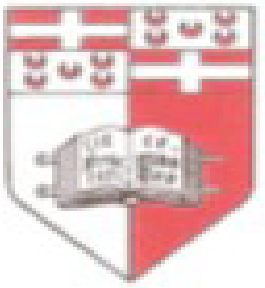
Sentiment Analysis of Twitter posts about news

Gebrekirstos Gebremeskel
M.Sc HLST
May 2011

University of Malta

Department of Computer Science and Artificial Intelligence

Masters Thesis



Gebrekirstos Gebremeskel

A Sentiment Analysis of Twitter Posts about news

Supervisors :
Mike Rosner, University of Malta
Gosse Bouma, University of Groningen

Programme: M.Sc HLST
Specialization: Erasmus Mundus Masters Program in Language and Communication Technologies (LCT)

May 2011

Sentiment Analysis of Twitter Posts About News

Name: GebreKirstos Gebreselassie Gebremeskel

Date:28-02-2011

Acknowledgments

All the work from beginning to end in this thesis is done in a rehabilitation center for tuberculosis patients in the Netherlands. In February 2010, I was diagnosed with tuberculosis after a serious of examinations which finally culminated with two minor operations on my stomach. After about three weeks in Hospital, I was transferred to Beatrixoord, a rehabilitation center. After taking medication for about three months, I was told that I had a multi-drug resistant (MDR) tuberculosis and that I had to start new medication. Also I was told that the treatment will take between a year and half and two years. That information by itself was a shock to me at that time. However as time went by, other types of problems slowly invaded my body and mind. I suffered from many side effects and bodily pains. Problems with my scholarship, tuition fees, residence permit, insurance, etc. surfaced. But the most difficult problem was the psychological problems I endured. The thesis is written during the most difficult time of my life hitherto, and in this difficult time, many people extended their help and some left indelible marks of kindness in my heart. I thought I should take this opportunity to express my gratitude to all those people who directly and indirectly extended their help during this difficult time in general and during the writing of this thesis in particular. Below I want to list some names, but I want to stress that it does not mean that those of you whose names are not stated did little help and do not deserve mention here. It is just that it is practically too long a list to include all names here.

The hospital people: I want to thank all the doctors and nurses of Martini hospital and Beatrixoord who took care of me when I was seriously ill. Particularly, I thank Dr. Van Atlena, my doctor. He usually gestures; he speaks rarely, smiles rarely and jokes rarely. Whenever, he comes to me or I go to him, he never makes me feel bad. I thank all the nurses who deal with the daily changing moods and sometimes belligerent behaviors of patients, yet maintain their high-level professionalism. However, I want to express my special gratitude to those nurses who went extra miles to help me, comfort me, and encourage me. I thank Tom for how he treated me the first day I was brought to Beatrixoord and for his subsequent checks and encouragements, Sibrie for she is simply a mom, Toos for always being angelic to me, Ellen for whenever she saw I was down, she came and sat by me, listened to me and encouraged me, Marleen

for when I was sad and down, she tapped me on my back and said "promise you wont be sad again", Nikita for always being warm and cheering, Maria and Annie for their friendly manners, Patricia, Paul, Mariella, Johan and Sonja.

The academic people: I thank all the people in the program who helped me during this difficult time. I would like to thank my advisor, Dr. Gosse Bouma, for his patience with me all the way from topic selection to the finishing of this thesis. I thank him for his help with all the administrative procedures, the insurance and tuition problems. I also would like to thank Professor Gisela, and Dr. Mike Rosner

Friends and special people I want to express my special thanks to all my friends who visited me, sat on the side of my bed, comforted and encouraged me, joked and cheered me up. My thanks go to Tadesse Simie, Henock Tilanhun and Dawit Tibebu for your countless visits, help, talking to family back home. I thank Henock for carrying out all the paper works related to my studies on my behalf and Tadesse for being a close friend. My special thanks go to Juan, my Spanish friend for his constant encouraging words, visits, presents, movies. He is amazing. Also my thanks go to Olga and Nadine for your visits and smiling balloons for bringing me a small laptop and an Internet stick so I do not get bored, Nick and Eliza for your visits and prayers for me, Addisu Abebe for his visists and encouragement. I want to thank Ruth, Jery, the Wondmu family and the Bekilla family for caring and bringing me Ethiopian food.

