations the tool will connect with software specifically made for data analysis and statistics.

The target audience of this thesis consists of fellow master students, recently graduated students and others who are interested in Twitter sentiment analysis. In writing this thesis I assumed that the reader has a basic understanding of common concepts in computer science. Therefore some basic technical knowledge about data analysis is a prerequisite.

1.2 Structure of the Thesis

This thesis is structured as follows. In chapter 2 the subject of the research and theory is explained, including challenges currently found in analyzing natural language messages. From this problem a main question and several sub questions are distilled. Those questions form the guidance through this thesis. Furthermore the scope of the project is led from the research questions and challenges. Chapter 3 provides the background of the different related fields. The main topics that are discussed are: sentiment analysis, Twitter and complex event processing. Subsequently, chapter 4 discusses the algorithms and the relevant literature. To answer the research questions a number of experiments have been composed, which are explained in chapter 5. The set-up, the execution and the results of these experiments are all covered in this chapter. The integrated system of the Twitter sentiment analysis tool is covered in chapter 6. Finally, chapter 7 draws the conclusions of the project. It will refer back to the proposed research questions in chapter 2 and answer them. Furthermore, possibilities of follow up research are discussed.

Chapter 2

Problem Statement

This chapter states the problem and associated questions of this research. Furthermore, the scope of this research is defined.

2.1 Research Questions

Main question: *How can Twitter messages be automatically and accurately classified with respect to their sentiment?*

In this project the main goal is to accurately classify tweets with respect to their sentiment. This is realized by developing a tool which can classify the tweets.

Sub question 1: *Which supervised or unsupervised learning algorithm achieves the best performance for sentiment analysis on Twitter?*

The data analysis research area covers the algorithms for the Twitter sentiment analysis tool that has to be developed. Related research fields are; artificial intelligence, data mining, machine learning, natural language processing and sentiment analysis. In data analysis different types of algorithms have been developed. Those algorithms can be grouped into supervised and unsupervised learning algorithms. The difference between these two models is that in supervised learning a model is created on a training set and in unsupervised learning a predefined model is assumed. The question is whether the one of the two performs better and achieves a higher classification accuracy.

Sub question 2: *Does Twitter's additional user information improve the classification accuracy?*

In the data analysis, the performance of a classification algorithm is often domain specific, and therefore depends on the domain where it is applied to. In general, different classification problems can have different superior algorithms. Also in case of one domain, a different subset can have a different superior algorithm. Therefore an algorithm which is the overall best, does not exist. Restricting the scope possibly leads to a better accuracy of an algorithm. This could be realized by narrowing the scope down to a smaller domain, by restricting the content to some topic. In addition it is interesting to have more information about the users to possibly segment the users into user groups.

Information such as the age, sex, location and background adds an extra dimension to the results. Most market surveys contain questions about the age, sex and location of the interviewees, this information is then used in the report and conclusions of the market research. Similarly, this information can be used for the reporting of the sentiment. The additional information, can make the sentiment more valuable for companies.

Sub question 3: *Is it possible to handle negation, sarcasm, irony and double negatives to improve the performance of the classifier?*

In natural language processing a difficult task is to cope with sentences with different meanings. In speaking, a tone is added from which you can derive the meaning better. As plain text does not contain pronunciation, tone and expressions by voice, it is more difficult to get the intended sentiment. If sarcasm, irony and double negatives could be parsed, the model will be more accurate. However the model could also be overfitted and classify many tweets as false positives. In such cases handling those figures of speech may not be beneficial.

Sub question 4: *Can the special features of Twitter's orthography, like hash-tags, retweets and emoticons be used to improve the classification of the messages?*

Twitter provides an orthography with some special features. These features such as hash-tags, retweets and emoticons could help in classifying the tweets. The hash-tags could help to group messages and some could even help to indicate a certain sentiment.

Sub question 5: *How many levels are optimal to represent a sentiment realistically and is still interpretable?*

Classifying the sentiment is often done in models with three levels: positive, negative and neutral. A model with more levels could improve the informativeness of the representation. However it is more difficult to label the data in more levels and the question is whether the algorithm can still achieve a good classification accuracy.

2.2 Scope Definition

Twitter is a massive medium where people post millions of tweets a day. Capturing tweets requires specialized tools, capable of handling a large stream of data. Creating a classification model on this stream of data requires a well specified domain to achieve a good performance. Therefore I restricted the scope to English tweets only. The English language is the original Twitter language and used most often, according to a study by the Semiocast [36]. In October 2011, 39% of the 180 million messages a day were in English. Besides the fact that English is the largest language on Twitter, it is one of the most spoken languages around the world. Introducing more languages makes the research more complex. Since the essential part of the research could be sufficiently studied with only one language, only the English language is considered in my research.

The domain is already restricted through the language choice. Still, the scope remains very broad, with all the English tweets. According to general ideas in data analysis literature, restricting the domain to a specific subject will possibly lead to better performance. I have chose to restrict the domain to mobile phones and their related IT brands. Some examples of such products are HTC Trophy, Nokia Lumia 800, Samsung Galaxy S2 and iPhone. Examples of their related brands are HTC, Nokia, Samsung and Apple. I also added some interesting brands for Avanade, such as Microsoft and Avanade. The main reason to choose the mobile phones domain is the interrelation with Twitter. The smart phones provide access to Twitter all day long. Research [29] showed that 43% of the users access Twitter by their mobile phone. Therefore the mobile phones are an interesting subject since a lot of the users own them and have opinions about it. Therefore tweets about bugs, defects or cool features are posted regularly. This leads to a large amount of sentiment data to apply sentiment analysis.

Chapter 3

Background Information

In this chapter the general background of the different related technologies that are involved is discussed, namely sentiment analysis, Twitter and complex event processing.

3.1 Sentiment Analysis

3.1.1 General

Sentiment analysis involves in several research fields; natural language processing, computational linguistics and text analysis. It refers to the extraction of subjective information from raw data, often in text form. However, also other media types could contain subjective data, like images, sounds and videos but these types are less studied. In accordance, in all media types different kinds of sentiments exist. The sentiment can refer to opinions or emotions, however these two types are related there is an evident difference. In sentiment analysis based on opinions, a distinction is made between positive and negative opinions. On the other hand, sentiment analysis based on emotions, is about the distinction between different kinds of emotions, like angry, happy, sad etcetera. The sentiment analysis that is considered in this research is based on opinions and is often referred in literature as opinion mining. When just sentiment analysis is stated in the remainder of this thesis, sentiment analysis based on opinions is meant. Sentiment analysis aims to determine the attitude of the opinion holder with respect to a subject. Other applications try to determine the overall sentiment of a document. Sentiment analysis can be difficult, hence Liu speaks of "the problem of sentiment analysis" in [24]. For example, a text can contain more than one opinion about the same object or about several objects. Consider the following example (similar to Liu's example in [24]):

Example 3.1. *"I recently bought a Nokia Lumia 800 from Amazon.com. When I unpacked the phone, I realized what kind of terrific phone I bought. The Windows Phone OS is very easy-to-use. However as with most smartphones, the battery life is dramatic. Yesterday, I showed the phone to my colleague, and he directly became enthusiastic. He said that the Nokia Lumia 800 is definitely better than his iPhone 4."*

In example 3.1, two types of opinions are stated, a direct opinion and an indirect opinion. A direct opinion is a straight-forward review about (a part of) an object; "the battery life is dramatic". An indirect opinion reviews an object with respect to another object: "the Nokia Lumia 800 is definitely better than his iPhone 4.". Direct opinions can in most cases unilateral parsed in positive/ negative and neutral, except for some nontrivial cases where even humans can't classify unambiguously. In indirect opinions the opinion can be parsed differently for the involved objects. A similarity or a difference between the two objects can be described with an indirect opinion. An object a can be equally good as the other object b and in another case a can be better than the other object b. In the former case the opinion is positive for both products; a similarity, in the latter case it is positive for a and negative for b; a difference. The opinions in those sentences could also be the reversed. This leads to four different indirect comparative opinions, which should be deduced from the message. Furthermore, an indirect opinions can be formalized with many different words, which express essentially the same opinion. For example the sentence "a is performing better than b" expresses almost the same opinion as the sentence "b does crash more often than a". Both sentences lead to a positive sentiment for a and negative for b, but are expressed in a totally different way. The deduction of an indirect comparative opinion is therefore hard. Returning to the direct opinion, a direct opinion is characterized by five properties that are formalized in a quintuple by Liu [24]. The quintuple is stated below in equation 3.1.

$$opinion = (o_j, f_{jk}, oo_{ijkl}, h_i, t_l) \tag{3.1}$$

where o_j is a particular object; f_{jk} is feature k of object o_j; h_i is an opinion holder; t_l is the time and the oo_{ijkl} the actual opinion of holder h_i about feature f_{jk} of object o_j at time t_l.

Determining the actual polarity or oo_{ijkl} of some sentence is the most difficult task of the five properties of the quintuple. The polarity of a message is whether it is positive or negative, in analogy to the electrical polarity. This sentiment is subjective because different people have different mental scales for what they consider to be a strong or a weak opinion. Therefore it can occur that a sentence is labeled as positive to somebody and as neutral by somebody else.

As discussed in the section 2.1, most common models capture opinions into two or three levels: positive, negative and additionally neutral. However it is also possible to use a more granular scale, as Wilson in [42].

In the handbook "Opinion Mining and Sentiment Analysis" from Pang et al. [32], more additional information about sentiment analysis can be found. It is a great reference as general background for sentiment analysis.

3.1.2 Business case

Why would you apply a sentiment analysis, what does it yield? As Peter Yared wrote in [43]: "Sentiment analysis isn't a solution unto itself, but it can be highly useful as a real-time feedback loop for advertising effectiveness and may soon be able to predict advertising results". It has always been difficult to track the return on investment of marketing expenditures. Marketers focus on measuring the increase of brand, color schemes or slogan awareness. A part of those marketing measures, e.g. the brand name publicity could be measured over social media. In such cases, sentiment analysis offers a less labor intensive way and therefore cost-avoidance way to measure the effectiveness of marketing activities.

3.2 Twitter

Twitter, a microblogging website which is nowadays familiar to most people, has made a great development in popularity and usage in the last couple of years. Firstly, I will discuss a historical background about the foundation of Twitter. Secondly, I will discuss the functional environment of Twitter.

3.2.1 History

In contrast to the popularity of Twitter at the moment, it begun as a backup project for a failed project. Noah Glass and Google employee Evan Williams founded a new company Odeo, which focused on a podcasting platform. Evan Williams asked a former colleague Biz Stone to join the company. Odeo started developing their podcasting platform until Apple launched the podcasting product iTunes. In consequence the podcasting platform from Odeo became irrelevant and less profitable. Together with the new hired web designer Jack Dorsey, Odeo started with some brainstorm sessions to come up with new projects. Jack Dorsey, had an idea around what people are doing at a given time. His idea lead to an SMS service for communicating to a small group of people. Noah Glass pushed the project and was actually seen as the ”spiritual leader” of this project.

In March 2006 the first prototype with the name ”twttr” was delivered, and Noah Glass was still very enthusiastic about it. The use of the SMS service limited the number of characters that could be used; an SMS can contain up to 160 characters [27]. Since the username already requires 20 characters of space, the length of a message is limited to 140 characters.

In 2007 the Twitter branch spun off as a separate company from Odeo [28]. A few months later Evan Williams decided to fire one of his most passionate employees Noah Glass, probably because the two had clashing personalities [9]. In the spring of 2007, Twitter really begun to grow after the SXSW Interactive conference in Austin, Texas. Even in 2008, it actually begun to grow exponentially. In parallel a company called Summize created a simple way to search and find sentiments about tweets. Twitter acquired Summize quite fast after they knew it could be beneficial. Nowadays this service is known as the searching functionality of Twitter. A former Google employee Dick Costolo, entered the Twitter company as COO in September 2009. He even became CEO when Evan Williams decided to step down as Twitter's CEO.

Nowadays Twitter is still growing, in 2012 to 500 million active users worldwide and 340 million tweets a day. Twitter is *the* individual mass medium to find out what is happening nowadays. It is even labeled as the fastest medium, since news on Twitter is a few minutes earlier than in other media.

3.2.2 Functionality

Twitter is used in many different ways, for both personal and business use. For personal use it is a great way to keep in touch with your friends and quickly broadcast information about where you are and what you are up to. More interesting is the business use, since personal use is not beneficial. For business, Twitter can be used to broadcast a company's latest news and blog posts or interact with customers. Through the great accessibility both the personal and business are involved together at Twitter. This aspect is valuable for companies since they can reach their customers and potential customers in an informal way. Twitter incorporates this all in its slogan: "Follow your interests, instant updates from your friends, industry experts, favorite celebrities, and what's happening around the world" [19].

Besides the direct business use of Twitter also an indirect use is interesting for business. For instance to search and aggregate personal messages about companies, brands or products. Twitter is accessible by the Internet for anyone. Therefore it is the fastest mass communication channel, even faster then renowned news websites. The real-time updates can be used by companies for different beneficial purposes. Even some books, like [12], are published about how Twitter could be beneficial for your company.

From all the information on Twitter, only a small fraction is interesting for a company. The information that is contained in the messages depends on the user's personal interest. The message could therefore have objective content like facts, and subjective content like opinions. This latter type of content is most interesting for companies if the opinion is about a related object. Such information could be aggregated and form an asset for the company.

The information is contained in the messages (tweets), which have a maximum length of 140 characters. This limited number causes creative people to use acronyms and abbreviations to enlarge the expressibility of their message. Those acronyms lead to a broader dictionary of words, but also make it harder to analyze the tweets, since they create a broader feature space.

Twitter offers a special orthography that includes special features, hash-tags, user mentions and retweets. The "#"-hash-tags are integrated in Twitter, and are used to categorize tweets. In basic usage this categorization is based on subject and topics. But these hash-tags can also be used to add an opposite direction of the tweet, like sarcasm or irony. Such tags can reverse the polarity of the message. The most used hash-tags at the moment are summarized in the trending topics.

With a "@" symbol tweets can be directed to another user. Normally, the tweets are posted in public, or to a restricted group. The prefix "@" with a username directs the message to a specific user. The other user is aware of this directed message, and can respond to it. Thus, conversations can arise by mentioning other user in tweets.

Another Twitter term is the retweet (RT), which is used to show the content of a tweet posted by another user. Users post retweets to note that the original message is interesting enough to send to their followers. An interesting question is whether you should include retweets into your sentiment analysis, since it is actually a repetition of a tweet.

In the line of this research, emoticons are interesting, because they state the mood of a user. This mood is in some cases related and relevant for the sentiment of the message. Smiling and sad emoticons give a good indication of the sentiment, however other emoticons like confused or embarrassed are less informative. Therefore only a part of the emoticons could be useful for sentiment classification.

The most difficult aspect is the overall freedom, because Twitter does not have a protocol about how to use it. This includes spelling mistakes, domain specific content and acronyms. Summarized, the Twitter data lacks a well-defined structure. It is a great challenge to create an applications which use Twitter data and accurately classifies the sentiment.

