Smirking or Smiling Smileys?

EVALUATING THE USE OF EMOTICONS TO DETERMINE SENTIMENTAL MOOD

TOBIAS HINDERSSON AND ELIAS LOUSSEIEF
Smirking or Smiling Smileys?

Evaluating the Use of Emoticons to Determine Sentimental Mood

Tobias Hindersson    Elias Lousseief

Degree Project in Computer Science, DD143X
Supervisor: Richard Glassey
Examiner: Örjan Ekeberg

CSC, KTH
May 9, 2015
Abstract

Machine Learning classifiers are commonly used for the purpose of Sentiment Analysis. These classifiers use annotated training data from which they learn to predict the sentiment of texts, for example whether a text conveys a positive or a negative sentiment. In this thesis we compare the performance of two sources of training data for the purposes of sentiment classification on Twitter: (i) tweets annotated by hand of a fixed quantity (≈2000 tweets) and (ii) tweets annotated automatically by an emoticon heuristic of increasing quantity (from 2000 tweets to 1.6 million tweets). The performance of these training sets are evaluated by training commonly used classifiers (Naive Bayes, Support Vector Machines and Maximum Entropy) and comparing the classification accuracy of the different data sets on a test set annotated by hand. These tests are made with varying use of n-gram models (unigrams, bigrams, and a combination of both) and the varying use of a stop word filter. We show that while the hand-annotated training set performs well in equally sized training sets, the automatically annotated training set exceeds the accuracy of the hand-annotated training set in all test setups but one when 1.6 million automatically annotated tweets are used for training.
Sammanfattning

Maskinlärningsalgoritmer används ofta för att utföra analys av känslomässig inställning; sentimentsanalys. Dessa algoritmer använder annoterad träningsdata för att lära sig att klassificera texter efter exempelvis huruvida de speglar ett positivt eller negativt sentiment. I den här uppsatsen företas sentimentsanalys av data från Twitter varvid effektiviteten utvärderas med avseende på två typer av träningsdata: (i) en fix mängd tweets som annoterats för hand (cirka 2000 tweets) och (ii) olika mängder tweets som genomgått automatisk annotering av en heuristik baserad på emoticons (från 2000 till 1.6 miljoner tweets). Effektiviteten som träningsdata hos dessa dataset har utvärderats genom att träna vanliga maskinlärningsalgoritmer (Naive Bayes, Support Vector Machines och Maximum Entropy) vartefter jämförelser gjorts av hur väl de lyckats klassificera ett set med testdata som annoterats för hand. Testerna har gjorts med olika typer av n-gram (unigram, bigram samt kombinationen av dessa) samt valbar inkludering av ett filter med stoppord. I studien framkommer att träningsdata annoterad för hand presterar bra i jämförelse med annoteringar som gjorts heuristiskt förutsatt att dataseten är av samma storlek. Då omfattningen av den heuristiskt annoterade träningsdata växer förbättras dock förmågan till korrepta klassificeringar, och när storleken uppgår till 1.6 miljoner tweets ger användning av handannoterad träningsdata bättre resultat i endast ett fall av de testupptäckningar som används.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td></td>
</tr>
<tr>
<td>Sammanfattning</td>
<td></td>
</tr>
<tr>
<td>List of Figures</td>
<td>iii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>v</td>
</tr>
<tr>
<td>Terms and abbreviations</td>
<td>vii</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 A New Era</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Studying the Internet</td>
<td>1</td>
</tr>
<tr>
<td>2 Problem Statement</td>
<td>3</td>
</tr>
<tr>
<td>3 Background</td>
<td>5</td>
</tr>
<tr>
<td>3.1 Sentiment Analysis</td>
<td>5</td>
</tr>
<tr>
<td>3.1.1 A Subdomain of NLP</td>
<td>5</td>
</tr>
<tr>
<td>3.1.2 Problems and Criticism</td>
<td>6</td>
</tr>
<tr>
<td>3.1.3 Subdomains and Approaches</td>
<td>7</td>
</tr>
<tr>
<td>3.2 Sentiment Analysis with Machine Learning</td>
<td>8</td>
</tr>
<tr>
<td>3.2.1 Classifiers</td>
<td>9</td>
</tr>
<tr>
<td>3.2.1.1 Naive Bayes (NB)</td>
<td>9</td>
</tr>
<tr>
<td>3.2.1.2 Support Vector Machines (SVM)</td>
<td>9</td>
</tr>
<tr>
<td>3.2.1.3 Maximum Entropy (MaxEnt)</td>
<td>9</td>
</tr>
<tr>
<td>3.2.2 Feature Extraction</td>
<td>10</td>
</tr>
<tr>
<td>3.2.3 Data Mining</td>
<td>10</td>
</tr>
<tr>
<td>3.2.3.1 Collecting Data</td>
<td>10</td>
</tr>
<tr>
<td>3.2.3.2 Interpreting Data</td>
<td>11</td>
</tr>
<tr>
<td>4 Method</td>
<td>13</td>
</tr>
<tr>
<td>4.1 The Procedure</td>
<td>13</td>
</tr>
<tr>
<td>4.2 Resources</td>
<td>14</td>
</tr>
<tr>
<td>4.2.1 Data Sets</td>
<td>14</td>
</tr>
<tr>
<td>4.2.1.1 Stanford Twitter Corpus</td>
<td>14</td>
</tr>
<tr>
<td>4.2.1.2 STS-Gold</td>
<td>15</td>
</tr>
<tr>
<td>4.2.2 Software Libraries</td>
<td>15</td>
</tr>
<tr>
<td>4.2.3 Classifiers</td>
<td>15</td>
</tr>
</tbody>
</table>
# Implementation