My special thanks go to Mrs. Angeliek van Hout. She has been exactly a mother to me. I thank her for the many walks together with her children, dinners for me and for my friends, introducing me to some amazing people, presents, and encouragements. I thank her so much. I also thank Ita de Regt for the many visits and paper works,taking me for walks, horse rides and for coffee and tea at her home. I thank Marco and Astrid for being so kind and generous, for their amazing offer to stay with them and finish my thesis in case there was financial problem. They are just amazing people to me. Thank you all. It would have been impossible for me to recover and finish my study without you all during this difficult time.

Contents

Chapter 1

Introduction

1.1 Motivation

"They called it the jasmine revolt, Sidi Bouzid revolt, Tunisian revolt... but there is only one name that does justice to what is happening in the homeland: Social media revolution", said a Tunisian revolutionary as quoted in Huffingtonpost[1]. Internet activist Wael Ghonim did not hesitate to call the Egyptian uprising "facebook revolution" in his interview with CNN journalist. The uprisings in North Africa and the Middle East have been dubbed "Twitter revolution", "Facebook revolution", or "social media revolution". Hyperbole aside, social media definitely contributed to the uprisings from beginning to end. Both facebook and Twitter had multiplying effect throughout the uprisings. While Facebook played a very important role at creating groups, organizing and mobilizing at the initial stages, Twitter helped the revolution gain momentum by recording events as they happen and spreading news to the world. The sentiment carried by Facebook or Twitter posts definitely inspired and galvanized people for more action.

With the proliferation of Web 2 applications such as microbloging, forums and social networks, there came reviews, comments, recommendations, ratings and feedbacks generated by users. The user generated content can be about virtually anything including politicians, products, people, events, etc. With the explosion of user generated content came the need by companies, politicians, service providers, social psychologists, analysts and researchers to mine and analyze the content for different uses. The bulk of this user generated content required the use of automated techniques for mining and analyzing since manual mining and analysis are difficult for such a huge content. A case in point of the bulk user-generated content that have been studied are blogs (Bautin et al., 2008) and product/movie (Turney, 2002) reviews. Today, traditional news outlets have gone online or have an online version of their news allowing news

[1] http://www.huffingtonpost.com/firas-alatraqchi/tunisias-revolution-was-t_b_809131.html

consumers to express their opinions about the news article, something that was almost impossible in the traditional printed-media. New online news outlets have also been created, many of them allowing user commenting. And this, like in blogs and product/movie reviews, has brought about a bulk of user generated content, and with it, the need by news outlets, analysts and researchers to know how news consumers have been affected. Among others, there is interest in whether news consumers reacted in a negative or a positive way.

One of the most popular microbloging platforms is Twitter. Twitter has become a melting pot for all - ordinary individuals, celebrities, politicians, companies, activists, etc. Almost all the major news outlets have Twitter account where they post news headlines for their followers. People with Twitter accounts can reply to or retweet the news headlines. Twitter users who have an account can also post news headlines from any other news outlet. When people post news headlines on Twitter, reply to news posts, or retweet news posts, it is possible that they can express their sentiment along with what they are posting, retweeting or replying to. The interest of this thesis is in what people are saying about news in Twitter. Specifically, the interest is in determining the sentiment of Twitter posts about news. This interest was inspired by a local IT company called Heeii in Groningen, the Netherlands. The company develops Twitter applications for web browsers.

1.2 Theoretical framework and Research Question

Sentiment analysis or opinion mining, as it is sometimes called, is one of many areas of computational studies that deal with opinion-oriented natural language processing. Such opinion-oriented studies include, among others, genre distinctions, emotion and mood recognition, ranking, relevance computation, perspectives in text, text source identification, and opinion-oriented summarization (Pang and Lee, 2008). Sentiment analysis (or more specifically, sentiment polarity analysis) can be defined as the mapping of text to one of the labels (elements) taken out of a finite predefined set or placing it on the continuum from one end to the other (Pang and Lee, 2008). The elements of the predefined set are usually 'negative' and 'positive', but they can also be any other elements such as 'relevant' and 'irrelevant', 'in favour of' or 'against', or other more than two elements such as 'negative', 'positive', 'neutral' or a range of numbers such as from 1 to 10.