3.3 Complex Event Processing

Complex event processing, or shortly CEP, is an emerging network technology developed by the need for near real-time analysis of data. With CEP large amounts of event data streams from multiple sources can be effectively analyzed. It acts, monitors and analyzes data in motion and makes decisions in near real-time. While typical relational database systems are query-driven, CEP provides an event-driven approach. This approach is characterized by high event data rates, continuous queries, and millisecond latency. Those properties make it impractical to use normal relational databases.

Event-driven applications use CEP, a technology that processes many events with the goal to identify meaningful patterns, relationships and data abstractions. Traditionally queries are processed against the data to get a finite result. Contiguous queries are standing queries where data is fired on in an infinite stream of data, it is often explained by "throwing data against a query instead of throwing a query against data". The contiguous query processes data in real-time. This technique looks promising with the upcoming focus area big data, and is fueled by large enterprises generating more and more data [26]. The need of event-driven applications comes from lots of industries such as: financial services, health care, IT monitoring, manufacturing, oil and gas, transportation, utilities, and web analytics to act in real-time on data.

A possible example of the application of CEP can be illustrated with an oil and gas company. For a big oil and gas company, it is very important not to waste oil, gas or materials during the harvesting. Obviously wasting is always bad, it would be beneficial to prevent it. During the harvesting the platform uses a lot of hard- and software to extract the oil. In the process of the extraction of oil awareness over infrequencies can indicate potential problems. Monitoring the machines in real-time can help to detect problems in an early stage. CEP can help with monitoring the huge amount of data from the machines in real-time. Infrequency can be detected through a predefined standing query. Thereon a precaution could be taken to prevent future problems and saving the environment, materials and money.

The advantages of complex event processing can also work out for a sentiment analysis application. With the CEP technique the messages could be retrieved in a continuous stream almost directly after they are posted. Subsequently the tweets can be processed immediately after they are retrieved. The high processing throughput makes a technique as CEP excellent to work with the data stream from Twitter.

Chapter 4

Data Mining for Sentiment Classification

In the previous chapter I discussed the different subfields that are involved in this project: sentiment analysis, Twitter and complex event processing. This chapter focuses on the processes and techniques of my research. The literature of the relevant algorithms to this problem domain is discussed in this chapter.

Sentiment analysis was first applied to online reviews. Later on when Twitter became popular, Jansen et al. [20] realized that microblogging was a potentially rich avenue for companies to explore their overall branding strategy. From then on, lots of research was based on Twitter. In this chapter pre-processing of the data, the feature selection and different classification algorithms are discussed.

4.1 Data Pre-processing

Pre-processing in general, is extensively studied and is used in a lot of applications that consider raw, unstructured data. When research became more focused on Twitter, the need of pre-processing increased proportionally, because many tweets are not properly formatted, or contain spelling errors. As Thelwall et al. concluded earlier in [38]: "Text based communication in English seems to frequently ignore the rules of grammar and spelling". Therefore pre-processing techniques are necessary, to acquire a more clean dataset. Cleansing the dataset increases the performance of the later classification system significantly.

In nearly all literature about Twitter sentiment analysis pre-processing techniques are incorporated [2, 14, 16, 21, 30, 31]. The applied techniques are quite straightforward, like filtering words, letters, punctuations and correcting simple errors.

Correcting the simple type errors, like misspells and repeated letters is based on dictionaries. The dictionaries are used to correct the errors. Similarly abbreviations and acronyms are replaced with words from a predefined dictionary. Another strategy which has been applied is removing useless content. Since stop words and punctuations don't have much influence on the sentiment, these are removed to decrease the diversity of used words in messages. Stop words are by definition extremely common words, which are removed from text in natural language processing in advance. Meanwhile, there is no definite list of stop words. All the dictionaries have therefore to be selected manually from different resources.

An example filtering technique is filtering the overuse of characters. A repetition of vowels, like 'coooooooool' is an example. An example of repetition of punctuations is 'cool!!!!!!!'. Those filtering techniques can be realized by recognizing an overuse of more than 2 subsequent characters.

More Twitter specific pre-processing techniques, based on Twitter's orthography can be used to filter out the special features (hash-tags, user mentions and retweets). Almost all papers that are focused on Twitter filter these kinds of Twitter orthography. With removing usernames, hyperlinks, urls and repeated letters, Go et al. [16] reduced the number of the features to about 46% of the original. The usernames had the largest impact on the reduction, which is plausible. The diversity of usernames causes a lot of new words, therefore leaving out the usernames results in a great reduction of the data.

The most common technique to replace and filter strings is a regular expression [39]. It provides a way to filter out errors and overuse, and more generally it provides a decent way to recognize substrings in strings. Regular expressions are a very powerful string matching technique which is used in many "search-and-replace" functions of applications. They are written in a formal language, which is compact but have the power to match with almost all string patterns.

Another useful technique, called stemming can be applied to reduce the large diversity of words. Stemming is a technique to reduce the conjugations of verbs to their stem, the original root form. An example of stemming is the reduction of "liked" to "lik" and "like" to "lik". This reduces the diversity of conjugations in which a word can occur, and as consequence reduces the amount of data.

Most of those described techniques are language specific. The language of a text plays an important role in the pre-processing but also in the later classification. As mentioned earlier it is important to acquire a dataset that is as clean as possible. If several languages are considered, they each produce a different structure, grammar and dictionary of words. Therefore determining the language is necessary to be able to select specific language content. This is the main research area of language identification and has been studied extensively. Papers about sentiment analysis focus solely on one language, most common is the English language, and to a lesser extent the Chinese language. Acquiring a single language dataset, requires filtering or translation. Both techniques need to classify the language of the text. Identification of languages can be performed by learning the distribution of characters per language. In the literature this distribution measure has proven to be effective and simple. The frequency of the letters and subsequent letters define the language. Different implementations use this principle, like TextCat [10] and LingPipe [4].

TextCat [10] is a text identification method, which has been applied to identify the language of newsgroup articles. They achieved outstanding results with an n-gram based approach. Their n-gram approach is based on n contiguous characters, which are individually counted. The k top most n-grams are almost always correlated to the language. The method of n-grams is widely discussed in the next section about feature selection. LingPipe works in a similar way based on the distribution of characters per language.

The outstanding results of 92% and higher from the TextCat language identifier should be refuted, since their data context is different. Their study focuses in particular on well-formatted texts, but tweets are not well-formatted. By first cleansing the tweets, identification may be improved, but will probably not reach the 92%.

After the language of a message has been identified, the model which has to be trained can assume that all messages are from the identified language. Therefore it can be assumed that only words from the identified language appear in messages.

4.2 Supervised Classification

4.2.1 Feature Extraction

Feature extraction is the process where properties are extracted from the data. These properties, called features are characteristics from the data, because in practice the whole input data is too large to use in classification. The features should be discriminative, to describe the original data as well as possible. On the other hand the features should reduce the space to prevent redundancies and a high dimensionality of the data. Researchers proposed and experimented with different features in the sentiment analysis, to test which extraction methods worked well. The following features are discussed: *n*-grams, the tf-idf measure and part-of-speech (POS) tagging.

N-gram model

The n-gram method is a relatively simple algorithm. A n-gram is a contiguous sequence of n items from a textual or spoken source. The items are usually letters or words. The letter variant is applied in TextCat and the words variant is applied as feature extraction method in this research. In case of unigrams ($n = 1$), each text (tweet) is a document and is split up into words. Counting the frequency of all the words in a all documents results in a word frequency table. High frequent words have a higher probability to appear in texts, therefore those words describe the dataset better. There are two possible ways of measuring the frequency among all documents: the summed term frequency and document frequency. Those two frequency measures are based on respectively the term frequency and the term presence. The term frequency ($tf(w, d)$) is the number of times that a word w occurs in a document d, stated in equation 4.1. In computing the term frequency, all occurrences of a word in a document are counted. Therefore the term frequency can assume a value in the interval $[0-n]$, where n is the total number of words in the document. The term presence ($tp(w, d)$) only checks if a word w is present within a document d, which results in a binary value. This measure is stated in equation 4.2. The main difference is therewith how the words are counted within a message.

$$tf(w, d) = |\{w \in d\}| \tag{4.1}$$

$$tp(w, d) = \begin{cases} 1 & \text{if } w \in d \\ 0 & \text{if } w \notin d \end{cases} \tag{4.2}$$

The summed term frequency ($df(w, D)$) sums all the term frequencies of a word w across all documents D, the formula is stated in equation 4.3. The document frequency refers to the number of documents in which a word occurs. This is formalized in equation 4.4, where the document frequency is $df(w, D)$ of word w across all documents D. Often the set of documents is referred as 'corpus' in more formal linguistic jargon.

$$stf(w, D) = \sum_{d \in D} tf(w, d) \tag{4.3}$$

$$df(w, D) = |\{d \in D : w \in d\}| \tag{4.4}$$

However, frequent words are not necessarily good features for classification. If the distribution of a highly frequent word is uniformly distributed over the classes, then its discriminative power is low. For example the word "special" is highly frequent but appears as often in positive, negative and neutral documents, and is therefore not discriminative for one of the classes. In classification of new cases the classifier can randomly select one of the classes. Therefore a feature selection method that takes into account how well a feature discriminates between the classes is additionally used.

Extending to higher order n-grams, the texts are not split up as single length items, but n-length items. In case of $n = 2$ (bigrams), the items consist of two consecutive words. The set contains all combinations of two words that are consecutive in the original text. The same holds for $n = 3$ (trigrams), and higher n values and also for letter based n-grams. Intuitively, those higher order n-grams seem to capture the relation of the consecutive words better, e.g. in negation.

Finally, a selection is made of the most valuable words. Only the top k most valuable n-gram features, according to the weight measure are selected to form the feature vector. This reduces the dimensionality of the data.

In the literature, several researchers [2, 16, 30, 33] have implemented classification models based on n-grams. The uni- and bigram variants are implemented the most, and to a lesser extent trigrams. Comparing those uni- and bigrams does not lead to an unambiguous answer which is superior. Pang et al. [33] achieved somewhat better results with unigram term presence than with unigram term frequency. They concluded therefore that the term presence works better than the term frequency. They also performed tests with bigram term presence and a combination of uni- and bigrams term presence, but these results did not outperform the results of the unigram term presence. Meanwhile, Go et al. [16] achieved better results with the combination of unigrams and bigrams than just unigrams or just bigrams respectively. The separate unigrams still worked out better than the bigrams in his research. In contrast, Pak et al. [30] achieved better accuracies with the bigrams than with the unigrams. This could be caused by the different application and the different domain he used. In accordance with Pang et al. and Go et al., Agarwal et al. tend to the same conclusion that unigrams are better applicable to Twitter than other n-gram models. It sounds somewhat counterintuitive because bigrams seem to capture the relation between words better than unigrams. However the fundamental ideas originate from movie reviews [33, 40], where models with unigrams outperformed models with higher order n-grams.

Tf-idf Measure

The tf-idf (term frequency-inverse document frequency) measure is a statistic that reflects the importance of a word across a set of documents. This measure is composed of two individual measures: the term frequency and the inverse of the document frequency. The term frequency and the normal document frequency were discussed is previous section 4.2.1 and their formulas are respectively stated in equation 4.1 and 4.4. The inverse document frequency is used to measure the rareness of a word across all the documents. How higher the value of the inverse document frequency, how more rare the word across the set of documents is. The inverse document frequency, $idf(w, D)$ of a word w across all documents D is shown in equation 4.5. This can be combined together with the term frequency to the tf-idf measure shown in equation 4.6. In this equation, the tf-idf is the $tf\text{-}idf(w, d, D)$ of a word w in a document d across a set of documents D.

$$idf(w, D) = \log \frac{|D|}{df(w, D)} \tag{4.5}$$

$$tf\text{-}idf(w, d, D) = tf(w, d) \cdot idf(w, D) \tag{4.6}$$

However, it is the question whether the tf-idf is a good measure for feature selection in this research. The tf-idf value is high, when a word occurs often in a document and does not occur often within all documents. In Twitter domain the documents are the tweets, which means that words with a high frequency within the tweet and a low frequency over all tweets have a high tf-idf value. These words do not seem to be the best words to use for classification, since they do not cover many tweets. Therefore the tf-idf measure is not considered as feature selection in this research.

Part-of-speech Tagger

A part-of-speech tagger or shortly a POS tagger, is a method of marking up a word in a text corresponding to a particular part-of-speech. Part of-speech is also known as word class or lexical category, which are more intuitive names. Within a text, all words are classified to their corresponding lexical category. The idea behind this is that only a limited set of words in a sentence indicate the sentiment, referred to as the sentiment-words. In the English language examples of lexical categories are noun, verb, adverb and adjective. Some of those lexical categories contain sentiment-words more often, such as adjectives and adverbs. A problem of the POS tagger could be a word that can appear in more than one lexical category, however it only expresses a sentiment in in one lexical category.

Since the lexical categories are not the same for all languages, each language needs to use its own POS tagger. In section 4.1 the language identification process has solved this problem. The identified language can then be used in the POS tagging method.

The POS tagging technique is often used, which has been applied in several papers [2, 16, 30, 33, 40]. In most papers it is used as additional feature to a n-gram model, unigrams or bigrams. Go et al. achieved worse results with the POS-tagger than without. Like Pang et al., Go et al. concluded that the POS tagger doesn't improve the quality of the model. In other papers the authors concluded the same, [6, 22]. However there are some papers, e.g. Pak et al. [30], which conclude that a POS-tagger actually improves the performance. They claimed that some POS-tags are good indicators of emotional texts. In line with Pak et al., Agarwal et al. [2] came up with results that improve with the addition of POS features. Additionally, Barbosa et al. achieved better results with a POS-tagger in [5], however his research was based on manually created biased dataset. This results of this last research are therefore less relevant for this research. Since Turney [40] didn't use the POS tagger together with the unigram model, he had no results to compare with. It seems that it is much harder to conclude something about the POS tagger than the n-gram model. There are some researchers which achieved better results with POS-tagging, while others achieved worse results. Even in the literature about movie reviews the POS-tagger did not outperform other models. Because of this discordance, the POS-tagger isn't a method with convincing properties for sentiment analysis, and is not applied in this research.

4.2.2 Classification

The area of classification algorithms has been studied extensively during the past decades. Classification algorithms like nearest neighbor, naive Bayes, maximum entropy and support vector machines (SVM) are applied in many different domains.

Nearest Neighbor

Starting with the most simple of those algorithms, the nearest neighbor algorithm. As far as I can tell, this algorithm hasn't been applied in sentiment analysis. This could be explained by the 'curse of dimensionality' in combination with binary variables, like the term presence. Nearest neighbor algorithms have been applied in domains where the number of dimensions is low. It is known from the literature [3, 18] that nearest neighbor has more difficulties in higher dimensional space. The variance of distances drops and therefore the classification becomes less meaningful. With the binary variables either the distance between points for a specific variable is minimal or maximal, because there is no continuous scale. Contrary to sentiment analysis, in text classification nearest neighbor achieves reasonable results in combination with the tf-idf measure. The disadvantages of applying the tf-idf measure as feature selection have been discussed in section 4.2.1. Therefore the nearest neighbor algorithm is not applied in this research.