## 4.3 Implementation

### 4.3.1 General Data Set Preprocessing

### 4.3.2 Runtime Input Preprocessing

#### 4.3.2.1 Filtering

#### 4.3.2.2 Stop Word Filter

#### 4.3.2.3 Stemming

### 4.3.3 N-grams

### 4.3.4 Tfidf Transformer

### 4.3.5 Classifier Settings

### 4.3.6 Test Setups

# Results

## 5 Results

### 5.1 Classifier Accuracy

#### 5.1.1 Output Table

#### 5.1.2 Diagrams

### 5.2 Additional Findings

#### 5.2.1 Effects of Stop Word Filter

##### 5.2.1.1 Naive Bayes

##### 5.2.1.2 Support Vector Machines

##### 5.2.1.3 Maximum Entropy

#### 5.2.2 Comparisons of Classifiers

# Discussion

## 6 Discussion

### 6.1 Training Set with Handmade Annotations or Not?

#### 6.1.1 Same Size Training Sets

#### 6.1.2 Different Size Training Sets

### 6.2 Additional Observations

#### 6.2.1 A Few Words on Stop Words

#### 6.2.2 Comparing the Classifiers

### 6.3 Comparison to Earlier Studies

### 6.4 Sources of Errors and Limitations

### 6.5 Future Research Topics

# Conclusion

## 7 Conclusion

## Bibliography
List of Figures

3.1 Determining the sentiment of a tweet using a classifier . . . . . . . . . . . . 8

5.1 NB classifier accuracy ................................................. 23
5.2 NB classifier accuracy with SW filter .................................. 23
5.3 SVM classifier accuracy ................................................ 23
5.4 SVM classifier accuracy with SW filter ............................... 23
5.5 MaxEnt classifier accuracy ............................................. 23
5.6 MaxEnt classifier accuracy with SW filter ............................ 23
5.7 Effect of adding the stop word filter on the NB classifier ......... 24
5.8 Effect of adding the stop word filter on the SVM classifier ..... 24
5.9 Effect of adding the stop word filter on the MaxEnt classifier ..... 25
5.10 Comparison of classifiers with unigrams ......................... 26
5.11 Comparison of classifiers with unigrams and SW filter ......... 26
5.12 Comparison of classifiers with bigrams .............................. 26
5.13 Comparison of classifiers with bigrams and SW filter .......... 26
5.14 Comparison of classifiers with both (unigrams and bigrams) .... 26
5.15 Comparison of classifiers with both (unigrams and bigrams) and SW filter 26
List of Tables

4.1 Emoticons used for automatic annotation \[3\] . . . . . . . . . . . . . . . . . 14
4.2 Software versions used for test runs. . . . . . . . . . . . . . . . . . . . . . 15
4.3 Used constructor parameter settings of Scikit-learn classifier classes. . . 19
4.4 List of variation parameters and possible values. . . . . . . . . . . . . . . 19

5.1 Classification accuracy for all parameters and classifiers. . . . . . . . . . . 22
List of Terms

Annotation
The (sentiment) class label associated with an input, generally used in the context of training data.

Classifier
A statistical machine learning algorithm for predicting or determining to which class an observation belongs.

Document
In the context of text classification, a document is a complete text, an article or a tweet for instance.

Emoticon
A sequence of text symbols that represent a facial expression, two prominent examples being “:)” and “:(”.

Feature
Some measurable property of the data that is being observed, in text classification, features are commonly the individual words, or the sequences of words from a text document.

Feature extraction
The process of extracting features from text for use in text classification.

Machine learning
A field within computer science that deals with learning algorithms.

N-gram
A sequence of N words or characters.

Natural language
A language that humans use to communicate. Contrasted with constructed languages such as computer programming languages.

Sentiment Analysis
The task of analysing the sentiment, emotions, attitudes, opinions and evaluations conveyed in text.


1 Introduction

1.1 A New Era

The Internet might be considered one of the most important inventions of the 20th century and it is safe to say that it has affected life for most of us. Its impact on society is hard to summarize and most businesses and activities seem to be affected by it in one way or the other. In the process, the World Wide Web has taken on many different roles. For example, Internet, and computers in general, have been used to rationalize pre-existing tasks by means of digitalization and improvement of communications. In other fields Internet has given rise to new possibilities and markets that did not exist before. The Web itself also constitutes an exceptional new phenomenon to study, both out of a technical viewpoint but also from a social perspective which is probably one of the areas in which the Internet has revolutionised our lives the most.

There is a more or less well-known principle stating that whatever gets uploaded to the Internet stays there forever. What is implied is that the Internet seems infinite and swallows (and saves) everything that comes its way. This fact, along with the recent habit of acting out a great deal of our social life online brings us closer to a view of the Internet as an unprecedented database of opinions, discussions and gossip. It turns out that the interest in studying this organic and constantly growing social documentation of humanity comes from many directions and is not always driven by a purely scientific incentive.

1.2 Studying the Internet

In general, traditional manual text processing can be problematic if the input is too large. For example, there is a chance that we miss things as focus drops, especially if the task is monotonous. Computers on the other hand can be automated to do a lot of tasks, but when it comes to human languages their analysis skills seem somewhat insufficient. Nevertheless, given our digital age and that the Internet now offers the possibility to study an almost infinite stream of information, it seems beneficial, let alone crucial, to learn how to let computers do this work.
In the field of Natural Language Processing, also referred to as NLP, computer processing of human languages is studied. Consequently, among other things, NLP deals with text processing and conclusions that might be drawn from it. In particular, to answer the question about what kind of sentiment is conveyed by a text we turn to Sentiment Analysis; a sub-division of NLP. An automation of this analysis task can take on different forms but is bound, in its final shape, to be represented by some sort of algorithm that processes an input text and then outputs a suggestion as to what sentiment is conveyed by the text. The output can be one of several different classes depending on what kind of sentiment classification we are looking for. Maybe we are interested in whether the text is ironic, mean, sweet or longing but in its simplest form, we might just want to determine whether the input is, in the broadest sense, positive or negative.

As previously said, different approaches might be used to achieve an algorithm like the above. One approach is to use Machine Learning (a field within computer science that deals with algorithms that learn) which basically involves two phases during which an algorithm, here also known as a classifier, is presented with training data and test data. The training data is pre-categorised, or annotated, and the task for the classifier is then to categorise the test data, hopefully with a high accuracy.
2 Problem Statement

Social media messages is a domain that Sentiment Analysis is attentive towards and that is also particular in many different aspects; messages are short, contain a certain kind of language and are often full of misspellings. Social media messages may also contain emoticons, less formally referred to as “smileys”. Some emoticons are sentimentally ambiguous whereas the subset of the most commonly used ones, at least in theory, has a clear symbolic meaning. Whether or not even these emoticons actually bear any contextual significance is not obvious, but given their inherent nature they would be very efficient to use as unambiguous sentiment markers.

Annotations made by hand is generally seen as the most accurate (even though the concept of accuracy might be questioned altogether, which will be discussed briefly in the Background section) and it is reasonable to assume that we ideally would like all data to be annotated by hand. This is however, as previously stated, problematic due to several reasons, but an approach while exploring how we can automate this task is to compare the efficiency of sentiment analysis techniques in terms of how well they perform compared to handmade annotations.

In this essay we have turned to the domain of Twitter with its 140-character limit messages. Here, the purpose of this thesis is to seek out how accurately, determined by the rate of correct classifications relative to corresponding handmade annotations, classifiers trained with tweets annotated by an emoticon heuristic perform sentiment analysis compared to classifiers trained with tweets annotated by hand. It is the assumption that classifiers trained with tweets annotated by hand will be superior and a result to the opposite would suggest that the used subset of emoticons, indeed, can serve as unambiguous overall sentiment markers in social media messages.
3 Background

3.1 Sentiment Analysis

3.1.1 A Subdomain of NLP

Natural Language Processing is a field that deals with computer processing of natural languages. By natural languages we mean languages spoken by humans, as opposed to computer languages which are not as complex and ambiguous as a natural language is. Typical applications are automated translations from one language to another, command processing in an input voice interface or the task of gathering information from a text. The common denominator is that a computer needs to process data in the form of a human language and gather information from it, as opposed to the conventional processing of programming languages which are more confined and less ambiguous.