All problems in opinion-oriented text analysis can be formulated as problems of text classification (Pang and Lee, 2008). Thus sentiment analysis can be redefined as classification of a textual entity into one of the elements or another of the predefined set described above. Sentiment analysis can be done at different levels - document, section, paragraph, sentence, or phrase levels. Most sentiment analysis work is done at a document level, but there are works on sentiment analysis at phrase and at sentence levels (Wilson et al., 2005, Hatzi-

vassiloglou and McKeown, 1997) Studies have shown that sentiment analysis at phrase level proved to be more difficult than at higher levels (Wilson et al., 2005).

Twitter post about news are short messages belonging to the lower levels of text. Mostly, they belong to the sentence level. The maximum number of characters that a Twitter post can have is 140. Because of the limit on the length of a Twitter post, Twitterers use abbreviated and slang language to overcome the limit. Moreover, there is a whole different language in the social media world that does not exist in traditional texts. For example, there are some words that are common to social media such as lol, rofl, wtf, emoticons (frowns, and smiles), etc. There are also some Twitter-specific languages such as RT (for retweet), #(hashtag), @(at), etc.

While Twitter sentiment analysis can be formulated as a classification problem like any other sentiment analysis, it is considerably different from other sentiment analysis studies because of the nature of the posts. There is a vast literature on general sentiment analysis and several Twitter-specific sentiment analysis researches[2]. All of the relevant literature on sentiment analysis of Twitter messages are term-based (see Pak and Paroubek (2010), Go et al. (2009), Barbosa and Feng (2010)). They first extract Twitter posts that contain a certain term and analyzes the sentiment of these extracted tweets. There is no Twitter-specific sentiment analysis study I know so far that attempted to examine Twitter posts about any specific domain such as news. There is not also any study that attempted to show whether context plays a role or not in the determination of the sentiment of a Twitter post.

1.2.1 Research question

Sentiment analysis of Twitter posts about news sets out to computationally analyze the sentiment of tweets about news. It attempts to find novel ways of extracting tweets about news from Twitter, and it examines whether context plays a role in the determination of the sentiment of tweets about news. Sentiment analysis of tweets about news is a study to do a three-classed (positive, negative, neutral) classification. It does experiments with different operations, feature selections, instance representations, and learning algorithms and recommends the combination that gives improved performance. I believe this research question makes a good departure from the existing sentiment analysis studies.

1.3 Importance, aims and outcomes

Generally, sentiment analysis helps companies and service providers to know the sentiment of their customers and users and to accordingly tailor their products and services to the needs of customers and users. Wright (2009) claims that "for many businesses, online opinion has turned into a kind of virtual currency that

[2]The literature on both general sentiment analysis and Twitter-specific sentiment analysis will be presented in chapter 2.

can make or break a product in the marketplace". But it is also of paramount interest for scientists such as social psychologists since it opens a window to tap into the psychological thinking and reactions of online communities. It helps to study and understand the general mind-state of communities at a particular time with regard to some issue. It can also be of use for political analysts in predicting election results during the campaign stage of political elections. Policy makers can also use it as input during policy making.

When news items are made available to communities, they certainly have an impact on their readers. It would be important for news agencies or other interested ones to know how news consumers have been affected or impacted by reading the news articles. Knowing news consumer reactions to news can also be used in decision making by politicians and policy makers. News agencies can also use consumer reactions to better their news coverage, presentation and content. A case in point is the coverage by Al jazeera of the uprisings in Egypt where they were collecting the tweets from the uprisings and broadcasting them. Similarly, CNN was constantly presenting tweets in their coverage of the uprisings.

The objective of this thesis is to find techniques to automatically determine the sentiment of tweets posted in reaction to news articles. It specifically aims to develop a classifier that can be used to automatically classify news comments into positive, negative or neutral. At the same time, it will show whether Twitter posts about news can be understood without the contents of the news. Another important outcome will be that it will show techniques of obtaining Twitter posts about news.

1.4 Organization of the thesis

The thesis is organized as follows. There are 4 more chapters. In chapter 2, methodology and approaches will be presented. Review of literature and methodological approaches will be presented. Under review of literature, general sentiment analysis, difficulty of sentiment analysis, feature engineering, classification, sentiment analysis techniques and sentiment analysis of Twitter messages in particular will be discussed. Under methodological approaches, the literature is interpreted, evaluation is briefly discussed and best methods and approaches are identified.