Naive Bayes

Mannng et al. distinguish two naive Bayes' models in [25]: the multinomial and the Bernoulli. The multinomial naive Bayes model generates one term from the vocabulary in each position of the document. The Bernoulli model generates an indicator for each term of the dataset. The value 1 indicates that a term is present and a 0 indicates absence. This matches with the unigram model, that is presented in section 4.2.1. The Bernoulli model does not count the multiple occurrence of terms whereas multinomial naive Bayes does. From the literature study in section 4.1, I concluded that the term presence worked better than the term frequency, and therefore I prefer single occurrence above multiple occurrence. Both models assume the same basic principle of Bayes, which is discussed below.

The naive Bayes algorithm uses Bayes' theorem (4.7). The formula $P(C|F)$ states the conditional probability of C given F, where C is the class label and F a feature.

$$P(C|F) = \frac{P(C)\ P(F|C)}{P(F)} \tag{4.7}$$

Bayes' theorem provides a mathematical rule explaining how you should change your existing beliefs in the light of new evidence. It allows to calculate unknown conditional probabilities from a known conditional probability together with the prior probabilities. In naive Bayes, it is assumed that the variables are independent from each other within each class. Namely, the presence of a feature is unrelated to the presence of any other feature. This is formalized in equation 4.8. Although the conditional independence assumption is not realistic, the model performs quite well. Equation 4.9 states the naive Bayes model, where the C is the class label and $F_1, .., F_n$ are the feature variables. Since the denominator of equation 4.7 does not depend on C, it becomes a constant scaling factor $\frac{1}{Z}$ in equation 4.9. In formal language it would be: the posterior probability is computed by multiplying the evidence scaling factor with the prior probability, multiplied by the product of the independent likelihoods. The advantage of naive Bayes models is the fact that a relatively small training set is sufficient to train the model, because independent variables are assumed only the variables for each class are needed. Together with the performance which is often quite good, the naive Bayes model is a good model to use as a reference for testing the quality of other models.

$$P(F_i|C, F_j) = P(F_i|C), \qquad \textbf{for } i \neq j \tag{4.8}$$

$$P(C|F_1, \ldots, F_n) \propto \frac{1}{Z}\ P(C)\ \prod_{i=1}^{n} P(F_i|C) \tag{4.9}$$

The naive Bayes classifier has been applied in quite a lot of papers about sentiment analysis. In a more general application in [33, 44] and in Twitter specific papers in [7, 14, 16, 30, 31]. Gebremeskel [14] achieved good results with the multinomial naive Bayes classifier, it even outperformed the SVM. Another classifier is the maximum entropy classifier. This classifier has been applied in for example, [16, 33]. Theoretically it should work better than the naive Bayes classifier. However the opposite is true in many practical problems. Therefore only naive Bayes models are considered in my research.

Support Vector Machine

Support vector machines exist in different forms, linear and non-linear. A support vector machine [13] is a supervised classifier. What is usual in this context, two different datasets are involved with SVM, a training and a test set.

Assume we only have two features, and therefore a two dimensional space with a hyper plane. In this two dimensional case the hyperplane is a line. In the next examples we assume this two dimensional space for simplification of the explanation.

In the ideal situation the classes are linearly separable. In such situation a line can be found, which splits the two classes perfectly. However not only one line splits the dataset perfectly, but a whole bunch of lines do. From these lines the best is selected as the "separating line". The best line is found by maximizing the distance to the nearest points of both classes in the training set, [8]. The maximization of this distance, can be converted to an equivalent minimization problem, which is easier to solve. The data points on the maximal margin lines are called the support vectors. An example of a linearly separable dataset is shown in figure 4.1.

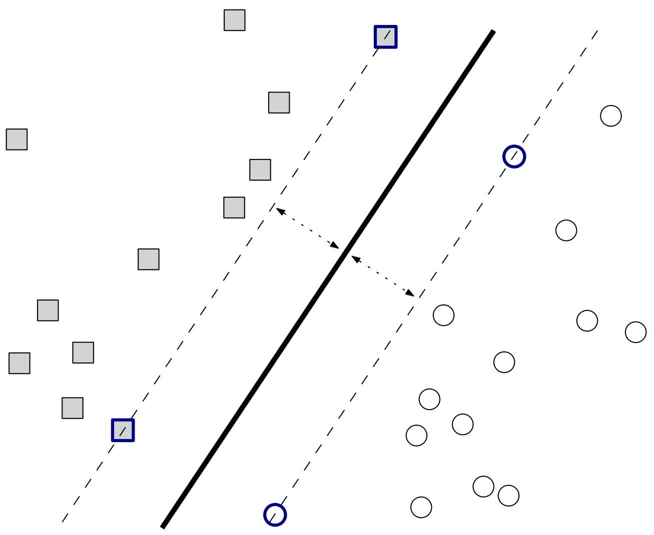

Figure 4.1: A linear SVM with support vectors and a maximum distance to both classes. The *support vectors* are the points that lie on the margin, and are emphasized with dark blue.

In practice the classes are usually not linearly separable. In such cases a higher order function can split the dataset. To accomplish a non linear SVM classifier a so-called kernel trick is applied. A function is applied to the dataset which maps the points in the non linear dataset to points in a linear dataset. Quite simple possible functions are the square root or the square, which could change the data to a linear space. This computation is done implicitly, therefore the user does not have to scale the data to a linear space. The only input that is required is which function type with corresponding parameters must be used. Figure 4.2 shows the separating line in a non linear dataset.

Still, in practice the model in figure 4.2 is not very realistic. Most often datasets are not nicely distributed such that the classes can be separated by a line or higher order function. Real datasets contain random errors or noise which create a less clean dataset. Although it is possible to create a model that perfectly separates the data, it is not desirable, because such models are overfitting on the training data. Overfitting is caused by incorporating the random errors or noise in the model. Therefore the model is not generic, and makes significantly more errors on other datasets. Creating simpler models keeps the model from overfitting. The complexity of the model has to be balanced between fitting on the training data and being generic. This can be achieved by allowing models which can make errors. A SVM can make some errors to avoid overfitting. It tries to minimize the number of errors that will be made. In figure 4.3 an example with outliers is shown. In this example it is not desirable to create a model that perfectly splits the data points. In that case the model is overfitting on the training data, which makes more errors on the test set. The three random outlying data points which are misclassified by the model are shown in red. Sometimes it is impossible to train a model, which achieves a perfect separation. This can only happen when two data points have an identical feature vector and a different class label. A more mathematical explanation of support vector machines can be found in [13] or other SVM literature.

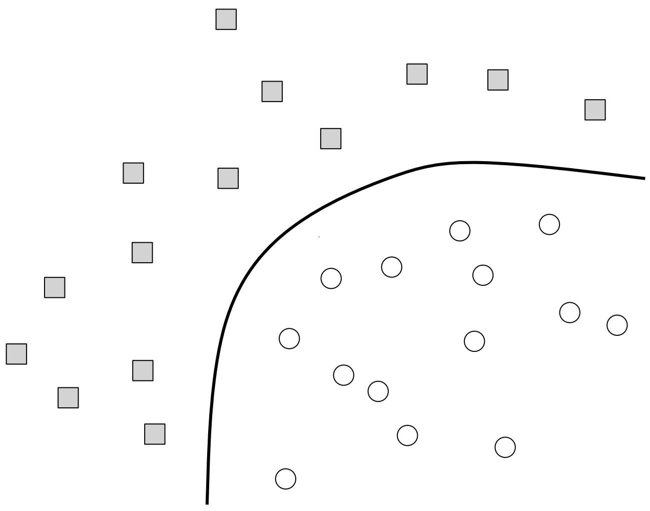
Figure 4.2: A non-linear SVM.

Support vector machines classifiers are applied in many papers, [2, 14, 16, 33]. They are very popular in recent research. This popularity due to the good overall empirical performance. Comparing the naive Bayes and the SVM classifier, the SVM has been applied the most. Go et al. concluded that the three classifiers he used, naive Bayes, maximum entropy and SVM had similar performance. Pang et al. however achieved the best results with the SVM classifier, comparing the same three classifiers. Summarizing the results, the SVM classifier worked out the best in the majority of the papers. Although there exist some differences in the overall performance of the classifiers, testing different classifiers will increase the quality of the system.

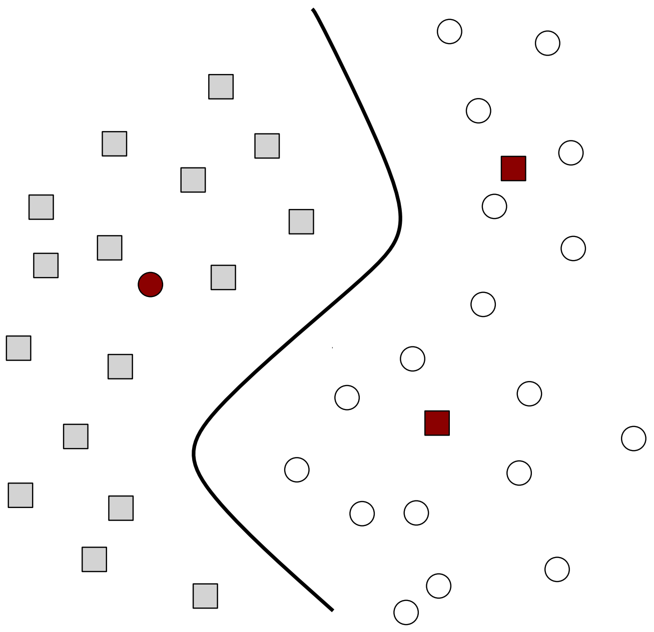

Figure 4.3: A SVM trained on a dataset with some outliers.

4.3 Unsupervised Classification

Point mutual information (PMI) is a simple association measure, which can be used for unsupervised learning. The classification is based on the average semantic orientation, which is stated below. The method makes use of reference words, for most positive and most negative association. For the sentiment orientation (SO) a point mutual information measure is used. It quantifies the discrepancy between the probability of the coincidence given the joint distribution and the probability of the coincidence, assuming independence. Point mutual information is defined as:

$$\mathrm{PMI}(w_1, w_2) = \log_2 \left(\frac{p(w_1, w_2)}{p(w_1)\ p(w_2)} \right) \tag{4.10}$$

This PMI measure is used in a sentiment orientation function SO (shown in equation 4.11), it formalizes the dependence of the positive and the dependence of the negative sentiment. The reference words "excellent" and "poor" are based on the one star and five stars rating respectively in a five star review rating system.

$$\mathrm{SO}(w) = \mathrm{PMI}(w,\ \text{"excellent"}) - \mathrm{PMI}(w,\ \text{"poor"}) \tag{4.11}$$

This semantic orientation function can be applied to all the extracted words in a message. Averaging all those sentiment orientation values of a message results in a quality. This number can be interpreted as a sentiment according to the positivity.

The point mutual information measure is used by Turney in [40]. Turney created an unsupervised classification algorithm based on the sentiment orientation of phrases of movie reviews. After applying his algorithm, he achieved somewhat disappointing results on a movie database. He explained the bad performance by the difficult movie review dataset. Rothfels et al. [34] achieved similar disappointing results with their implementation on reviews.

Zagibalov et al [45] implemented an almost unsupervised approach to seed word selection for sentiment classification. This seed word selection starts with human-selected seed and uses an iterative method to extract a training subcorpus. Similar to the PMI measure they measured the association with positive and negative words. Their results were much better than with the PMI measures, and in most cases close to the results of supervised classifiers.

The area of unsupervised classification algorithms is somewhat underdeveloped compared to supervised classification algorithms. In the field of Twitter sentiment analysis, unsupervised learning algorithms are even less studied. Whilst reviews mostly have a (star)rating, Twitter doesn't have a sentiment label, therefore manually labeling is a prerequisite to apply supervised classification algorithms in Twitter domain. Manually labeling data isn't necessary in the application of unsupervised classification algorithms. However for the validation of the model a class label is always necessary, for supervised and unsupervised models.

In this research, the applicability of unsupervised classification algorithms will be investigated. However the focus of my research will be more on the supervised algorithms, since they have achieved a better overall performance.

Chapter 5

Experiments

This chapter describes the environment, the setup of the experiments and finally the results of the experiments. Those three components will be described extensively in their own section.

5.1 Experiment preliminaries

The input data for the experiments, as well as for the integrated system is described in this section. Furthermore an overview of the development environment and the scientific environment will be discussed.

5.1.1 Input data

The company Twitter offers different methods to access their data. Two types of application programming interfaces (API's) are offered to communicate with Twitter. A REST API, which provides a simple interface to the Twitter functionalities, and a Streaming API which is a powerful real-time API. The access to the public data of Twitter is extremely limited with the REST API, and less so for the Twitter Streaming API. The Twitter Streaming API is designed for the more inventive uses of the Twitter API. Neither of the two provides full access to the public posted tweets. For full access to the Twitter data, commercial packages can be used from a third-party. DataSift is such a company which provides full access based on a price per use base. I have chosen to use the Twitter Streaming API, because the limitations of this API are not restricting the essence of my research. The non-full access of the Twitter Streaming API still provides a huge amount of sampled data for scientific purposes.

The Twitter Streaming API provides a low latency access to the stream of Twitter's data. Efficiently processing large amounts of data can be realized with complex event processing (CEP). Complex event processing consists of processing many events, analyzing the events and taking a subsequent action. This is all done in real-time, therefore the characteristics, high-throughput and low-latency processing of event streams are necessary. Since the release of Microsoft SQL Server 2008 R2 in 2010, Microsoft offers its own CEP product. Their CEP technique StreamInsight is integrated in the C# .NET environment. Because Avanade focuses solely on Microsoft-enabled solutions, StreamInsight is a prerequisite for the experiments, based on its positive features. The common property of the Twitter Streaming API and StreamInsight is the low-latency characteristic.

Datasets

Besides the real-time data from Twitter, a dataset from a third party is acquired. This dataset from Sananalytics [35] contains 5513 tweets about Microsoft, Twitter, Google and Apple. Hereinafter this dataset is referred to as the Sananalytics dataset. Each tweet is pre-classified as positive, negative, neutral or irrelevant. The irrelevant class represents the tweets which are not identified as English in the pre-processing phase. This corresponds with my method where I filter out the English tweets. In contrast, my method does not label the non-English tweets as irrelevant, but leaves them out. The dataset contains therefore 3727 'relevant' messages. The distribution of the three classes positive, neutral and negative is: 570/ 2503/ 654. The labels of the messages in this Sananalytics dataset are assigned by an American male who speaks English fluently. The dataset is filtered based on hash-tags of Google, Microsoft and Twitter and the account of Apple (@apple). Actually the account @apple is not an official account of Apple Inc, however the tweets are almost all related to Apple Inc. The dataset is built up from 15 October until 20 October 2011.