One might think that the concept of NLP is relatively new but research has been conducted in this field since the early 1940s [5] at the dawn of the computer. Recently however, it has emerged as a very active topic of study in computer science, which has several reasons. First of all, we now live in an age when the world is getting computerized at a high pace; computers are available and used for a wide array of tasks in varying environments which in turn encourages software development. Furthermore, with the interconnectivity of the Internet, computers are now part of the infrastructure in most societies. As part of this, NLP can act as an aiding tool for trivial service tasks like translations and searching for misspellings, as well as more elaborate tasks such as bridging availability to people with special needs. As a science tool the evolution of NLP has also been necessitated by the fact that we now, through the Internet, have more data than ever before in history available for analysis. This data constitutes an unprecedented opportunity to document human chatter, with both commercial and scientific parties of interest as a result, and would be hard to process without automated mechanisms.

Sentiment Analysis is the subfield within Natural Language Processing that focuses on sentiment conveyed by text. Practical applications of Sentiment Analysis are for example finding out the general opinion on a product from a set of reviews or categorising blog
articles by some sentiment classification. The sentiment might consist of a series of attitudes or classes or boil down to just positive and negative (and possibly neutral) attitudes.

Sentiment Analysis and NLP are topics which lie in the cross section between several other, not necessarily closely related fields. Linguistics, psychology and computer science (artificial intelligence) are usually mentioned in the list of disciplines involved [5].

3.1.2 Problems and Criticism

Sentiment Analysis, though interesting and seemingly powerful, comes with a few potential problems on different levels. One of the major practical objections concerns the accuracy of the results. Some suggest that it is reasonable to be happy with a 30% accuracy even though 70% seems to be the more common figure [4]. Let us elaborate philosophically on the notion of accuracy; we shall find that there are at least three layers of problems surrounding the way natural languages are used.

The first problem concerns syntax and the fact that there are grammatically correct natural language sentences that can actually mean two different things depending on the context. The second problem concerns semantics and amounts to the fact that human interaction involves situations where the expressed semantics (even if they are syntactically unambiguous) are not always the implied ones. This problem raises the question as to how we should deal with sarcasm and jargong. The third problem relates to whether or not we are actually expressing the truth (even if it is syntactically and semantically unambiguous); it is not an uncommon idea to believe that we prefer to mask ourselves and share things and express ourselves online in a manner which makes us look better to others [4]. As if that would not be enough, two humans discussing the same unambiguous text might have vastly different ideas as to what sentiment is conveyed throughout it [4]. The task of determining sentiment then does not always seem to even have an objective “correct” answer, and a reasonable thought seems to be that we interpret texts differently depending on several factors, out of which two are cultural and personal background. Trusting computers with this daunting task then seems to be a questionable decision, but if we perform analysis we are likely to take its output into consideration. Given the uncertainty of the results, this is dangerous and misleading rather than helpful according to some voices, and hence analysis should not be done.

Another important aspect to bring up is the question of personal integrity. The general argument concerns the fact that Sentiment Analysis has lobbyists within marketing companies and large brands who wish to easily gather what individuals in the public domain think about their products. This might be considered indirect monitoring which is getting more and more common these days to the benefit of for example directed marketing. Some argue that such monitoring, in a broader sense, falls in a moral grey
zone and support their claim by the fact that it often happens in the background without the immediate knowledge of the user [4].

Another perspective is to consider the notion of companies analysing our online output as a passive kind of power and a possibility to make a difference. Critics, however, refer to companies as acting in favour of their own benefits and not in the interest of their customers. In the case where sentiment analysis would be used to determine adds and commercial content online for a user, it could be seen as a sort of discrimination. In effect, it limits our horizon towards new products with the rationale that we only take interest in familiar things, something that might prove wrong [4].

Other critics argue that the line between private and public is very thin in the social domain online [4]. Even if we assume that sentiment analysis is only done on data that is available to everyone, it does not necessarily mean that this data is publicly available for any purpose whatsoever. If we would know that our words are weighed in a way that would have impact on our future online lives, would we express ourselves differently? Some choose to defend the personal integrity whereas other support the opinion that everyone must inform themselves before using the Internet and must thus face the consequences [4].

### 3.1.3 Subdomains and Approaches

Sentiment Analysis can be divided into several subproblems, each with their own approaches and associated research. Bing, Liu [6, p. 4] defined the three domains: document-level, sentence-level and aspect-based. Document-level sentiment classification is the most simple and most well-researched domain in which a document, for example a product review or a social media post, is classified by the general opinion which it expresses. Sentence-level classification is similarly only concerned with a general sentiment, but of a single sentence rather than an entire document. Aspect-based sentiment classification rely on heavier natural language processing techniques to identify different parts of texts such as what object is the target of a sentiment, the different aspects of that object, the sentiment itself, and the holder of that sentiment among other potential parameters. Document-level and sentence-level sentiment classification in comparison are simpler because, to reiterate, only the general sentiment of the whole document or sentence is of interest requiring less preprocessing. One example of a document-level data source is Twitter where each tweet would constitute a document, and considering its 140-character limit on tweets sentence-level methods may also apply.

Within these domains, there are also different approaches as to how the analysis is accomplished. The primary two groups of methods being machine learning methods and lexicon-based methods. The lexicon-based approach typically relies on lexicons containing weighted sentiment associations of individual words which are then used as the basis for analysis. The machine learning approach relies on preannotated training data,
documents that have already been classified, in order to train classification algorithms that are then used to classify other documents. The advantage of the machine learning approach is that the training data can be optimized for specific topics. If the documents being analysed are camera reviews for instance, the training data could easily be assembled to properly reflect that topic, whereas the lexicons used in the lexicon-based approach are more generalized. Another advantage of machine learning is the amount of publicly available data that can be used for training.

### 3.2 Sentiment Analysis with Machine Learning

Machine learning is a problem solving technique that uses algorithms that learn from data in order to make predictions or decisions. The applications are many; from natural language processing and computer vision to stock market analysis. These methods can be divided into supervised learning and unsupervised learning, where the former uses annotated or labeled data and the latter uses data that is not annotated, but that is rather annotated by the algorithm as part of its process.

The algorithms specifically used for sentiment analysis are known as statistical classifiers and the type of learning is supervised, where the data is annotated either by hand or automatically by some heuristic prior to training. The first step in using a classifier is training, wherein features are extracted from annotated training data resulting in a vector of features that is then used to train the classifier. In the context of text classification, a feature might be a word, a sequence of words, or a sequence of characters. In other domains a feature might be some measurable property of the object being classified. Once a classifier is trained, it can be used for prediction. The document being tested for prediction will have its features extracted using the same extraction process as was used for the training data. The document can then be classified using its feature vector.

**Figure 3.1:** Determining the sentiment of a tweet using a classifier.
3 Background

3.2.1 Classifiers

Some commonly used classifiers for sentiment analysis are Naive Bayes (NB), Support Vector Machines (SVM), and Maximum Entropy (MaxEnt). These three classifiers in particular, and variations on them, have previously been shown to be efficient for sentiment analysis [3, 13]. An example of a well-performing variation was proposed by Wang et al. [19] who showed that an SVM classifier with NB features (referred to as NBSVM) could perform well. The same study also compared NB with SVM and found that NB typically performed better at short snippets sentiment tasks, whereas SVM performed better with longer documents. For the purpose of this thesis, we are only interested in the basic variations on these three classifiers.