In chapter 3, data collection and preprocessing will be discussed. Under this chapter Twitter APIs, test data, training data, and Twitter posts about news will be discussed. Chapter 4, discusses experiment and results. Here, keyword-based classification, supervised approaches, and preprocessing of data are presented. Under supervised approaches, results for several algorithms using different representation formats are presented. Finally, in chapter 5, analysis and conclusions are provided. Factors affecting performance, evaluations, challenges and conclusions are provided in this chapter.

Chapter 2

Methodology and Approaches

2.1 Review of Literature

There exists substantial research on the subject of sentiment analysis. Although there were some previous attempts to study sentiment analysis, most active research on the area came with the explosion of user-generated content in social media, discussion forums, blogs and reviews. Since most sentiment analysis studies use or depend on machine learning approaches, the amount of user-generated content provided unlimited data for training. The research on sentiment analysis so far has mainly focused on two things: identifying whether a given textual entity is subjective or objective, and identifying polarity of subjective texts (Pang and Lee, 2008).

In the following sections, I will try to review literature on both general and Twitter-specific sentiment analysis with a main focus on those which are relevant to the thesis work at hand. I will first present general ideas about sentiment analysis such as what makes sentiment analysis more difficult than other text classification tasks, approaches, machine learning techniques to sentiment analysis, and feature engineering. Pang and Lee (2008) present a comprehensive review of the literature written before 2008. Most of the material on general sentiment analysis is based on their review. After the general sentiment analysis review, I will discuss Twitter-specific sentiment analysis.

2.1.1 General Sentiment analysis

Sentiment analysis has been done on a range of topics. For example, there are sentiment analysis studies for movie reviews (Pang et al., 2002), product reviews (Dave et al., 2003, Na et al., 2004), and news and blogs (Godbole et al., 2007, Bautin et al., 2008). Below some general sentiment analysis concepts are discussed.

2.1.2 Difficulty of sentiment analysis

Research shows that sentiment analysis is more difficult than traditional topic-based text classification, despite the fact that the number of classes in sentiment analysis is less than the number of classes in topic-based classification (Pang and Lee, 2008). In sentiment analysis, the classes to which a piece of text is assigned are usually negative or positive. They can also be other binary classes or multi-valued classes like classification into 'positive', 'negative' and 'neutral', but still they are less than the number of classes in topic-based classification. The main reason that sentiment analysis is more difficult than topic-based text classification is that topic-based classification can be done with the use of keywords while this does not work well in sentiment analysis (see Turney, 2002).

Other reasons for difficulty are: sentiment can be expressed in subtle ways without any ostensible use of negative words; it is difficult to determine whether a given text is objective or subjective (there is always a fine-line between objective and subjective texts); it is difficult to determine the opinion holder (example, is it the opinion of the author or the opinion of the commenter); there are other factors such as dependency on domain and on order of words (Pang and Lee, 2008). Other factors that make sentiment analysis difficult are that it can be expressed with sarcasm, irony, and/or negation.

2.1.3 Classification and approaches

As elaborated in the introduction chapter, sentiment analysis is formulated as a text-classification problem. However, the classification can be approached from different perspectives suited to the work at hand. Depending on the task at hand and perspective of the person doing the sentiment analysis, the approach can be discourse-driven, relationship-driven, language-model-driven, or keyword-driven. Some of the perspectives that can be used in sentiment classification are discussed briefly in the subsequent subsections.

2.1.3.1 Knowledge-based approach

In this approach, sentiment is seen as the function of some keywords. The main task is the construction of sentiment discriminatory-word lexicons that indicate a particular class such as positive class or negative class. The polarity of the words in the lexicon are determined prior to the sentiment analysis work. There are variations to how the lexicon is created. For example, lexicons can be created by starting with some seed words and then using some linguistic heuristics to add more words to them, or starting with some seed words and adding to these seed words other words based on frequency in a text (see Turney, 2002). For some domains of tasks, there are publicly available discriminatory word lexicons for use in sentiment analysis. http://twitrratr.com/ and http://www.cs.pitt.edu/mpqa/ are two examples. http://twitrratr.com/ provides sentiment lexicons for Twitter sentiment analysis.

2.1.3.2 Relationship-based approach

Here the classification task can be approached from the different relationships that may exists in or between features and components. Such relationships include relationships between discourse participants, relationships between product features. For example, if one wants to know the sentiment of customers about a product brand, one may compute it as a function of the sentiments on different features or components of it.