Beside the Sananalytics dataset, a self-built dataset is collected and a sample is hand labeled. This dataset contains more than 900.000 tweets in the period from 16 May until 30 May 2012 about the mobile phones and their brands listed in Appendix C. From this dataset a random sample of 5000 is taken to create an unbiased dataset, and is hand labeled by myself. The random sample is not uniformly distributed over the classes, just like the distribution of the Sananalytics dataset. The number of neutral cases in the self-built dataset is even larger and the distribution accords 3441/ 721/ 435 for respectively the neutral, positive and negative classes.

5.1.2 Development Environment

The development environment is based on Microsoft technologies, since Avanade is a Microsoft party. The system is implemented with C#.NET, ASP.NET and Silverlight technologies. The data is stored in Microsoft's SQL Server 2008 R2 database, since it offers a stable database environment. StreamInsight is the Microsoft technology that offers to process the complex events (CEP). With StreamInsight the tweets can be handled as events and processed with low-latency and a high throughput. Furthermore, pre-processing is based on standing queries and computationally fast algorithms. Initially the classification of the tweets was part of the analysis in StreamInsight. However the classification of a tweets does not correspond to the low-latency characteristic. Therefore the classification process is separated from StreamInsight's analysis phase, and performed in a batch process. The analysis of StreamInsight consists of the pre-processing and filtering of the tweets before the actual classification.

5.1.3 Scientific Environment

The scientific environment R [15] provides the functions to retrieve statistics of the quality of the system. R is a software environment for statistical computing and graphics. It is a popular scripting language among statisticians and data analysts. The scripting language R provides an open source command-line interface. R also allows the user to write his own functions, which can be invoked from the command-line. However, the environment to write functions does not provide any support for indicating errors in an early stage, e.g. a smart compile functionality, which make it somewhat harder to use. R provides some basic functions which can be used in computations. Additionally, packages can easily be installed and loaded to gain extra functionalities. A huge number of packages are available to support different functionalities, e.g. for text-mining, support vector machines and naive Bayes.

The scientific environment R can be used in isolation, but for the integrated system a port from C#.NET to R is used. Two parties provide such a port: statconnDCOM [37] and R.NET [1]. The R.NET seemed to work out simpler in the testing phase, however it is not as stable as statconnDCOM. In addition the statconnDCOM is a more developed product which is also available in a commercial variant, whereas R.NET can been seen as an open source hobby project. The stability forces me to choose for the statconnDCOM as connector between C#.NET and R. The analysis function is defined in R and invoked from the C#.NET environment.

5.2 Experiment Design

5.2.1 Pre-processing

For the integrated system it is important to have a high quality input, to ensure the overall quality of the system. The data from Twitter with additional information doesn't provide many useful properties to structure the raw messages. Therefore to improve the quality of the input data a pre-processing phase is required. The pre-processing phase consists of several techniques: keyword filtering, message caching, language filtering and message cleansing.

The pre-processing phase is tested on a self acquired experimental tweet dataset. The collected dataset consists of 30388 tweets, including retweets and is a different dataset than discussed in section 5.1.1. This dataset was created during a 3,5 hours execution of the the pre-processing part of the tool. This tool will be extensively discussed in chapter 6. During the scope definition the decision to analyze tweets about mobile phones is taken. Therefore the keyword filtering is based on three popular mobile phones. The tweets that are acquired are about the mobile phones iPhone, Nokia Lumia and Samsung Galaxy. This decision is based on their popularity at the moment and the different operating systems iOS, Windows Phone and Android respectively.

Keyword Filtering

The keyword filtering is provided by the Twitter Streaming API, which allows to limit the messages that are included in the stream. A list of terms can be passed as parameter to the Twitter Streaming API. Then only the tweets which contain an element of the list are included in the data stream. The size of this term list is limited to a maximum of 400 terms. The relevant mobile phones and brands that are used, are listed in Appendix C. Therefore only tweets containing the predefined mobile phones and brands are included in the stream.

Message Caching

The Streaming API streams all tweets almost directly after they are posted. It can happen that a tweet occurs multiple times in the stream, therefore the same message can be captured several times. Ignoring this problem could lead to a distorted sentiment. A caching mechanism could prevent tweets to occur multiple times in the dataset, which should lead to a more realistic sentiment.

Each tweets contains an id, which is a 64-bit unsigned integer. This allows to cache the tweets based on their tweet id. With a cache size of 50 the datasets did not contain any duplicate tweet ids. Even a test with a cache size of 20 did not contain any duplicates in a test set of 17602 instances. However in the implementation, I have chosen a cache size of 50, for the extra margin. With implementing an id cache it is not guaranteed that there is no duplication within the dataset, there is only no duplication in the cache.

Language Filtering

Since Twitter is used all over the world, the messages can be in any language. Sentiment analysis makes use of language specific content and therefore the language plays an important role. In section 2.2, I defined the scope to solely English tweets. In the application and analysis I assume that every tweet is English. Before this assumption can be made the non-English messages have to be filtered out.

The implemented language filtering is based on the TextCat text categorization. NTextCat is a .NET variant of the TextCat classifier, which is used in C#. You can train this classifier model your-self, or use pre-classified models. My implementation uses the original pre-classified language models of TextCat, because they are extensively used and proven to work. Language classification is based on letter based n-grams, which is extensively discussed in section 4.2.1. The original included language models cover a wide range of languages, even some languages that are quite similar to English. I removed languages like: Scots, Irish, Welsh and Scots Gaelic to enlarge the number of messages classified as English. The messages that originally are classified as one of those languages, have now a higher probability to be classified as English.

Message cleansing

Tweets contain several special features, collectively referred to as "Twitter orthography", as discussed in section 3.2.2. Extracting this content will create the opportunity to use this content as features. Those features can then be used for the classification of the actual message. Furthermore, because of the limited message size and the freedom in composing, incorrectly formatted tweets are contained in the dataset. To create a more pure dataset for training the classifier, message cleansing techniques are applied. My message cleansing consists of several techniques to filter out abbreviations, acronyms and letter overuse.

All those techniques rely on regular expressions, which is a powerful string replacement algorithm.

Different filters are applied in the experiments, which are used in sequel. The following filters were applied sequentially: keyword filtering, Twitter feature filtering, overuse filtering and replacement filtering. The first case consists of only keyword filtering, since only messages in the defined domain are considered. The second case consists of case 1 plus a filtering based on the special Twitter features: "RT", "#"-hashtags, "@user"-mentions, emoticons and urls. The next step is filtering overuse of letters or punctuations (dots, question and exclamation marks). The last filtering step, acronym and abbreviations replacement is fueled by the character number restriction of Twitter. A short research on frequently used acronyms and abbreviations, leaded to a dictionary for replacement of the word elements.

5.2.2 Feature Extraction

In sequel to the pre-processing techniques, the features need to be extracted before the classification can start. As became clear from the literature study relatively simple algorithms worked out quite well. I decided to implement a feature vector based on word unigrams. This decision is grounded by the performance discussed in the literature study and the simplicity of the model.

The filtered dataset from the pre-processing is used to extract features, which are subsequently used for creating a classification model. This feature extraction and classification is implemented in R. To create the unigrams all messages have to be split into separate words. This will lead to a large list of words. A frequency table will provide an overview of frequently occurring words in the messages. Frequently occurring words have a higher probability to be contained in new messages. Since the classifier has to classify new data based on the known data, the frequent words are good features to train on. However frequent terms are not always the best features, as become clear in this research. For example, when a frequent word is equally divided among the classes, it hasn't got any discriminative power. In decision trees, this discriminative power has been formalized in a mutual information measure. A decision tree is a predictive model which maps observations about an item to conclusions about the item's class label. The construction of a decision tree tries to find the best split at each level of the tree. The best split is obtained by computing the gain of mutual information. Examples of such measures are the misclassification error, Shannon entropy and the Gini index.

Several feature selection methods are compared on performance, including the Gini measure. The bases for the selection is the input, which consists of the messages, the different classes and the length of the feature vector. For each class the words are counted, which results in a matrix of word frequencies per class. These word frequencies per word are used to calculate the Gini index. The Gini index that I use, is actually the inverse of the real Gini index, since total equality should have a worse score than total inequality. In equation 5.1. the inverse Gini index (IGI) of word w is stated. The term $p(c_k|w)$ is the conditional probability of class c_k given the that the word is w. The term $p(c_k)$

is the probability of class k, and $p(w)$ is the probability of word w.

$$IGI(w) = \sum_k [p(c_k|w)]^2 \tag{5.1}$$

Basically, the Gini index is used in a weighted form, in which the weighted mean of the children is computed. In such case my dataset is split at the occurrence of a word w, which is shown in figure 5.1. In the left child node the messages that included word w end up, and in the right node the messages that do not included word w end up. Because most message do not contain the word w, they end up in the right child node. Therefore the right node has almost the same distribution of the classes as his root node. Since the distribution of the root node and the right child node are almost equal, this is not interesting for the determination of sentiment-word w. Therefore, I only use the inverse Gini index of the node that includes word w, which is the grayed left child node in figure 5.1.

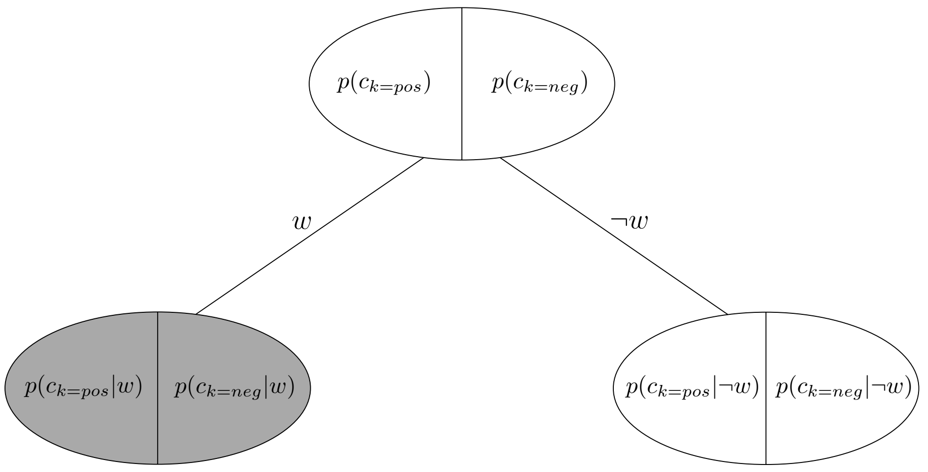

Figure 5.1: The split node based on whether it contains word w, in my Gini measure only the grayed node is used.

The system uses the following custom defined algorithm to select the features. The method takes the messages with the their class label and the different classes as input. Each tweet is then split up into word unigrams. I developed four quality measures: the document frequency, the inverse Gini index, a sentiment-word indicator and a combined measure of the document frequency with the inverse Gini index. The document frequency and the inverse Gini index have already been discussed. The sentiment-word indicator is a binary value whether the word is a sentiment-word, according to a sentiment-word list. This sentiment-word list consists of solely positive and negative sentiment-words from the "subjectivity" dataset, which is available in the sentiment package of R. A fragment of this sentiment-word list is listed in Appendix A. The combined function uses the inverse Gini index together with the document frequency as is shown in equation 5.2. The $IGI(w)$ refers to the inverse Gini index from equation 5.1 and the $p(w)$ is the probability of word w. The logarithm in the formula is used to constrain the influence of the frequency. Furthermore, the plus 1 in the function is needed to overcome a zero value of the logarithm. Those four quality measures are used to rank the word unigrams. Because four quality measures are used, the ranking that can build depends on which quality measure is used first in the sorting. A different order of those 4 measure results in different ranking. From the ranking only the top 100 unigrams words are selected as features to form the feature vector. The different rankings can only be compared after the classification, since the performance of the feature selection can not be analyzed in earlier stage.

$$WM(w) = IGI(w) \cdot (\log_2(p(w)) + 1) \tag{5.2}$$

5.2.3 Classification

For the classification I considered many algorithms to classify the tweets. However not all of these algorithms are included, I made a selection to include in the thesis. The tests that are not included, achieved similar results to the included tests. The following models are included in this thesis:

Sananalytics dataset

1.1 Simple algorithm

1.2 Simple algorithm 2

1.3 Naive Bayes algorithm

1.4 SVM algorithm, linear with the WM measure

1.5 SVM algorithm, polynomial with term presence

1.6 SVM algorithm, polynomial with sentiment-words

1.7 SVM algorithms with k-fold cross-validation

Self-built dataset

2.1 SVM algorithm, polynomial with sentiment-words

2.2 SVM algorithm, polynomial with term presence

2.3 SVM algorithms with k-fold cross-validation

I started with the relatively simple unsupervised algorithm that makes use of the sentiment-word list "subjectivity". The model compares each word from a message with the sentiment-word list. Each word in the sentiment-word list has been assigned to the positive or negative class. Each word also has a subjectivity value which expresses if the word is weak subjective or strong subjective. Therefore four sentiment classes can be distinguished: strong positive (sp), weak positive (wp), weak negative (wn) and strong negative (sn). The function $tsv(m)$ in equation 5.3, computes the total sentiment value of a message m by summing all sentiment values $sv(w)$ of all the words. The $sv(w)$ formula in equation 5.4 determines a positive value for the positive sentiment-words and a negative value for negative sentiment-words. There is also made a separation between the strong subjective and the weak subjective words. A parameter l can be set to influence the impact of strong subjective words. I tested with some values of l to compare the results with each other. Finally, the function $simple(m)$ determines the sentiment from the total sentiment value.

$$tsv(m) = \sum_{w \in m} sv(w) \tag{5.3}$$

$$sv(w) = \begin{cases} 1 & \text{if } w \text{ is positive} \wedge w \text{ is weak subjective} \\ l & \text{if } w \text{ is positive} \wedge w \text{ is strong subjective} \\ -1 & \text{if } w \text{ is negative} \wedge w \text{ is weak subjective} \\ -l & \text{if } w \text{ is negative} \wedge w \text{ is strong subjective} \\ 0 & \text{else} \end{cases} \tag{5.4}$$

$$simple(m) = \begin{cases} \text{positive} & \text{if } tsv(w) > 0 \\ \text{neutral} & \text{if } tsv(w) = 0 \\ \text{negative} & \text{if } tsv(w) < 0 \end{cases} \tag{5.5}$$

In sequel another simple model is built. This idea was fueled by the implementation of the first simple model. The first simple model computes a score based on manually assigned scores to the four different sentiments. Instead of assigning a score to the four sentiment classes, the score can also be learned from the data. For each tweet the frequency of the four sentiment classes are computed. With those frequencies, linear discriminant analysis determines how the four sentiment classes contribute to the actual sentiment of a tweet. I also tested with the quadratic discriminant analysis, but the results of this model are not included, since they were similar to the linear discriminant analysis.

Furthermore, I implemented a naive Bayes algorithm. The feature vector consists of the 100 most frequent word unigrams, with respect to document frequency. I applied the naive Bayes function that is included in the R package 'e1071'. This classifier is used as reference in comparison to the performance of the other algorithms.