3.2.1.1 Naive Bayes (NB)

The Naive Bayes classifier is a generative classifier that applies Bayes’ Theorem with a “naive” assumption of independence between features. NB lives up to its namesake by being simple compared to other classifiers, but despite its simplicity, it still manages to perform well [20]. NB is used in different forms for the purpose of text classification with two common variants being Multinomial NB and Multi-variate Bernoulli NB, each variant making a different assumption of the distribution of the feature vector. Multinomial Naive Bayes has been shown to outperform other NB variants on text documents [8, 9] and has previously been used for sentiment classification [3, 12].

3.2.1.2 Support Vector Machines (SVM)

The Support Vector Machines classifier is a discriminative classifier that is trained by finding the hyperplane that maximizes the distance between any mapped training data points of the given classes. A class is then decided on an input document by its relative mappings to that hyperplane. A notable property of the support vector classifier is its kernel function that determines the decision boundary. This kernel function is typically linear, polynomial, a radial basis function or a sigmoid function [16]. Determining which type of kernel function to use for text classification appears to be an open problem, with both polynomial [10] and linear [3, 19] kerneling functions having been previously used in research.

3.2.1.3 Maximum Entropy (MaxEnt)

Maximum Entropy is a technique for estimating probability distributions of data, which is commonly used for certain NLP tasks like language modeling, part-of-speech tagging, and text classification [11]. Maximum entropy is based on the idea of having a preference
towards a uniform distribution in an effort to maximize the entropy of that distribution, while also satisfying any given constraints. Using Maximum Entropy for binary classification is equivalent to using logistic regression to find the distribution over the classes of the training data [3].

3.2.2 Feature Extraction

In order to optimize the accuracy of the classifiers it is necessary to consider the feature extraction process. The most simple feature extraction process would be to convert a text document to a feature vector where each feature is a word from the document. An alternative to this approach is to group consecutive words together as features. These sequences of words are commonly referred to as (word) n-grams, where n is the number of consecutive words included in a sequence. For example, the sentence “Monkey tasered in France.”, would be represented by the bigrams (n=2) “Monkey tasered”, “tasered in”, and “in France”. Bigrams have in some cases been shown to be slightly more efficient at sentiment classification than unigrams (n=1) [19]. Go et al. [3] also found that combining unigrams and bigrams for feature selection performed well with NB and MaxEnt classifiers. There is also a variant of the n-gram model that uses single characters, rather than words as n-gram tokens, an approach which has been shown to be viable when analysing noisy or irregular data [1].

Reducing the amount of features prior to training could improve the performance of classifiers. A commonly used preprocessing method for tweets is to filter URLs, usernames and common Twitter keywords (such as “RT”) [2] [12]. Another method involves the removal of so-called stop words [12] which are words that bear no specific sentiment, sometimes very common, that might be excluded to not mislead the classifier during processing. Examples of stop words include particles like “a”, “an” and “the”, and common pronouns. Yet another preprocessing step that can be used to reduce the amount of the features is stemming or lemmatization; algorithms which convert words to their root stem or lemma respectively. A lemma is the dictionary form of a word and a stem is the part of a word that is common to inflectional variants that are formed by adding affixes. For example, the lemma of the word “better” is “good”, and both the stem and the lemma of “walking” is “walk”. The benefits of using a stemmer or a lemmatizer is that the amount of features can be reduced.

3.2.3 Data Mining

3.2.3.1 Collecting Data

Having representative training data is needed for classifiers to be efficient. In the domain of Twitter, collecting data for sentiment classification amounts to compiling lists of
tweets and annotating them by sentiment classes. One method of gathering this data requires individuals to annotate tweets by hand; a process that, as previously mentioned in section 2, is associated with some problems. Despite its shortcomings, this method is presumably still the most accurate there is. The alternative is to use some heuristic to automatically determine the sentiment of tweets. One such method was proposed in 2009 [3] and further tested in 2010 [2, 12]. Emoticons were used to automatically annotate tweets into groups of “positive” and “negative”. This heuristic puts some limitations on the available training data as only tweets containing emoticons can be used. The benefit of using this method is that it allows for greater volumes of data to be processed than is otherwise pragmatically possible by hand. It also allows for the data to be tailored towards whatever topic is being targeted by the analysis and provides the means to keep it up to date, as sentiment can sometimes shift. This method, sometimes called Distant Supervision [3], has previously performed well but its classification accuracy remains to be formally compared to that of hand-annotated training data.

There are certain properties of training data sets that could impact the performance of classifiers. Saif et al. [15] found a correlation between classification accuracy and the data sparsity and vocabulary size of training data. They also found that there is a strong correlation between the vocabulary size and the quantity of tweets in a data set. Increasing the quantity of tweets when training could therefore improve the accuracy.

### 3.2.3.2 Interpreting Data

An understandable objection to sentiment analysis might be that not all texts contain sentiment [4]. Even though it is a popular source of data, this holds true in the case of Twitter as tweets are often restatements of news headlines or just plain links. Among social media, Twitter therefore seems like a less than ideal choice of source material with Facebook messages or regular text messages presumably conveying sentiment more often. However, these sources of data are not open for public use whereas Twitter, at least partially is; the bottom line being that input data is different in different domains and might have to be adapted thereafter.

As previously mentioned in the section 3.1.2, it is quite possible that two persons disagree on the sentiment of a text. This might be a problem if we are compiling our own training set of tweets annotated by hand; a problem that might at least be partially solved. One example is the Twitter sentiment corpus STS-Gold which was annotated by three graduate students, each student annotating the same tweets [15]. In this case, whenever there was disagreement among the students, the tweet in question was excluded from the final data set to account for potential interpretation issues.
4 Method

This chapter begins with an overview description of the process out of a non-technical viewpoint. We then proceed to go into detail on different parts of the setup required as well as provide explanations to some of the choices that have been made along the way.

4.1 The Procedure

An investigative comparison will take place. The goal is to determine how the performance of a classifier is affected by the use of training data annotated by an emoticon heuristic, as opposed to training data annotated by hand. Twitter has been chosen as the domain of interest and therefore tweets will be the input document type. Three main data sets will be chosen with different roles during this process; two training sets and one test set, all of which will be different. One of the training sets will be annotated by hand and the other will be annotated based on an emoticon heuristic. The test set will be annotated by hand (for verification purposes) and the success of the classifier will be determined as an accuracy percentage originating from comparing its classifications to the handmade annotations of the test set. As target classes are concerned, a simple strategy will be used to grant focus to the main question and the classes in use will be limited to only “positive” and “negative”.

It is desirable that the comparison does not only yield incidental and non-representative results, which would be more likely with few or only one setup while testing. As a countermeasure we will decide on multiple test setups which shall all be used during the process. Even though this does not entirely eliminate the risk that results are of incidental kind, it reduces it.