2.1.3.3 Language models

In this approach the classification is done by building n-gram language models. Presence or frequency of n-grams might be used. In traditional information retrieval and topic-oriented classification, frequency of n-grams is shown to give better results. Usually, the frequency is converted to TF-IDF to take term's importance for a document into account. However, Pang et al. (2002), in sentiment classification of movie reviews found that term-presence gives better results than term frequency. They indicate that uni-gram presence is more suited for sentiment analysis. But a bit later than Pang et al. (2002), Dave et al. (2003) found that bi-grams and tri-grams worked better than uni-grams in sentiment classification of product reviews.

2.1.3.4 Discourse structures and semantics

In this approach, discourse relation between text components is used to guide the classification. For example in reviews, the overall sentiment is usually expressed at the end of the text (Pang et al., 2002). As a result the approach to sentiment analysis, in this case, might be discourse-driven in which the sentiment of the whole review is obtained as a function of the sentiment of the different discourse components in the review and the discourse relations that exist between them. In such an approach, the sentiment of a paragraph that is at the end of the review might be given more weight in the determination of the sentiment of the whole review. Semantics can be used in role identification of agents where there is a need to do so. for example "Manchester beat Liverpool" is different from "Liverpool beat Manchester".

2.1.4 Sentiment analysis techniques

Using one or a combination of the different approaches in subsection 2.1.4, one can employ one or a combination of machine learning techniques. Specifically, one can use unsupervised techniques, supervised techniques or a combination of them.

2.1.4.1 Unsupervised Techniques

In unsupervised technique, classification is done by a function which compares the features of a given text against discriminatory-word lexicons whose polarity

are determined prior to their use. For example, starting with positive and negative word lexicons, one can look for them in the text whose sentiment is being sought and register their count. Then if the document has more positive lexicons, it is positive, otherwise it is negative. A slightly different approach is done by Turney (2002) who used a simple unsupervised technique to classify reviews as recommended (thumbs up) or not recommended (thumbs down) based on semantic information of phrases containing an adjective or adverb. He computes the semantic orientation of a phrase by mutual information of the phrase with the word 'excellent' minus the mutual information of the same phrase with the word 'poor'. Out of the individual semantic orientation of phrases, an average semantic orientation of a review is computed. A review is recommended if the average semantic orientation is positive, not recommended otherwise.

2.1.4.2 Supervised Techniques

The main task here is to build a classifier. The classifier needs training examples which can be labeled manually or obtained from a user-generated user-labeled online source. Most used supervised algorithms are Support Vector Machines (SVM), Naive Bayes classifier and Multinomial Naive Bayes. It has been shown that Supervised Techniques outperform unsupervised techniques in performance (Pang et al, 2002). Supervised techniques can use one or a combination of approaches we saw above. For example, a supervised technique can use relationship-based approach, or language model approach or a combination of them. For supervised techniques, the text to be analyzed must be represented as a feature vector. The features used in the feature vector are one or a combination of the features in 2.1.4.4 subsection.

2.1.4.3 Combined Techniques

There are some approaches which use a combination of other approaches. One combined approach is done by (Liu et al., 2004). They start with two word lexicons and unlabeled data. With the two discriminatory-word lexicons (negative and positive), they create pseudo-documents containing all the words of the chosen lexicon. After that, they compute the cosine similarity between these pseudo-documents and the unlabeled documents. Based on the cosine similarity, a document is assigned either positive or negative sentiment. Then they use these to train a Naive Bayes classifier.

Another combined approach is done by Melville et al. (2009). They used what they call "a unified framework" that allows one to "use background lexical information in terms of word-class associations, and refine this information for specific domains using any available training examples" (Melville et al., 2009, p. 1275). They proposed and used a different approach which they called polling multinomial classifier, which is another name for multinomial Naive Bayes. They used manually-labeled data for training, unlike (Liu et al., 2004). They report that their experiments with data reveal that their incorporation of lexical knowledge improves performance. They state that they obtained better

performance with their approach than approaches using lexical knowledge or training data in isolation, or other approaches that use combined techniques. There are also other types of combined approaches that are complimentary in that different classifiers are used in such a way one classifier contributes to another (Prabowo and Thelwall, 2009).

2.1.4.4 Feature Engineering

Since most of sentiment analysis approaches use or depend on machine learning techniques, the salient features of text or documents are represented as feature vector. The following are the features used in sentiment analysis.