The last 4 models are support vector machine models which are applied on the Sananalytics dataset. The first SVM model is a linear model based the top 100 features, according to the combined weighted measure in 5.2. The second SVM model is polynomial with a feature vector that consists of the 100 most frequent words with respect to the term presence. The third SVM model is a polynomial SVM based on sentiment-words. Finally, I applied the SVM models in k-fold cross-validation. Cross-validation is a technique for assessing how the results of the analysis will generalize to an independent dataset. In k-fold cross-validation the dataset is randomly divided into k "folds", parts. The folds are not completely random, they are divided according to the stratified cross-validation principle. Stratified cross-validation divides the folds randomly, with the additional constraint that the fold distributions are similar to the original distribution of the classes. The classification is applied k times with one of the k folds as test set and the remaining $k-1$ folds as training set.

On the self-built dataset three experiments are applied, two SVM models and a experiment in cross-validation. The two SVM models differ in the way of feature selection; the first is based on sentiment-words and the second on the document frequency. The last experiment applies k-fold cross-validation on this dataset with several SVM models.

5.3 Discussion of Results

This section shows the results of different experiments that I executed during my research. First the results of pre-processing are shown. These results are followed by results of the feature selection and classification. These results of the feature selection and the classification are combined, since they can not be separated from each other. The classification algorithm needs the features to classify, and the feature selection does not achieve meaningful results without the classification.

5.3.1 Pre-processing

	English	**non-English**	**unclassified**
keyword filtering:	10,928	4,050	15,410
Twitter feature filtering:	14,529	7,242	8,617
overuse filtering:	14,629	7,568	8,191
replacement filtering:	14,710	7,546	8,132

Table 5.1: Results of the pre-processing, applied cumulatively.

As discussed in the experiment design in section 5.2.1, the pre-processing phase consists of different filtering steps which are applied in sequence. The results are shown in table 5.1. First, I experimented with the TextCat classifier on the raw Twitter messages. With only keyword filtering, 10,928 messages of the 30388 are classified as English. This could be improved by filtering out the Twitter content with the help of regular expressions. The number of English tweets increases to 14,529 and the number of unclassified messages decreased from 15,410 to 8,617. With filtering out the overuse of dots, duplicated letters, question and exclamation marks, the English classified messages increased to 14,629 and the number of unclassified messages decreased to 8,191. The last filtering step is the replacement of acronyms and common used abbreviations. The results of this filtering step were disappointing. The number of English classified messages increases to 14,710. Although the results improve, I expected that the replacement of acronyms and abbreviations would improve the results more significantly. This little improvement could be caused by the cumulatively applying of the filtering steps, because a lot data is already filtered out. It could also be explained by the way of classification of TextCat. Since the classifier is trained on the overall usage of letters and letter combinations, acronyms and abbreviations replacement do not change the distribution significantly.

To validate the classification results, I took a random sample of the results and manually labeled them. A random sample of 310 should cover 1% of the data, since the dataset contains 30,388 instances. The manually labeled instances are compared with the predicted labels. The results are shown in the confusion matrix in table 5.2. With these numbers, various performance measures can be computed.

		Predicted class		
		English	**Other**	
Actual class	**English**	160	41	201
	Other	2	107	109
		162	148	

Table 5.2: Confusion matrix of a random sample of the results.

The precision, recall and accuracy are:

$$\begin{aligned}
\text{Precision} &= \frac{tp}{tp+fp} &&= \frac{160}{160+2} &&= 98.77\% \\
\text{Recall} &= \frac{tp}{tp+fn} &&= \frac{160}{160+41} &&= 76.60\% \\
\text{Accuracy} &= \frac{tp+tn}{tp+tn+fp+fn} &&= \frac{160+107}{160+107+2+41} &&= 86.13\%
\end{aligned}$$

tp = true positives: the number of tweets that are correctly classified as English

tn = true negatives: the number of tweets that are correctly classified as non-English

fp = false positives: the number of tweets that are incorrectly classified as English

fn = false negatives: the number of tweets that are incorrectly classified as non-English

The precision is the fraction of all English labeled tweets that are correctly labeled as English. Recall is the fraction of all English tweets that are correctly labeled as English. The accuracy is the fraction of all tweets that are labeled as English. When examining the results, I observed a high precision, and somewhat lower recall and accuracy. Although ideally all of those measures should be as high as possible and approaching 100%, the precision measure is the most important in this case. The inclusion of non-English tweets would pollute the training data and possibly degrade the predictive performance of the system. Because the proportion of non-English tweets after the pre-processing is small, the assumption that all tweets in the training set are English is better grounded. The somewhat lower recall causes that some English are incorrectly classified as another language. This means that some of the English tweets are filtered out in the pre-processing. But the question is whether those out filtered messages are useful for training, which is doubtful since the language is even not clearly determinable by the model.

5.3.2 Feature Extraction & Classification

The executed experiment in this section are based on two datasets, the Sananalytics dataset and the self-built dataset. As discussed in section 5.2.3, I have tested 7 models on the Sananalytics dataset: a simple unsupervised model; a simple model with linear discriminant analysis; a naive Bayes model; three different SVM models and a k-fold cross-validation with different SVM models. Since the performance of the self-built dataset confirmed the performance of the tests on the Sananalytics dataset, I did not test extensively on this self-built dataset. On the self-built dataset I applied 2 SVM models and k-fold cross-validation with different SVM models.

Sananalytics Dataset

I based the first tests on the Sananalytics, the results on this dataset are denoted in the tables as test 1, followed by the model number. In data mining it is common to compare the quality of new models to the "majority model". The majority model classifies all the cases to the largest class to minimize the number of errors. In case of the Sananalytics dataset, the neutral class is the majority with 2,503 out of 3,727. The majority model achieves therefore a classification accuracy of 67.2% on this dataset. A new classification model should beat this model to have some meaning in data mining context. However with this dataset the majority model achieves a rather high classification accuracy. If the dataset is uniformly distributed, the classification accuracy is only 33.3%.

The first six models are trained and tested on the entire dataset, the so-called in-sample testing. The seventh test is applied on a different training and test set, and is called out-of-sample testing. This seventh model applies the k-fold cross-validation strategy. K-fold cross-validation is used to prevent overfitting and creating a more realistic view of the quality of the model.

		Predicted class			
		Positive	**Neutral**	**Negative**	
Actual class	**Positive**	224	274	72	570
	Neutral	459	1637	407	2,503
	Negative	126	293	235	654
		809	2,204	714	

Table 5.3: Confusion matrix of model 1.1 according to the simple approach.

Model 1.1: The first model that is created and applied, is based on a simple approach. This simple model has been explained in the previous section 5.2.3. As shown in table 5.3, this simple model unsupervised learning algorithm performs even worse than the majority model. The classification accuracy for this simple model is 56.2%. This is a confirmation of the choice for supervised models generally perform better. Examining the row and column sums, the distribution of all the classified tweets is relatively close to the real distribution of the tweets. In reporting perspective this looks interesting, because the report is based on the overall sentiment and not on the individual sentiment per tweet. However the distribution of classified tweets is close to the actual distribution of this dataset, it will not directly extrapolate to other datasets. This is because the classification accuracy on message level is rather low.

Actual class	*Predicted class* →	**Positive**	**Neutral**	**Negative**	
	Positive	40	519	11	570
	Neutral	46	2,392	65	2,503
	Negative	15	563	76	654
		101	3,474	152	

Table 5.4: Confusion matrix of model 1.2 with linear discriminant analysis based on strong positive, weak positive, weak negative and strong negative word frequencies.

Model 1.2: In addition to the simple model in model 1.1, a more advanced model based on discriminant analysis is created and applied. This model has also been described earlier in section 5.2.3. The classification accuracy for this model is 67.3% and performs almost the same as the majority model with a classification accuracy of 67.2%. The results of the prediction are shown in table 5.4. In contrast to model 1.1, this model predicts a very large proportion of 93.2% of the messages to the neutral class. However, the classification accuracy of this model is higher than model 1.1, it is not wanted that such large proportion is classified as neutral. This is probably cause by the dominance of the neutral class in the dataset.

Actual class	*Predicted class* →	**Positive**	**Neutral**	**Negative**	
	Positive	388	153	29	570
	Neutral	902	1,443	158	2,503
	Negative	222	194	238	654
		1,512	1,790	425	

Table 5.5: Confusion matrix model 1.3 with the naive Bayes algorithm.

Model 1.3: For the third model, naive Bayes model is used to compare the other models. The results of this classification are shown in table 5.5. The classification accuracy is only 55.5%. It performs even worse than the majority model, and since other settings did not improve the accuracy, I decided to stop improving this model. Zooming in on the results per category we see that naive Bayes made a quite different prediction in comparison to the simple model 1.1. In this model a lot of neutral cases are classified as positive. The high number of positive predictions could be caused by a more frequently occurring positive words than negative words in the feature vector. In general, the naive Bayes model predicts more cases to the positive and the negative classes. An explanation of this symptom is the bases of the feature selection. The feature selection is namely computed from the positive and negative tweets, since neutral sentiment-words does not exist. Therefore the absence of positive and negative sentiment-words should indicate the a neutral tweet. Therefore the number of neutral predictions is reduced. The naive Bayes algorithm appears rather to be chosen for the negative and positive, caused by the independence assumption. The effect of the joint distribution seems to be less than the product of the individual distributions, and therefore the independence assumption is not justified.

		Predicted class			
		Positive	**Neutral**	**Negative**	
Actual class	**Positive**	43	521	6	570
	Neutral	4	2,489	10	2,503
	Negative	0	527	127	654
		47	3,537	143	

Table 5.6: Confusion matrix of model 1.4: linear SVM based on the combined WM measure.

Model 1.4: This first SVM model 1.4 is a based on the combined measure WM stated in equation 5.2. The results obtained are shown in table 5.6. It shows were somewhat disappointing results with a classification accuracy of 71.3%. This result was obtained with a linear SVM trained on the features that were selected from all messages. The feature selection is based on the combined document frequency count and mutual information and limited to the top 100 words. The feature selection considers all tweets: positive, neutral and negative. Because of the large number of neutral tweets, many neutral words are selected as feature words. The inverse Gini index function from equation 5.1, also takes this frequency into account in the determination of the purity of the word among the classes. Concluding, the neutral class dominates this model, whereas you would like to have a model that focuses on the positive and negative messages. The next models try to focus more on the positive and the negative sentiment classes.

		Predicted class			
		Positive	**Neutral**	**Negative**	
Actual class	**Positive**	237	328	5	570
	Neutral	5	2,485	13	2,503
	Negative	2	285	367	654
		244	3,098	385	

Table 5.7: Confusion matrix of model 1.5 with a polynomial SVM of degree 3 based on the term presence.

Model 1.5: The disappointing performance of model 1.4 lead to some ideas for improvement in model 1.5. One of the improvements was to use a polynomial SVM instead of a linear SVM. The other improvements were fueled by the dominance of the neutral class and the occurrence of non sentiment-words in the feature vector. The results of this improved model 1.5 are shown in table 5.7. The dominance of the neutral words is reduced by only considering words from the positive and negative tweets in the feature selection. With this improvement the classification accuracy is boosted to 82,9%. The improvement of model 1.5 is achieved by the reduction of neutral false positives, and achieves a higher true positive ratio for the positive and the negative class. The feature vectors of the positive and negative examples show more responses, therefore this model seems to be more intuitively correct. However, this improved performance could also be due to overfitting since it focuses on exclusive words in this dataset.

Looking to the selected words in the feature vector, object words seem to occur relatively often. Since the object words do not express a sentiment in essence, it is the question whether these words are good indicators for the sentiment. Examples of feature words which do not have anything to do with the sentiment are "Android" and "phone". If those words are used as feature words, the model is likely to be biased. It is not desirable to select those words which do not express a sentiment by itself. For example, if "Android" is positive in almost all cases in the training set, the model learns that Android is a good indicator of the positive sentiment. In the test set however, Android doesn't have to be positive at all, it could be negative at that moment. In essence the sentiment does not depend on the word Android, but it is an object, over which the sentiment could be expressed. Therefore by excluding object words such as "Android", the model should become more robust.

		Predicted class			
		Positive	**Neutral**	**Negative**	
Actual class	**Positive**	151	409	10	570
	Neutral	40	2,415	48	2,503
	Negative	13	420	221	654
		204	3,244	279	

Table 5.8: Confusion matrix of model 1.6 with a polynomial SVM of degree 3 based sentiment words and subsequently the combined WM measure.

Model 1.6: Model 1.5 only considers positive an negative tweets to select the features. However in those positive and negative messages non sentiment-words, like "Android" still frequently occur. A model that disregards such words, should solely focus on sentiment-words. In the improved model 1.6 sentiment-words have a higher priority then non sentiment-words. The result of this improved model are shown in table 5.8. If a message contains a sentiment-word, this word will be directly at the top of the feature selection list. In other words sentiment-words have the highest priority in the selection of the features. The sentiment-words are selected from a "subjectivity" dataset, which is available in the sentiment package of R. Incorporating these sentiment-words in the model achieves a classification accuracy of 75,1%. This is somewhat worse in comparison to model 1.5, however this model can not be based on every arbitrarily word, such as object words. The model should be less biased than model 1.5, which selects features exclusively based on the word frequencies from positive and negative tweets.

Model 1.7: The three models 1.4, 1.5 and 1.6 are all tested in cross-validation, but they all resulted in the same confusion matrix shown in the table 5.9. As

Actual class	*Predicted class:* **Positive**	**Neutral**	**Negative**	
Positive	0	570	0	570
Neutral	0	2,503	0	2,503
Negative	0	654	0	654
	0	3,727	0	

Table 5.9: Confusion matrix of model 1.7 with the k-fold cross-validation with $2 \leq k \leq 10$.

the table shows, with k-fold cross-validation all tweets are classified as neutral, which accords with the majority model. Even different k values doesn't worked out to tend the model predicting cases as positive or negative. Taking out the neutral tweets and train and test exclusively on a positive and negative dataset, classified to the majority, in this case the negative class.

Classification to the majority is most probably caused through the sparsity of feature words among the dataset. The training set does not cover the test set, and vice versa. This could be explained by the fact that the extracted features does not describe the sentiment in the test set. The relatively small dataset in combination with the short messages is probably the cause of this sparsity. For example, Go et al. tested on much larger dataset of 1.6 million tweets in [16]. Ideally the training set should cover the whole feature space. In practice this is not the case, however it is pursued to be. To overcome this issue the size of the training set should be increased significantly. In case of this dataset, increasing the size is not possible.

Additionally, a small improvement of about 1% can be achieved with adding some standard emoticons to the feature vector. Obviously in positive tweets, positive emoticons occur and in negative tweets, negative emoticons occur. More remarkable is therefore the fact that positive emoticons do occur relatively frequent in neutral tweets, but negative emoticons hardly do. The emoticons improves the model, however less than expected. This is caused by the fact that positive emoticons not only occur in positive tweets, but also in neutral tweets. The improvement model classifies only a few cases better than a model without the emoticons.

Concluding the simple models together with the naive Bayes performed worst, the SVM models performed better. The radial and in particular the polynomial SVM perform significantly better than the linear SVM. Beside the actual text of a tweet, also a filtered message is considered in the tests. The filtered out content consists of the "#"-hash-tags, "@user"-mentions and urls. The improvement with a model based on the filtered message does not change significantly. Because this filtered content does not occur with a high frequency as unique "word". Therefore this content is mostly not considered as feature in the feature selection. The filtered text is however preferred above the actual text, since fewer word are considered in the feature selection.