The result of this comparison will be the compilation of the results from the different setups used. The final goal is then to try and draw conclusions from these output results as a response to the main question about whether annotations based on an emoticon heuristic can measure up against handmade annotations. At this time, we will also elaborate on the effects of altering different test parameters, if any.
4.2 Resources

This section lists external resources in use throughout the comparison procedure without going into too much detail as to how these are used.

4.2.1 Data Sets

Acquiring data sets can be a large part of the kind of investigation at hand. Twitter was chosen as the source primarily for two reasons, which might in turn be connected. First of all, most social media communities store their users’ data securely in private, and it is not available to the general public. The contrary is of course a necessary condition in this case. Twitter on the other hand not only has user data publicly available thanks to the fact that tweets in general can be entirely public, but also has an API allowing people to access and download public tweets. This surely has previously encouraged people to use tweets as an input document type which in part explains why a large part of the previous studies similar to this one has been based on tweets. Subsequently, the latter also being the second reason for choosing Twitter; being able to compare results to previous studies is of great value.

Ultimately, instead of using the Twitter API to compile new sets of tweets, data is obtained from publicly available preannotated sets previously used in research. Some of these data sets are categorised according to, but not limited to, the classes “positive” and “negative”. Tweets in categories other than the two mentioned have, as a consequence, been excluded from all parts of our process; trainingwise as well as testwise.

4.2.1.1 Stanford Twitter Corpus

The Stanford Twitter Corpus was created in 2009 and consists of several data sets [3]. The largest data set contains 1.6 million tweets annotated automatically using an emoticon heuristic. A subset of the emoticons was mapped to a positive sentiment whereas another subset was mapped to a negative sentiment. Some emoticons were left out. The heuristic then processed the tweets and made annotations according to the pre-determined mapping. After the procedure, the emoticons were removed from the individual tweets before making the set available to the public. This training set, both as a whole and in parts, takes on the role of the training set with automated annotations based on an emoticon heuristic in this comparison process.

Table 4.1: Emoticons used for automatic annotation [3].

<table>
<thead>
<tr>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>:)</td>
<td>:(</td>
</tr>
<tr>
<td>:-)</td>
<td>:-(</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>:D</td>
<td>=)</td>
</tr>
</tbody>
</table>

The Stanford Twitter Corpus also contains a smaller data set with 359 hand-annotated tweets out of which 177 are negative and 182 positive. This was initially used to test
the Stanford Twitter Corpus and is also used here playing the role of the test set. The test set has not had any emoticons removed and may thus contain such.

4.2.1.2 STS-Gold

Contains 3000 tweets that were selected from the 1.6-million-tweet data set of the Stanford Twitter Corpus, making this data set a subset of the latter, albeit with different annotations [15]. Needless to say, this means that this set has also had its emoticons removed. Each tweet in the set has been annotated by three graduate students using the classes “positive”, “negative”, “neutral”, “mixed” and “other”. Whenever the graduate students would disagree on the sentiment of a tweet, it was removed from the data set.

STS-Gold contains 2205 tweets; 1402 negative and 632 positive, the rest being annotated as “neutral”, “mixed” or “other”. Throughout the comparative process to come, it is used as the training set with tweets annotated by hand.

4.2.2 Software Libraries

Several choices of software libraries exist and it is beyond the scope of this thesis to list them all. Two libraries are considered, both for use with the Python programming language. The NLTK library [7] was first considered but the desire to use a Multinomial Naive Bayes classifier then resulted in the use of the Scikit-learn [14] library instead. Scikit-learn also provides several other important tools and the final software setup contains mostly Scikit-learn classes with NLTK providing certain services such as a stop words corpus, a tokenizer and a stemmer.

Table 4.2: Software versions used for test runs.

<table>
<thead>
<tr>
<th>Software</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>3.4.2 and 2.7.1</td>
</tr>
<tr>
<td>NLTK</td>
<td>3.0.1</td>
</tr>
<tr>
<td>Scikit-learn</td>
<td>0.16.1</td>
</tr>
</tbody>
</table>

4.2.3 Classifiers

Three classifiers are used; Naive Bayes (NB), Support Vector Machines (SVM) and Maximum Entropy (MaxEnt). The Naive Bayes classifier is a Multinomial Naive Bayes, and the SVM is used with a linear kernel. The reason why these particular classifiers were chosen is that the paper which established the Stanford Twitter Corpus also uses them [3]. This allows for recreation of their results, as well as the ability to compare them to the results acquired with hand-annotated data sets.
4.3 Implementation

This section describes the implementation necessary to perform the given comparison, including how resources are used and integrated into our testing environment.

The overall implementation amounts to the processing of the input and the setup of a so-called Pipeline: a Scikit-learn software library class consisting of the different parts needed to perform an analysis run. Processing of the input involves both reading the input from files as well as running it through our own preprocessing. The input is then passed on to the Pipeline which consists of three integral parts with the actual classifier algorithm being the final one. The first part performs feature extraction on behalf of the software library and simply counts word appearances and turns them into a representation that is easier for the classifier to process. The middle step of the Pipeline is a transformer that performs a normalization of the word count vectors provided by the previous step.

4.3.1 General Data Set Preprocessing

The Stanford Twitter Corpus training set that was annotated by an emoticon heuristic contains 1.6 million tweets whereas the STS-Gold collection only contains 2000 tweets. While it is easier to facilitate training with a large training set with an automated annotation method, it would still not be entirely right to compare a classifier resulting from a training session with 1.6 million input documents to one which had only trained with 2000 input sequences. A reasonable idea would be to shrink the former to the size of the latter. However, because of the aforementioned principle, it would still be interesting to see if and how the results change depending on the size of the training set with automated annotations. The 1.6-million-tweet data set from the Stanford Twitter Corpus is therefore divided into several subsets so that a comparison can take the size of the automatically annotated training data set into account. The smaller size subsets are put together through the use of a program that randomly chooses a given amount of tweets from the large corpus guaranteeing a disposition of tweets in the two target classes differing in no more than 5 percentage points. Final unit sizes are 2000, 10,000, 100,000 and 1,600,000. Henceforth, these subsets along with the entire original set will be referred to as SF2k, SF10k, SF100k and SF1.6M, the final one being the unaltered original set.

4.3.2 Runtime Input Preprocessing

Tweets are read from a text file and preprocessed before they are fed to the Scikit-learn classes. It is worth mentioning that all characters are later converted to lower case in the
first step of the Pipeline \cite{17} and that this subsequently is not necessary to implement in the preprocessing external to the library classes.

### 4.3.2.1 Filtering

As previously mentioned in section 4.2.1.2, some of the tweets from the STS-Gold training set are annotated with more target classes in mind than just the two used in this thesis ("positive" and "negative"). Tweets with such annotations are continuously excluded from all parts of the comparison process. Additionally, three preprocessing filters were used:

- **User filter**: Replaces Twitter usernames (beginning with "@") with the token "USER".
- **URL filter**: Replaces URLs (beginning with "http", "https" or "ftp") with the token "URL".
- **Repeating characters filter**: Replaces three or more consecutive identical characters with two occurrences of the same character.