Self-built dataset

The Sananalytics dataset only contains tweets about Apple, Google, Microsoft and Twitter, which does not correspond with my defined domain in section

2.2. Similarly to the Sananalytics dataset, the tweets from the self-built dataset are not uniformly distributed over all the products and brands, the neutral class forms the majority. This is just the reality where the largest amount of tweets is actually neutral. Also popular brands and products are more common to tweet about, and therefore appear more often in the dataset. There are some terms that dominate the dataset like iPhone, Apple and Samsung. Other terms seem to occur less often, which is in line with the test set or real data. During the hand labeling some trends can be observed, e.g. the negativity about Blackberry.

As discussed in section 5.2 only 3 models are tested on this self-built dataset. The reason is that the results on this dataset confirm the results on the Sananalytics dataset, on which already a lot of tests are applied. These tests did not improve the performance significantly and especially the tests in cross-validation all achieved disappointing results.

Actual class	*Predicted class* →	**Positive**	**Neutral**	**Negative**	
	Positive	144	567	10	721
	Neutral	51	3,769	24	3,844
	Negative	12	335	88	435
		207	4,671	122	

Table 5.10: Confusion matrix of model 2.1 based on the SVM algorithm with sentiment-words and subsequently the combined measure WM

Model 2.1: The first model that is trained on this dataset is a polynomial SVM. Table 5.10 shows model 2.1. Model 2.1 in this case is a polynomial SVM algorithm based on the sentiment words and the combined measure of the document frequency and the inverse Gini index, the WM measure. This model achieves a classification accuracy of 80.0%.

Actual class	*Predicted class* →	**Positive**	**Neutral**	**Negative**	
	Positive	317	399	5	721
	Neutral	28	3,809	7	3,844
	Negative	9	284	142	435
		354	4,492	154	

Table 5.11: Confusion matrix of model 2.2, which is based on a polynomial SVM of degree 3 based on the term presence.

Model 2.2: Model 2.2 is trained with a polynomial SVM of degree 3 based on document frequency, the results are shown in table 5.11. The classification accuracy that is achieved is 85.4%. This model confirms that the term presence performs better than the sentiment words in combination with the combined measure. The explanation can be found by the coverage of the data, in which the term presence from model 2.2 takes the words which occurs most frequently, the model 2.1 does not. Therefore the feature vector of model 2.2 covers more messages than model 2.1 and can easier separate the tweets from each other.

		Predicted class			
		Positive	**Neutral**	**Negative**	
Actual class	**Positive**	0	721	0	721
	Neutral	0	3,844	0	3,844
	Negative	0	435	0	435
		0	5,000	0	

Table 5.12: Confusion matrix of model 2.3 with applying k-fold cross validation with $2 \leq k \leq 10$.

Model 2.3: Similar to the k-fold cross-validation on the Sananalytics dataset, the k-fold cross-validation on the self-built dataset also accords with the majority model, shown in table 5.12. The self-built dataset isn't much larger than the Sananalytics dataset, therefore the cause can be the relatively small size of bot datasets. As mentioned earlier, the sparsity of data can also be a cause. This is confirmed by some other tests, where the training set is reduced stepwise from the whole dataset to $^4/_5$ of the dataset and a random sampled test set of $^1/_5$ of the dataset. The performance of the model goes stepwise down as expected.

Summarizing the results of the experiments on the Sananalytics dataset and the self-built dataset, the in-sample result of the best SVM model is acceptable with classification accuracy of about 83%, but the out-of-sample experiments are disappointing. When the supervised models are applied out-of-sample the results accords with the majority model. The models are therefore overfitting in the training set.

Chapter 6

Implementation

The implementation of the Twitter sentiment analysis tool consists of different components some of which have been discussed in the classification algorithm in section 5.2. In figure 6.1 the architectural structure of the integrated system is shown. The individual components are grouped by color, to indicate the subsystems. The described flow of the data can be read from the the architectural structure, it becomes more clear with the data flow diagram in figure 6.2. The figure provides a more clear view of how the data flows through the system. The highlighted colors in the data flow diagram correspond to the highlighted colors in the architectural structure. In figure 6.2, there is a cycle between the data warehouse and the analysis phases. This must be read according the numbering, the data flows from the data warehouse to the analysis phase, after the analysis the data is again stored in the data warehouse and finally the analyzed data is aggregated.

Pre-processing and Storing Phase

The source of the system is obviously Twitter, it is indicated in the top *blue* block. The subsystem indicated in *blue* is the process of pre-processing and storing useful data in the data warehouse. The Twitter messages are actually collected with the Twitter Streaming API, which allows a direct access to the latest posted tweets. The shown reference database provides the terms for the filtering track of the Twitter Streaming API. Those terms consist of several popular mobile phones and their related brands, a full list of the products and brands is included in Appendix C. The track of the Twitter Streaming API restricts the messages that are processed. The next phase is the pre-processing which is implemented in StreamInsight [17]. The advantages of StreamInsight have been discussed at length in section 3.3. The tweets are queried with a "standing" query, which filters the relevant tweets that can be used in the classification. Part of this filtering is the language identification; with a language identifier exclusively the English messages are let through. Thereafter common abbreviations are filtered with a predefined dictionary. The entries of this dictionary consist of two elements which are stored in a two-column table in the reference database. The entry consists of abbreviation and the corresponding full word. Algorithm 6.1 shows in pseudo-code how the abbreviations are replaced with a regular expression.

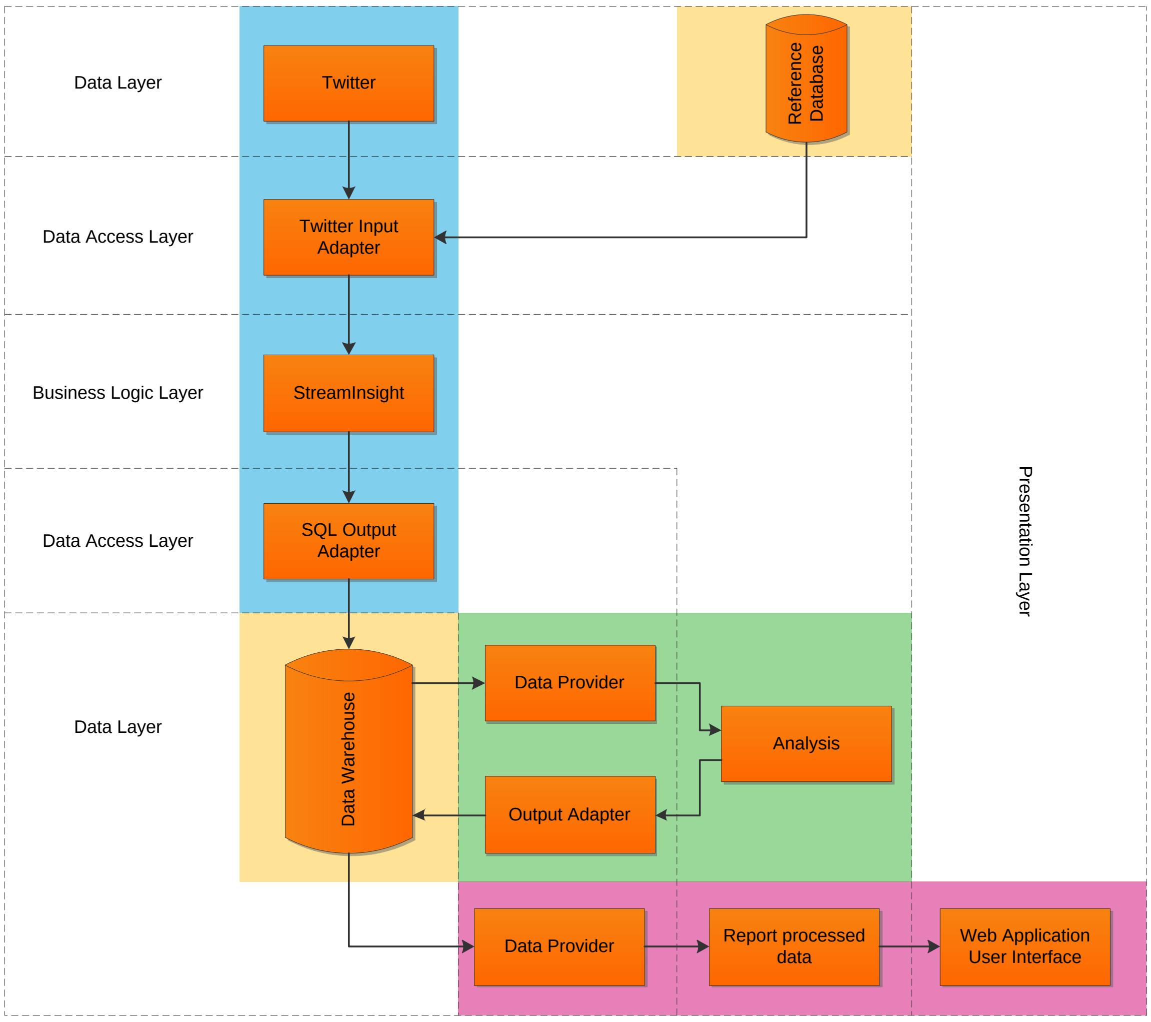

Figure 6.1: Architectural structure of the integrated Twitter sentiment analysis system.

Furthermore the retweets are filtered out, since they are repetitions of previously posted messages. The filtered messages are stored in the data warehouse; together with other useful information.

Data Storage

The central place of the application is the data warehouse, it is used to store the tweets with their sentiment. The stored tweets with their sentiments can then be aggregated for reporting. The useful information that is stored in the

Algorithm 6.1 Replace abbreviations function.

Require: x and y are respectively the tweet and the Map from the abbreviation to full word

```
1: function REPLACEABBREVIATIONS($x, y$)
2:     for ($i$=0; $i < y$.length; $i$++) do
3:         $x \leftarrow$ Regex.Replace($x$, $y$[i].First, $y$[i].Second)
4:     end for
5:     return $x$
6: end function
```

data warehouse includes for instance: the message, the sentiment, the date and the author. More about the exact stored information can be found in figure B.1 in Appendix B. Besides the data warehouse, a database is implemented with three tables. These tables are used in the pre-processing phase, for filtering the products, the brands and the abbreviations. The data that is stored in those three reference tables, is derived from reference data. The abbreviations that are used are for instance collected through a list with the most commonly used abbreviations, as discussed in section 6. Similarly the products and brands are manually constructed, with some popular mobile phones with their related brands. The database storages of the system are highlighted with *yellow* in figure 6.1 and 6.2.

Analysis Phase

The actual classification of the tweets is executed in periodical batches. The classification is executed in the statistical environment R, which has been discussed in section 5.1.3. I have chosen to batch the analysis process to prevent huge amounts of calls to R; one per event/ tweet. Every minute, the tweets are batched in a R-vector, with a maximum of 1000. The batch of tweets is sent to R and analyzed with the trained model. The trained model is used to classify the tweets.

Classification is based on a feature vector, which consist of 100 feature words. For each tweet a feature vector is composed, with the term presence of the 100 feature words. The selection of the feature words is based on several quality measures, which have been tested in section 5.2.

Furthermore, a R-vector of class labels is returned to C#, from which the data warehouse updates the sentiment labels of the tweets. This analysis process is highlighted in *green* in figure 6.1 and 6.2.

Reporting Phase

The reporting process in *purple*, is the system that the end-user will use to query the sentiment about some brand or mobile phone.

The user interface of the application is built with the Silverlight technology, which provides a powerful and interactive user experience for web applications. Silverlight runs client-side, which translates to fast web page response and influences the user experience positively. Creating a Silverlight application makes it easy to bind data to graphs and interactive components. Those interactive components give a smooth look-and-feel on the client-side. The data for the Silverlight client is retrieved by communicating with a Windows Communication Foundation (WCF) service. A WCF is a service to communicate asynchronous from an endpoint to another endpoint. A WCF service is more flexible and is approachable by more protocols than an usual web service. The implemented WCF services provide the products, brands and the aggregated data for the sentiment reports. The figures of user interface can be found in the Appendix B. The first figure shows the start screen, where the user can select a product or brand which can be analyzed. After this selection, the analysis will start and the next screen shows the results of the analysis. The results are plotted in different types of diagrams and are shown in the second figure. The combo-box allows to switch between some reporting periods from a week until half a year prior to the current date.

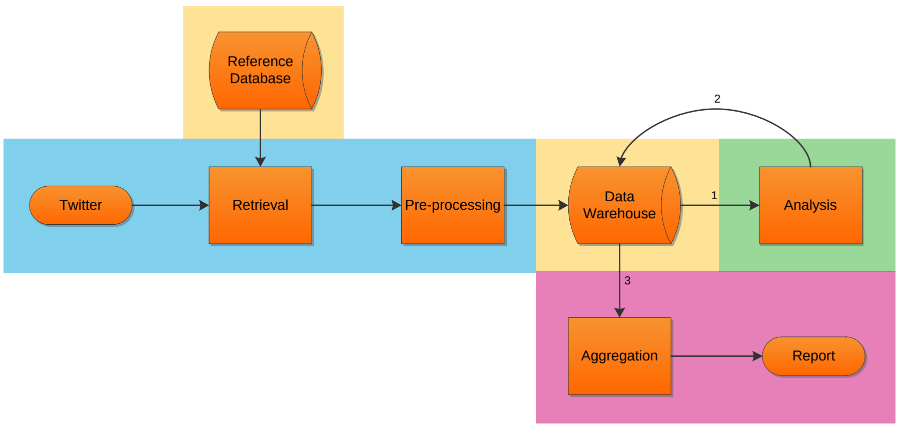

Figure 6.2: Flow diagram of the Twitter sentiment analysis tool.

Chapter 7

Conclusions

In this chapter the research questions are answered and the conclusions of this research are stated. In the scope definition some decisions have been made to limit the scope of this research. In the final section possibilities are described for future work.

7.1 Research Questions

Sub question 1: *Which supervised or unsupervised learning algorithm achieves the best performance for sentiment analysis on Twitter?*

From the literature study in section 3.1, I found out that most Twitter sentiment analysis systems makes use of supervised algorithms. Only a few researchers [34, 40, 45] came up with an unsupervised system. Initially, an unsupervised system would be superior, because no user input is needed for the classification of tweets. Therefore no labels are required for the execution of the algorithm. But an unsupervised algorithm still needs labels for the validation of the model. During my research, I primarily focused on supervised algorithms to achieve a meaningful performance. Unsupervised algorithms are often based on clustering and primitive principles. Since clustering of high dimensional data is difficult to retrieve meaningful results, the performance of such models is worse. Other simple models based on primitive principles like counting the positive and negative words also achieve a worse overall performance. From the experiments in section 5.3.2 and from the literature as well, I conclude that the SVM algorithms achieve a better performance than the unsupervised algorithms. In my tests the naive Bayes algorithm achieved a much lower quality on non-uniform distributed datasets, the SVM models worked out better on the these datasets. The SVM algorithm with the term presence achieved the best performance when the whole set is used as training and test set. The supervised models are trained on specific data and therefore achieve an better in-sample performance. Out-of-sample, in the cross-validation none of the models work better than the majority model. The supervised models seem to accord with the majority model, the unsupervised models even achieve a performance.

Sub question 2: *Does Twitter's additional user information improve the classification accuracy?*

At first sight Twitter's additional user information can be a way to structure the data, at least to get more structure in the almost completely unstructured data of Twitter. The structure is necessary to achieve a good performance of the classification algorithms. Twitter provides a language property and a location property. However this language property is user dependent, it is the language in which the Twitter environment is used. Therefore the messages with a user language property of English can still be composed in another language. Somewhat the same holds for the location property, which users can fill in completely free. The user can leave it empty or even fill it in with a fictional place like, "Here and there" or "Bikini Bottom". The fraction of useless content is larger than the fraction of useful content and therefore the additional information is useless. The additional information looked very promising to create some structure in the unstructured data and retrieve some additional information about the users. If Twitter is able to force the correctness of the information it could be beneficial for third-parties or tools. The additional information can then be used to segment the users into user groups based on age, language and location. However it is the question whether Twitter wants to provide such personal information to third-parties. In their privacy statement they make a clear statement that the additional information is entirely optional, to protect the user's privacy. Therefore it is not supposable that Twitter force the correctness of the additional information.

Sub question 3: *Is it possible to handle negation, sarcasm, irony and double negatives to improve the performance of the classifier?*

As became clear from my literature study, handling sarcasm, irony and double negatives is difficult. This is caused by the difficulty of recognizing those figures of speech purely based on text. Although it is possible to recognize some figures of speech, it results in a low precision. The low precision means that more tweets are recognized as a figure of speech then there actually are. Therefore tweets without figures of speech are indicated as tweets with a figure of speech, which means a higher false positive rate. Since I used feature extraction based on n-grams with n is 1, figures of speech are not included in my models. Intuitively, higher order n-grams can overcome negations, but negations are not always contiguous. From past research [2, 33] it is known that unigrams generally had a better performance than higher order n-grams. During the hand labeling I found out that some of the tweets contained sarcasm, but not in all of those cases it was trivially determinable that they were sarcastic. This is caused by the absence of tone and facial expression in text. A model that achieves a good performance on the data and ignores the figures of speech should be the most important. Concluding, sarcasm and irony are almost impossible to classify correctly, since these are very subtle in text. The only way these could be found is by adding a special hash-tag to the tweet. Negation could be captured in the model, with part-of-speech tagger and higher order n-grams, but the performance of those models does not always increase significantly, as concluded in section 4.2.1. Therefore handling figures of speech does not improve the performance of the system directly.

Sub question 4: *Can the special features of Twitter's orthography, like hash-tags, retweets and emoticons be used to improve the classification of the messages?*

The special features which Twitter's orthography provide, can indeed be used to improve the quality of the model. When some positive and negative emoticons are included in the feature vector, the performance will improve a percent at most on a dataset which contains positive, negative and neutral tweets. Apart from the positive and the negative tweets, the neutral messages also contain emoticons. The emoticons are not only used to express a sentiment, but as the name suggests mainly to express emotions. Not every emotion can be labeled as either positive or negative. Between the positive and the negative class the separation based on emoticons is much clearer. The frequency of positive emoticons in negative tweets is low, the same holds for the opposite. Therefore positive emoticons are a good indicator for non-negative tweets and negative emoticons for non-positive tweets. The retweets are a way of liking someone else's tweet. The question is whether a retweet should be included in the dataset, since they are a repetition of a previous tweet. A retweet could also be interpreted as a form of accordance with the original tweet and therefore an accordance with the opinion. The hash-tags can be used to improve the quality of the model, but not every hash-tag is useful. Therefore a selection of the hash-tags is required, to make it valuable. This selection is a labor intensive process, which is another supervised process in addition to the hand labeling of the tweets. It is therefore not recommended to include hash-tags in the feature vector, since it bias the classification.

Sub question 5: *How many levels are optimal to represent a sentiment realistically and is still interpretable?*

From the tests and results of this research it became clear that most of the tweets can be classified as neutral. Therefore the majority class is neutral, further splitting the positive or negative classes, will even decrease the size of those classes and therefore increase the imbalance of the dataset. A classification algorithm will therefore focus even more on the majority class neutral. If the model incorporates the neutral sentiment class it is not desirable to use more levels. However if the neutral class is ignored, a model with 4 levels is possible. It really depends on the distribution of the classes, which ideally have to be uniform. If the classes are not more or less uniformly distributed a model with 2 (or 3 with neutral) levels of scales is preferred. Extra levels will only disturb the distribution in such situations. Therefore it really depends on the distribution of the classes, if more levels increase the uniformity it is preferred. However, if it decrease the uniformity it is not preferred. Often an increase of the number of levels cause a decrease of the uniformity, therefore a three level model is preferred.

Main question: *How can Twitter messages be accurately classified based on their sentiment?*

In the first place what became clear of this research is that Twitter messages are rather unstructured, which make it hard to classify. If a support vector machine model is trained on the whole dataset and tested on the same dataset the messages can be classified with a accuracy of 84%. The classification accuracy in-sample is therefore quite acceptable, it performs much better than the majority model. However if k-fold cross-validation is applied, the model simply predicts the majority neutral class. The performance out-of-sample is disappointing. Therefore the SVM model from this research can not be used to classify Twitter messages in practice. The practicability of sentiment prediction could therefore not be proven, therefore further research is needed.

7.2 Conclusion

At the start of this project I thought that the limited size of a tweet should make the classification task more easy. However after this research rather the opposite is true, because by the limited size, a tweet expresses less information. The number of words in a tweet is obviously limited by the character restriction. Therefore a lot of messages are needed to cover a valuable amount of data. This can be confirmed with the test results from the experiments in chapter 5.

Furthermore, the data from Twitter is unstructured. In comparison to reviews which have at least a good grammar and syntax, tweets do not have these properties. Therefore part-of-speech taggers work out less well than in movie reviews. Beside the unstructured data, a lot of noisy data is available on Twitter. Since Twitter became more popular in the last couple of years, more companies get involved on Twitter. These companies provide more noisy data, such as advertisements, game updates, vacancies and a much more useless content. This is caused by the open format of Twitter, which allows anyone to post a tweet in any possible format: questions, answers, facts, opinions and conversations. Most important is therefore to select the useful tweets out it, in this case the opinions. This is immediately one of the most challenging tasks.

If I zoom in on the algorithms that are tested, I can say that the support vector machine algorithm performs the best when the model is tested in-sample. The polynomial SVM of degree 3 based on the document frequency performs the best. This is probably caused by the higher coverage of the data than when the same model is based on sentiment-words or on the combined measure of frequency and the inverse Gini index. The datasets contained a large proportion of neutral tweets of about 66%, which sets a high majority model. The impact of neutral tweets on the model is therefore high, and the SVM algorithm will therefore likely choose the neutral class. Taking out those neutral tweets will increase the performance of the model, compared to the majority model. However in cross-validation both the models with and without the neutral class create a model that is similar to the majority model, according to the confusion matrix. This is presumably caused by the relatively small dataset, and through the sparsity of the dataset. The training set does not seem to cover the test set. Ideally, the model should be trained on a dataset which approaches a dataset that covers every possibility. My datasets do not approach such dataset, they are simply too small. The conclusion can therefore be drawn that the difference of the in-sample and the out-of-sample classification accuracy is caused by overfitting. The classification models fit a model on the training set which is not generalizable to other datasets

Summarizing, tweets are short unstructured messages, these properties make it hard to make a good classification. Furthermore, the Twitter data is very noisy, since it is a publicly accessible medium. Unsupervised classification models perform worse than supervised algorithms. Although supervised algorithms such as SVM have problems when applying in out-of-sample, the unsupervised models even perform worse. This is caused by overfitting on the training set, since in-sample testing achieves an acceptable performance. The data in the different folds do not overlap, and therefore the majority model comes out. A very large amount of data is necessary to possibly overcome this problem.

7.3 Future work

During my research I have limited the scope in different ways to focus on a specific area. The limitations that are made, are based on the domain and the algorithms. The domain included mobile phones and their related brands, the algorithms that were used are the SVM, naive Bayes, discriminant analysis and a simple unsupervised approach. Other algorithms could be selected and tested as future work, however in past research the SVM worked out the best. For new work it might be interesting to look to other interesting domains. There already exist companies that focus on Twitter sentiment analysis in other domains. An example of such a company is SNTMNT [23] which predicts the AEX stock prices for the upcoming few days. Their products works with somewhat similar algorithm as in this research, a support vector machine based on n-grams. However they claim to achieve an accuracy of 84.3%, it is not explained or funded by test reports. Therefore I doubt if it works as well as they claim.

The SNTMNT company confirms the fueled interest in social analytics. I think this research field will remain in the spotlight the coming years. The question is whether the supervised algorithms achieve a good performance out-of-sample, and therefore are applicable in practice. At the moment this is most difficult and therefore most valuable for companies. The social analytics still stays in the research stadium, because it can not achieve a reliable out-of-sample performance.

The base of an unsupervised algorithms still look quite interesting, especially to non-experts. People without the expertise often sketch an ideal picture of a fully automatic classification system of the sentiment of tweets. In practice, this is rather difficult, and the performance of unsupervised algorithms do not approach the performance of most supervised algorithms. Therefore I think the main focus of sentiment analysis should remain on supervised algorithms. When proven models of supervised algorithms are found, the focus may change to unsupervised algorithms.

Remarkable in the hand labeling was the huge amount of advertisement and news items. Filtering out those tweets should increase the robustness of a model. An important step is therefore to create a so-called two-stage model. In the first stage the neutral messages are removed, including the advertisements and news feeds. Then in the second stage the distinction between the positive and the negative tweets can be made. The neutral tweets don't influence the model, since they are filtered out in the first stage. But for training, the model still needs a lot of data, since the short messages do not cover a much data.

Appendices

Appendix A: Sentiment-word list R

#	word	subjectivity	class
445	avariciously	strongsubj	negative
446	avenge	strongsubj	negative
447	aver	strongsubj	positive
448	averse	strongsubj	negative
449	aversion	strongsubj	negative
450	avid	strongsubj	positive
451	avidly	strongsubj	positive
452	avoid	weaksubj	negative
453	avoidance	weaksubj	negative
454	awe	strongsubj	positive
455	awed	strongsubj	positive
456	awesome	strongsubj	positive
457	awesomely	strongsubj	positive
458	awesomeness	strongsubj	positive
459	awestruck	strongsubj	positive
460	awful	strongsubj	negative
461	awfully	strongsubj	negative
462	awfulness	strongsubj	negative
463	awkward	strongsubj	negative
464	awkwardness	strongsubj	negative

Table A.1: A fragment of the ”subjectivity” sentiment-word list in R.

Appendix B: Data Warehouse model

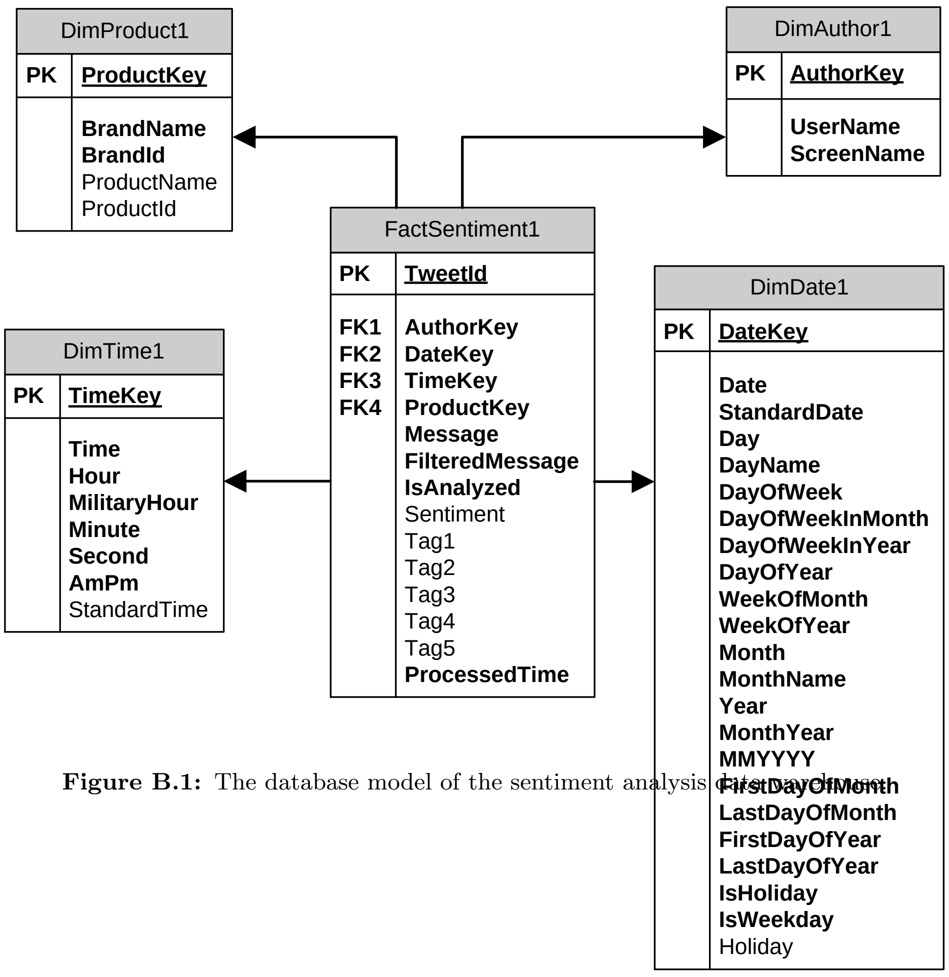

Figure B.1: The database model of the sentiment analysis data warehouse.

Appendix C: Brands and Mobile Phones

Brands	
1.	Accenture
2.	Apple
3.	Avanade
4.	Blackberry
5.	DELL
6.	Google
7.	HTC
8.	Microsoft
9.	Nokia
10.	Samsung
11.	Sony

Table C.1: the brands table which is used for the retrieval of the tweets.

Products	
1.	Blackberry Bold
2.	Blackberry Curve
3.	Blackberry Torch
4.	Galaxy Ace
5.	Galaxy S
6.	Galaxy S2
7.	HTC Radar
8.	HTC Sensation
9.	HTC Trophy
10.	HTC Wildfire
11.	iPhone
12.	Lumia
13.	Nexus
14.	Sony Xperia

Table C.2: The mobile phones table which is used for retrieval of the tweets.