### 4.3.2.2 Stop Word Filter

All setups are tested both with and without a stop word filter as a parameter of variety. In the Pipeline, a list of stop words is added to its first step where the class performing feature extraction, the CountVectorizer, simply ignores words present in the given wordlist. Additionally, the use of a stop word filter in this case also removes the resulting "USER" and "URL" tokens that might remain from the previous filtering process.

The stop words used in the filter originates from NLTK \cite{7}, which contains an English stop word corpus of 127 words to which the "USER" and "URL" tokens were added.

### 4.3.2.3 Stemming

The snowball stemmer, included in NLTK \cite{7}, is used to convert individual words to their stem before training.

### 4.3.3 N-grams

During the comparison process unigrams, bigrams and a combination of both are used. Unigrams and bigrams are standard and the use of both of them is included for the sake of completeness. As a bonus, the combination of the two has proven efficient with Naive Bayes and Maximum Entropy classifiers as mentioned in section 3.2.2.
The use of character grams was considered but rejected at this point due to too far-going implications on the rest of the setup.

4.3.4 Tfidf Transformer

A tfidf transformer is a standard device, not previously mentioned, that processes vectors of word counts and normalizes them in different ways (turns them into tf or tf-idf format). It processes the input vectors and alters the counts, or weights, so that they better reflect the impact of different words. This is done in relation to the overall frequency of a word during a session and results in the significance of a certain word being greater the less frequent it is [18]. The tfidf transformer can also perform further normalization so that all weights are in a zero-to-one interval. A tfidf transformer is one of the standard steps of the Scikit-learn Pipeline class and given its function, its presence is intuitively motivated and is thus included as a Pipeline step in the process to come.

4.3.5 Classifier Settings

The classifiers have several parameters that can be altered for the purpose of optimization. Optimization is done against the hand-annotated training set STS-Gold with the unigram model and stop words. The motivation behind this is first of all that it is reasonable to tune a classifier to correspond as well as possible to handmade annotations, since these are as close to a subjective “truth” as a given annotator can come. Furthermore, the use of stop words is reasonable due to the fact that tweets are short messages that should suffer even more than longer texts from having irrelevant word pollution. The use of STS-Gold then finally justified using the unigram model, since that setup is likely to return fewer and more common features than bigrams would, and subsequently yield a better result with the test set.

Classifier parameters are, for each classifier, changed in a heuristic way until no better results can be achieved. The results from these tunings are hence used as reference values against which the success of test runs with classifiers trained with the other training sets is determined.
Table 4.3: Used constructor parameter settings of Scikit-learn classifier classes.

**Naive Bayes:**
MultinomialNB(alpha=1.0, fit_prior=False, class_prior=None)

**Support Vector Machines:**
LinearSVC(penalty='l2', loss='hinge', dual=True, tol=0.0001, C=8,
        multi_class='ovr', fit_intercept=False, intercept_scaling=1, class_weight=None,
        verbose=0, random_state=None, max_iter=1000)

**Maximum Entropy:**
LogisticRegression(penalty='l2', dual=False, tol=0.0001, C=5.5, fit_intercept=True,
           intercept_scaling=1, class_weight=None, random_state=None,
           solver='liblinear', max_iter=100, multi_class='ovr', verbose=0)

4.3.6 Test Setups

Test setups are established by altering and combining the following parameters:

Table 4.4: List of variation parameters and possible values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifier</td>
<td>{NB</td>
</tr>
<tr>
<td>Stop words</td>
<td>{yes</td>
</tr>
<tr>
<td>N-grams</td>
<td>{unigrams</td>
</tr>
<tr>
<td>Training set</td>
<td>{STS-Gold</td>
</tr>
</tbody>
</table>
5 Results

This chapter contains the results of all the test setups as described in section 4.3.6. In section 5.1.1 the unfiltered data from the testing is presented in a table to give a full overview of the results. This is followed by section 5.1.2 presenting multiple diagrams of the classifier accuracy on the different training sets, with two diagrams for each classifier: one diagram with the stop word filter and one without the stop word filter.

We also present additional findings in sections 5.2 not central to the problem statement, but nonetheless interesting secondary results. In section 5.2.1 the effects of adding the stop word filter on the different classifiers and training sets are shown in bar charts. In section 5.2.2 the classifier accuracy is shown with diagrams grouped by n-gram model and usage of stop word filter.

5.1 Classifier Accuracy

Parameters were varied according to section 4.3.6 and all possible combinations were tried with the results presented in the table in section 5.1.1. The listed accuracy is defined as:

\[
\frac{N(\text{correct classifications})}{N(\text{classifications})}
\]
5.1.1 Output Table