Appendix D: Graphical User Interface of the User Application

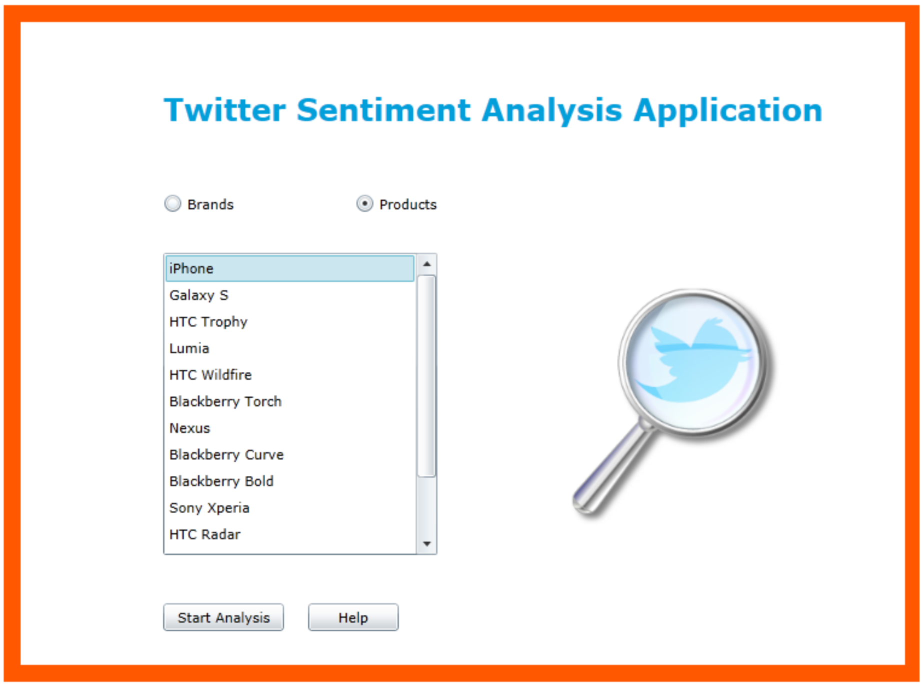

Figure D.1: Graphical user interface of the start screen of the Silverlight user application.

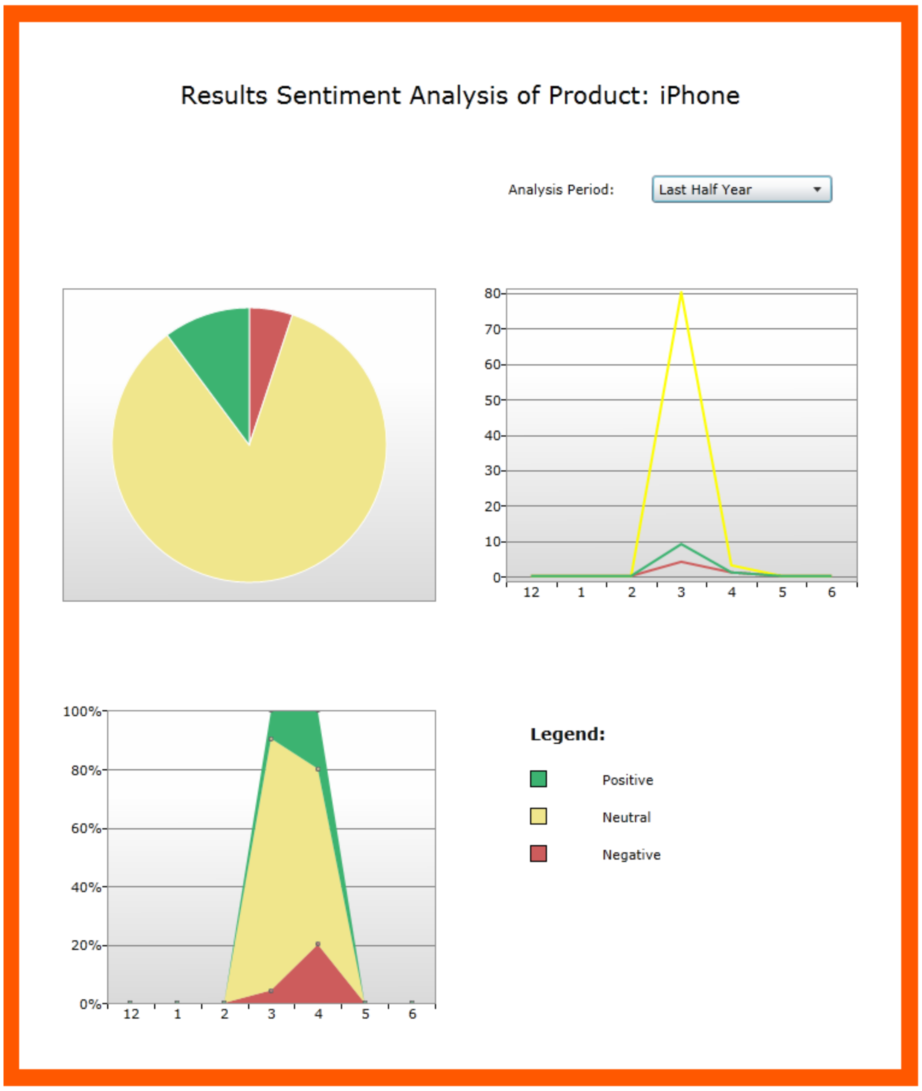

Figure D.2: Graphical user interface of the report screen of the Silverlight user application.

Glossary

classification the prediction process where a class label is assigned to each data point.

cross-validation a technique for assessing how the results of a classification model will generalize to an independent dataset.

document frequency the number of documents in which a word occurs.

feature extraction the reduction of the dimensionality of the data, by selecting features that describe the data the best.

feature vector set of features that is used to describe the data.

features relevant information that describe the data, to reduce the dimensionality of the data.

Gini index a mutual information measure that captures the inequality among values of a frequency distribution.

in-sample testing training and testing a model on entirely the same dataset.

k-fold cross-validation a cross-validation technique that divides the data in k folds, parts, and uses k times $k-1$ folds as training set and 1 fold as test set.

LDA linear discriminant analysis.

naive Bayes a probabilistic classifier based on applying Bayes' theorem together with the independence assumption.

out-of-sample testing training and testing on different set of data, to prevent overfitting.

overfitting incorporating random errors or noisy in a model instead of the underlying relationship.

PMI point mutual information.

polarity the sentiment class (positive, neutral or negative) to which a document belongs to.

POS part-of-speech.

QDA quadratic discriminant analysis.

sentiment analysis identifying and extracting subjective information from raw data.

summed term frequency summing all the term frequencies of a word across all documents.

Support Vector Machine a supervised learning algorithm that maps the data points in space and separates the associated classes with hyperplane which has a maximum distance to the classes.

SVM support vector machine.

term frequency a measure, which relies on the total number of occurrences of a term within a document.

term presence a binary measure, which relies on if a term is present within a document.

tweet a Twitter message, limited by the maximum number of 140 characters.

Twitter a social medium where users can share their interests, stories, ideas and opinions with a tweet.

Bibliography

[1] Kosei Abe. *R.NET*. 2011. URL: rdotnet.codeplex.com/.

[2] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. "Sentiment analysis of Twitter data". In: *Proceedings of the Workshop on Languages in Social Media*. Association for Computational Linguistics, 2011, pp. 30–38.

[3] Charu C. Aggarwal. "Towards Meaningful High-Dimensional Nearest Neighbor Search by Human-Computer Interaction". In: *In ICDE*. 2002, pp. 593–604.

[4] Alias-i. *LingPipe*. 2008. URL: http://alias-i.com/lingpipe.

[5] Luciano Barbosa and Junlan Feng. "Robust sentiment detection on Twitter from biased and noisy data". In: *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. COLING '10. Association for Computational Linguistics, 2010, pp. 36–44.

[6] Steven Bethard, Hong Yu, Ashley Thornton, Vasileios Hatzivassiloglou, and Dan Jurafsky. "Extracting Opinion Propositions and Opinion Holders using Syntactic and Lexical Cues". In: 2006, pp. 125–141.

[7] Albert Bifet and Eibe Frank. "Sentiment Knowledge Discovery in Twitter Streaming Data". In: *Discovery Science* 6332 (2010). Ed. by Bernhard Pfahringer, Geoff Holmes, and AchimEditors Hoffmann, pp. 1–15.

[8] Bernhard E. Boser and et al. "A training algorithm for optimal margin classifiers". In: *Proceeding of the 5th annual ACM workshop on computational learning theory*. ACM Press, 1992, pp. 144–152.

[9] Nicholas Carlson. *The Real History Of Twitter*. 2011. URL: http://articles.businessinsider.com/2011-04-13/tech/29957143_1_jack-dorsey-twitter-podcasting.

[10] William B. Cavnar and John M. Trenkle. "N-Gram-Based Text Categorization". In: *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*. 1994, pp. 161–175.

[11] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*. 2001.

[12] Joel Comm. *Twitter Power 2.0: How to Dominate Your Market One Tweet at a Time*. Revised edition. Wiley, Apr. 2010.

[13] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. 1st ed. Cambridge University Press, 2000.

[14] GebreKirstos Gebreselassie Gebremeskel. “Sentiment Analysis of Twitter Posts About News”. Sentiment Analysis. Feb. 2011.

[15] Robert Gentleman and Ross Ihaka. *The R Project for Statistical Computing.* 1997. URL: `http://www.r-project.org/`.

[16] Alec Go, Richa Bhayani, and Lei Huang. “Twitter Sentiment Classification using Distant Supervision”. In: *Processing* (2009), pp. 1–6.

[17] Torsten Grabs, Roman Schindlaue, Ramkumar Krishnan, Jonathan Goldstein, and Rafael Fernndez. “Introducing Microsoft StreamInsight”. Sept. 2009.

[18] Alexander Hinneburg, Charu C. Aggarwal, and Daniel A. Keim. “What Is the Nearest Neighbor in High Dimensional Spaces?” In: *Proceedings of the 26th International Conference on Very Large Data Bases.* VLDB ’00. Morgan Kaufmann Publishers Inc., 2000, pp. 506–515.

[19] Twitter Inc. *Twitter.* July 2006. URL: `http://www.twitter.com`.

[20] B.J. Jansen, M. Zhang, K. Sobel, and A. Chowdury. “Twitter power: Tweets as electronic word of mouth”. In: *Journal of the American Society for Information Science and Technology* (2009).

[21] Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. “Target-dependent Twitter Sentiment Classification.” In: *ACL.* Ed. by Dekang Lin, Yuji Matsumoto, and Rada Mihalcea. The Association for Computer Linguistics, 2011, pp. 151–160.

[22] Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. “Twitter Sentiment Analysis : The Good the Bad and the OMG !” In: *Artificial Intelligence* (2011), pp. 538–541.

[23] Vincent van Leeuwen, Kees van Nunen, Tim Harbers, and Durk Kingma. *SNTMNT.* 2011. URL: `http://www.sntmnt.com`.

[24] Bing Liu. “Sentiment Analysis and Subjectivity”. In: *Handbook of Natural Language Processing* 1 (2010). Ed. by N Indurkhya and F JEditors Damerau, pp. 1–38.

[25] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *An Introduction to Information Retrieval.* Cambridge University Press, 2008.

[26] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela Hung Byers. *Big data: The next frontier for innovation, competition, and productivity.* May 2011. URL: `http://www.mckinsey.com/insights/mgi/research/technology_and_innovation/big_data_the_next_frontier_for_innovation`.

[27] Mark Milian. *Why text messages are limited to 160 characters.* May 2009. URL: `http://latimesblogs.latimes.com/technology/2009/05/invented-text-messaging.html`.

[28] Claire Cain Miller. *Why Twitters C.E.O. Demoted Himself.* 2010. URL: `http://www.nytimes.com/2010/10/31/technology/31ev.html`.

[29] Melissa Miller. *43% of Twitter Users Access Twitter From a Mobile Phone.* Sept. 2011. URL: `http://blog.hubspot.com/blog/tabid/6307/bid/25026/43-of-Twitter-Users-Access-Twitter-From-a-Mobile-Phone-Data.aspx`.

[30] Alexander Pak and Patrick Paroubek. “Twitter as a Corpus for Sentiment Analysis and Opinion Mining”. In: *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC’10)*. European Language Resources Association (ELRA), 2010.

[31] Alexander Pak, Patrick Paroubek, De Paris-sud, Laboratoire Limsi-cnrs, and F Orsay Cedex. “Twitter Based System : Using Twitter for Disambiguating Sentiment Ambiguous Adjectives”. In: *Computational Linguistics* July (2010), pp. 436–439.

[32] Bo Pang and Lillian Lee. “Opinion Mining and Sentiment Analysis”. In: *Foundations and Trends in Information Retrieval* 2.1-2 (Jan. 2008), pp. 1–135.

[33] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. “Thumbs up? Sentiment Classification using Machine Learning Techniques”. In: *Language* 10.July (2002). Ed. by J Hajic and YEditors Matsumoto, pp. 79–86.

[34] John Rothfels and Julie Tibshirani. “Unsupervised sentiment classification of English movie reviews using automatic selection of positive and negative sentiment items”. June 2010.

[35] Niek J. Sanders. *Twitter Sentiment Corpus*. Version 0.2. 2011. URL: `http://www.sananalytics.com/lab/twitter-sentiment/`.

[36] Semiocast. *Arabic highest growth on Twitter English expression stabilizes below 40%*. 2011. URL: `http://semiocast.com/publications/2011_11_24_Arabic_highest_growth_on_Twitter`.

[37] Statconn. *statconnDCOM*. 2012. URL: `http://rcom.univie.ac.at/`.

[38] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. “Sentiment Strength Detection in Short Informal Text”. In: *Journal of the American Society for Information Science and Technology* 61.12 (2010), p. 21.

[39] Ken Thompson. “Regular Expression Search Algorithm”. In: *Communications of the ACM* 11.6 (1968), pp. 419–422.

[40] Peter D. Turney. “Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews”. In: *Computational Linguistics* pages.July (2002), p. 8.

[41] Helga Wiesenhofer, Martin Ebner, and Isidor Kamrat. “Is Twitter an Individual Mass Communication Medium?” In: *Proceedings of Society for Information Technology & Teacher Education International Conference 2010*. Ed. by David Gibson and Bernie Dodge. AACE, 2010, pp. 1712–1717.

[42] Theresa Wilson, Janyce Wiebe, and Rebecca Hwa. “Just how mad are you? Finding strong and weak opinion clauses”. In: *Proceedings Of The National Conference On Artificial Intelligence*. Vol. 04. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004, pp. 761–769.

[43] Peter Yared. *Why sentiment analysis is the future of ad optimization*. 2011. URL: `http://venturebeat.com/2011/03/20/why-sentiment-analysis-is-the-future-of-ad-optimization/`.

[44] Hong Yu and Vasileios Hatzivassiloglou. “Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences”. In: *Proceedings of the 2003 conference on Empirical methods in natural language processingVolume 10*. Ed. by Michael Collins and MarkEditors Steedman. Vol. 3. Association for Computational Linguistics, 2003, pp. 129–136.

[45] Taras Zagibalov and John Carroll. “Automatic Seed Word Selection for Unsupervised Sentiment Classification of Chinese Text”. In: *Proceedings of the 22nd International Conference on Computational Linguistics COLING 08* August (2008), pp. 1073–1080.