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Classifier accuracy</th>
<th>Naive Bayes</th>
<th>SVM</th>
<th>MaxEnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set</td>
<td>N-gram</td>
<td>SW filter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STS-Gold</td>
<td>unigram</td>
<td>no</td>
<td>0.69638</td>
<td>0.78273</td>
</tr>
<tr>
<td>SF2k</td>
<td>unigram</td>
<td>no</td>
<td>0.74095</td>
<td>0.69359</td>
</tr>
<tr>
<td>SF10k</td>
<td>unigram</td>
<td>no</td>
<td>0.75487</td>
<td>0.75487</td>
</tr>
<tr>
<td>SF100k</td>
<td>unigram</td>
<td>no</td>
<td>0.80501</td>
<td>0.74373</td>
</tr>
<tr>
<td>SF1.6M</td>
<td>unigram</td>
<td>no</td>
<td>0.78830</td>
<td>0.78552</td>
</tr>
<tr>
<td>STS-Gold</td>
<td>unigram</td>
<td>yes</td>
<td>0.75766</td>
<td>0.80223</td>
</tr>
<tr>
<td>SF2k</td>
<td>unigram</td>
<td>yes</td>
<td>0.75209</td>
<td>0.70752</td>
</tr>
<tr>
<td>SF10k</td>
<td>unigram</td>
<td>yes</td>
<td>0.75209</td>
<td>0.73538</td>
</tr>
<tr>
<td>SF100k</td>
<td>unigram</td>
<td>yes</td>
<td>0.80223</td>
<td>0.72702</td>
</tr>
<tr>
<td>SF1.6M</td>
<td>unigram</td>
<td>yes</td>
<td>0.81616</td>
<td>0.74652</td>
</tr>
<tr>
<td>STS-Gold</td>
<td>bigram</td>
<td>no</td>
<td>0.67967</td>
<td>0.66852</td>
</tr>
<tr>
<td>SF2k</td>
<td>bigram</td>
<td>no</td>
<td>0.62953</td>
<td>0.60167</td>
</tr>
<tr>
<td>SF10k</td>
<td>bigram</td>
<td>no</td>
<td>0.71309</td>
<td>0.70195</td>
</tr>
<tr>
<td>SF100k</td>
<td>bigram</td>
<td>no</td>
<td>0.76602</td>
<td>0.73816</td>
</tr>
<tr>
<td>SF1.6M</td>
<td>bigram</td>
<td>no</td>
<td>0.84680</td>
<td>0.79109</td>
</tr>
<tr>
<td>STS-Gold</td>
<td>bigram</td>
<td>yes</td>
<td>0.65460</td>
<td>0.63788</td>
</tr>
<tr>
<td>SF2k</td>
<td>bigram</td>
<td>yes</td>
<td>0.63231</td>
<td>0.61838</td>
</tr>
<tr>
<td>SF10k</td>
<td>bigram</td>
<td>yes</td>
<td>0.67967</td>
<td>0.66852</td>
</tr>
<tr>
<td>SF100k</td>
<td>bigram</td>
<td>yes</td>
<td>0.76602</td>
<td>0.71588</td>
</tr>
<tr>
<td>SF1.6M</td>
<td>bigram</td>
<td>yes</td>
<td>0.80223</td>
<td>0.75209</td>
</tr>
<tr>
<td>STS-Gold</td>
<td>both</td>
<td>no</td>
<td>0.62396</td>
<td>0.75487</td>
</tr>
<tr>
<td>SF2k</td>
<td>both</td>
<td>no</td>
<td>0.69638</td>
<td>0.71588</td>
</tr>
<tr>
<td>SF10k</td>
<td>both</td>
<td>no</td>
<td>0.76880</td>
<td>0.76602</td>
</tr>
<tr>
<td>SF100k</td>
<td>both</td>
<td>no</td>
<td>0.81058</td>
<td>0.76880</td>
</tr>
<tr>
<td>SF1.6M</td>
<td>both</td>
<td>no</td>
<td>0.83565</td>
<td>0.81337</td>
</tr>
<tr>
<td>STS-Gold</td>
<td>both</td>
<td>yes</td>
<td>0.69916</td>
<td>0.77994</td>
</tr>
<tr>
<td>SF2k</td>
<td>both</td>
<td>yes</td>
<td>0.72145</td>
<td>0.69916</td>
</tr>
<tr>
<td>SF10k</td>
<td>both</td>
<td>yes</td>
<td>0.78830</td>
<td>0.74652</td>
</tr>
<tr>
<td>SF100k</td>
<td>both</td>
<td>yes</td>
<td>0.83844</td>
<td>0.75487</td>
</tr>
<tr>
<td>SF1.6M</td>
<td>both</td>
<td>yes</td>
<td>0.82730</td>
<td>0.78273</td>
</tr>
</tbody>
</table>
5.1.2 Diagrams

Red colored lines indicate the accuracy with the STS-Gold as training set.

**Figure 5.1:** NB classifier accuracy

**Figure 5.2:** NB classifier accuracy with SW filter

**Figure 5.3:** SVM classifier accuracy

**Figure 5.4:** SVM classifier accuracy with SW filter

**Figure 5.5:** MaxEnt classifier accuracy

**Figure 5.6:** MaxEnt classifier accuracy with SW filter
5.2 Additional Findings

5.2.1 Effects of Stop Word Filter

This section summarizes how performance was affected when stop words were added to each and every one of the test setups.

5.2.1.1 Naive Bayes

Figure 5.7: Effect of adding the stop word filter on the NB classifier

5.2.1.2 Support Vector Machines

Figure 5.8: Effect of adding the stop word filter on the SVM classifier
5.2.1.3 Maximum Entropy

Figure 5.9: Effect of adding the stop word filter on the MaxEnt classifier
5.2.2 Comparisons of Classifiers

This section compares the accuracy of the classifiers using the same n-gram models.

Red colored lines indicate the accuracy with the STS-Gold as training set.

**Figure 5.10:** Comparison of classifiers with unigrams

**Figure 5.11:** Comparison of classifiers with unigrams and SW filter

**Figure 5.12:** Comparison of classifiers with bigrams

**Figure 5.13:** Comparison of classifiers with bigrams and SW filter

**Figure 5.14:** Comparison of classifiers with both (unigrams and bigrams)

**Figure 5.15:** Comparison of classifiers with both (unigrams and bigrams) and SW filter
6 Discussion

6.1 Training Set with Handmade Annotations or Not?

To answer the research question given in section 2 in a simple and straight manner, the results from section 5.1 are examined in detail. Same size training sets are chosen as a starting point. Initial discussion focuses on the comparison between the classification accuracy of the hand-annotated training set and the accuracy of the automatically annotated training set of the same size. The size of the automatically annotated training set being compared is then increased to see how this comparison evolves as the quantity of automatically annotated tweets changes.

6.1.1 Same Size Training Sets

Since the hand-annotated training set STS-Gold contains about 2000 tweets, the aforementioned initial comparison is between SF2k and STS-Gold. These two are used in a number of test setups in which they are compared in terms of classification accuracy, with all other parameters unaltered. If any test setup yields some result where a classifier trained with SF2k has a higher accuracy than the same test setup with STS-Gold, then an instance is found that contradicts initial expectations of the hand-annotated set being superior in terms of classification accuracy. Out of the 18 test setups observed in Table 5.1, not considering the larger quantity training sets, this was the case for one third of the setups. This goes against the initial assumption to some extent, even though it hardly proves it wrong. The following observations were made:

- The Naive Bayes classifier notably managed to perform better when trained with SF2k rather than STS-Gold with the unigram model and without the stop word filter, and when using the combination of unigrams and bigrams, both with and without the stop word filter. STS-Gold performed better than SF2k in 3 out of 6 test setups when using the Naive Bayes classifier.
- The Support Vector Machines classifier performed better when trained with STS-Gold on every test setup.
• The Maximum Entropy classifier performed better when trained with SF2k with the bigram model and with the stop word filter, and when using the combination of unigrams and bigrams, both with and without the stop word filter. STS-Gold performed better than SF2k in 3 out of 6 test setups when using the Maximum Entropy classifier.

The above vaguely indicates that when using equal size training sets, a classifier trained with hand-annotated STS-Gold has an advantage over training sets annotated automatically by an emoticon heuristic. An interesting fact is that both Naive Bayes and Maximum Entropy, without exception, performed better when trained with SF2k than when trained with STS-Gold when the combination of unigrams and bigrams was used. Even though less convincing than predicted, the initial assumption of hand-annotated training sets being superior to heuristically annotated training sets stands.

6.1.2 Different Size Training Sets

When increasing the quantity of tweets used for classifier training, there is a clear trend observable in all of the plots in section 5.1.2. Even though there are a few exceptions, the classification accuracy increases with the quantity of automatically annotated tweets. This indicates that although emoticons can reflect sentiment in an ambiguous way in smaller sample sizes, when accumulated in larger training sets they become more accurate in their representation of sentiment.

The largest training set, SF1.6M, perform better than STS-Gold in 17 out of 18 test setups as seen in table 5.1. As previously mentioned in section 4.3.1, it might seem unfair to compare the results of training sets of such different sizes, but it is important to consider that obtaining large quantities of training data is trivial with the emoticon heuristic. Thus it is justified to take large training sets with automated annotations into account as we are looking to answer the research question.

SF1.6M used with the combination of unigrams and bigrams resulted in the highest accuracy in four of the six diagrams shown in section 5.1.2. A notable exception is the setup where the bigram model was used with Naive Bayes and without the stop word filter as seen in Figure 5.1. In this setup, SF1.6M managed to get the the highest overall accuracy with 84.7%.

The only test setup in which STS-Gold performed better than SF1.6M was with the Support Vector Classifier with the unigram model and the stop word filter. In this setup STS-Gold achieved 80.2% accuracy compared to the 74.6% accuracy of SF1.6M in the same setup.
6.2 Additional Observations

Several additional observations can be made, both from examining the primary diagrams in section 5.1.2 and from examining the additional findings in section 5.2. For example, by looking at the primary diagrams in section 5.1.2 we see that the bigram model, relative to the unigram model and the combination of unigrams and bigrams, perform poorly with small training sets. A possible explanation for this is that the bigrams of certain tweets are not recognized well enough by the classifier when trained with a small training set. Because of the larger feature space which bigrams hold over unigrams, more data is needed to ensure that classifiers with bigram features can identify the sentiment of tweets than is needed for a classifier with unigram features. The problem therefore occurs when the classifier has not learnt enough to be able to make good predictions for the tweets in the test set, due to the small size of the training data.

6.2.1 A Few Words on Stop Words

Each classifier was tested with five training sets and each of these training sets was tested with the unigram model, the bigram model and the combination of unigrams and bigrams. Additionally, all of these variations were tested with and without the use of the stop word filter. This yields 45 test setups with the stop word filter and 45 without, as seen in Table 5.1. Out of the 45 test setups with the stop word filter, only 20 of them gave better results than the corresponding setups without the filter, as observed in Figures 5.7, 5.8 and 5.9. This means that the remaining 25 test setups were unchanged or even worse with the stop word filter. The latter was in fact the case for most of them.

This observation might seem peculiar considering that the addition of the stop word filter is intended to remove redundant features in an effort to improve the classification accuracy. A possible reason for this expectation not being observed in the results might be either incidental or explained by the stop word filter incorrectly removing certain words that do carry sentiment, reducing the accuracy of the classifiers.

6.2.2 Comparing the Classifiers

From looking at the diagrams in section 5.2.2 we see that Support Vector Machines generally performed the worst among the three classifiers. Naive Bayes is the most accurate classifier even though Maximum Entropy also performed quite well. Considering the document domain of Twitter and its short messages, Naive Bayes was expected to perform better than Support Vector Machines; the former being seen as good with short input documents and the latter being better with long input documents [19].
6.3 Comparison to Earlier Studies

The test setups using the automatically annotated training sets achieved comparable results to the paper which first established the emoticon heuristic [3] which could be considered a baseline for this thesis. Of note is the accuracy of the Naive Bayes classifier with the bigram model, as seen in figure 5.1, which exceeded the accuracy of the combined model of unigrams and bigrams despite performing worse in the original study. The tests otherwise performed as expected given this baseline, minor differences being explained by the addition of the snowball stemmer, the inclusion of the stop word filter and individual parameter tuning of classifiers.

6.4 Sources of Errors and Limitations

The fact that we have attempted multiple setups indicates, but does not rule out, that our results are not incidental. However, there are multiple implicit and explicit limitations inherent in the comparison process made and each one of them can potentially be a source of error. The following is a primary list of factors that might be used differently to achieve other results:

- Classifiers might be tuned differently and against a different training set with changed parameters as a probable outcome.
- Other training and test data might be used altogether. In particular, the used test set is rather small which means that the results are less general than they could be.
- Accessories, such as stemmer, tfidf transformer and stop words, might be excluded or used differently.

6.5 Future Research Topics

The following is a list of suggestions on further research that could extend the findings of this report.

- Reconstructing the process while taking into account what is said in section 6.4. The purpose would be to try to make new findings by altering these parameters.
- Attempt to determine whether there is a constraint in terms of document count, as to when training sets with heuristic annotations can compete with hand-annotated training sets. This report has established that the former generally need to be more comprehensive but has failed to establish if there is a relationship between the sizes of the sets, and if so, what the relationship might be.
• Explore whether there is a difference between Twitter and other social media and if this method is applicable there as well.

• Explore whether findings apply to heuristic versus handmade annotations in general, even outside domains where emoticons are used. This would require the deployment of other heuristics.

• Determine whether or not time is relevant to the findings of this report. There is a chance that the use of emoticons have changed over time which would then, in turn, alter the results from training classifiers with sets annotated heuristically. Investigating this could lead into an entirely new direction where the use of emoticons would be the main theme.

• The heuristically annotated training data used in this report could be annotated with a new emoticon heuristic. This new heuristic could take all emoticons into account, as opposed to only a subset of them as the current one does. The new heuristic could be crafted so that it handles cases where multiple emoticons are present simultaneously in a document. It is unknown what the heuristic used to annotate part of the training data in this study handles these situations.

• This study could be remade with character n-grams instead of word n-grams. Worth mentioning in connection to this is that misspellings, which social media are often full of, would have less significance in such a setup.

• Focus more on training data than we have done in this report. Preprocessing and classifiers have been more highlighted here whereas there are reasons to believe that the initial choice of training data might be just as important in terms of the final results.
7 Conclusion

The question whether training data with automated annotations by an emoticon heuristic can compete with training data with handmade annotations can be answered in different ways. If the sizes of the training sets are required to be the same, then our tests indicate that annotations made by hand will yield classifiers with better overall performance in terms of accuracy, even if the results were not as overwhelming as predicted. However, with automated annotations, it is simple to acquire large sets of data for training purposes. The undertaken comparison then suggests that as the training sets with automated heuristic annotations grow, they will gradually perform better and better in comparison to training data with handmade annotations. In fact, often they will end up performing better than a relatively small hand-annotated training set as soon as they are large enough. Training sets with handmade annotations can analogously and on the contrary to the case with automated annotations, be assumed to be small, making the conclusions of this result even less presumptuous.

Furthermore, trends point in the same direction despite the use of multiple different combinations of a set of parameters. This adds more weight to the presented conclusions even if there is still, and will always be on occasions such as this, a chance that the yielded results are incidental and merely a result of the specific set of parameters or resources used.

From a bigger perspective, this study suggests that a subset of common emoticons are often used in a way such that the emoticon and the content of the message correctly reflect the same sentiment. More exact, it is not uncommon to use emoticons in ambiguous and contradictory ways, but in the long run, the use of emoticons seem to converge towards being used to reflect sentiment as presumably intended. This subsequently justifies the use of emoticons as sentiment markers for automated heuristic annotations in training data as long as the input document frequency is large enough. What is meant by “large enough” is relative to several factors, but “as large as possible” is a good guideline. In situations where this needs to be more exact, a test setup with the purpose to get an idea about these values are recommended.

It is furthermore reasonable to believe that this method would perform similarly in other domains, including other social media websites, where emoticons are used. The use of a different heuristic could presumably extend the method to other domains as well.
Bibliography