Term presence or term frequency : In standard Information retrieval and text classification, term frequency is preferred over term presence. However, Pang et al. (2002), in sentiment analysis for movie reviews, show that this is not the case in sentiment analysis. Pang et al. claim that this is one indicator that sentiment analysis is different from standard text classification where term frequency is taken to be a good indicator of a topic. Ironically, another study by Yang et al. (2006) shows that words that appear only once in a given corpus are good indicators of "high-precision" subjectivity.

Term can be either uni-grams, bi-grams or other higher-order n-grams. Whether uni-grams or higher-order n-grams give better results is not clear. Pang et al. (2002) claim that uni-grams outperform bi-grams in movie review sentiment analysis, but Dave et al. (2003) report that "bi-grams and tri-grams give better product-review polarity classification".

POS (Part of speech) Tags : POS is used to disambiguate sense which in turn is used to guide feature selection (Pang and Lee, 2008). For example, with POS tags, we can identify adjectives and adverbs which are usually used as sentiment indicators (Turney, 2002). But, Turney himself found that adjectives performed worse than the same number of uni-grams selected on the basis of frequency.

Syntax and negation: Collocations and other syntactic features can be employed to enhance performance. In some some short text (sentence-level) classification tasks, algorithms using syntactic features and algorithms using n-gram features were found to give same performance (Pang and Lee, 2008). Negation is also an important feature to take into account since it has the potential of reversing a sentiment (Pang and Lee, 2008).There are attempts to model negation for better performance (Das and Chen, 2001, Na et al., 2004). Na et al. (2004) report 3% accuracy improvement for electronics product reviews by handling negation. Note also that negation can be expressed in more subtle ways such as sarcasm, irony and other polarity reversers.

2.1.5 Twitter-specific sentiment analysis

There are some Twitter-specific sentiment analysis studies. Twitter sentiment analysis is a bit different from the general sentiment analysis studies because Twitter posts are short. The maximum number of characters that are allowed in Twitter is 140. Moreover Twitter messages are full of slang and misspellings (Go et al., 2009). Almost all Twitter sentiment classification is done using machine learning techniques. Two good reasons for the use of machine learning techniques are 1) the availability of huge amount of Twitter data for training, and 2) that there is test data which is user-labeled for sentiment with emoticons (avoiding the cumbersome task of manually annotating data for training). Read (2005) showed that the use of emoticon for training is effective. Below I present some of the most relevant studies on Twitter sentiment analysis.

A Twitter sentiment analysis study by Go et al. (2009) does a two-classed (negative and positive) classification of tweets about a term. Emoticons (for positive ':)', for negative ':(') were used to collect training data from Twitter API. The training data was preprocessed before it was used to train the classifier. Preprocessing included replacing user names and actual URLs by equivalence classes of 'URL' and 'USERNAME' respectively, removing repeated letters to 2 (huuuuuuungry to huungry), and removing the query term. To select useful uni-grams, they used such feature selection algorithms as frequency, mutual information, and chi-square method. They experiment with three supervised techniques: multinomial Naive Bayes, maximum entropy and support vector machines (SVM). The best result, accuracy of 84%, was obtained with multinomial Naive Bayes using uni-gram features selected on the basis of their MI score. They also experimented with bi-grams, but accuracy was low. They claim the reason for this low accuracy is data spareness. Incorporating POS, and negation into the feature vector of uni-grams does not also improve results.

The above experiment does not recognize and handle neutral tweets. To take into account neutral tweets, they collected tweets about a term that do not have emoticons. For test data, they manually annotated 33 tweets as neutral. They merged these two datasets with the training data and test data used in the above two-classed classification. They trained a three-classed classifier and tested it, but the accuracy was very low, 40%.

Another study by Barbosa and Feng (2010) used a two-phased approach to Twitter sentiment analysis. The two phases are: 1) classifying the dataset into objective and subjective classes (subjectivity detection) and 2) classifying subjective sentences into positive and negative classes (polarity detection). Suspecting that the use of n-grams for Twitter sentiment analysis might not be a good strategy since Twitter messages are short, they use two other features of tweets: meta information about tweets and syntax of tweets. For meta-info, they use POS tags (some tags are likely to show sentiment, eg. adjectives and interjections) and mapping words to prior subjectivity (strong and weak), and prior polarity (negative, positive and neutral). The prior polarity is reversed when a negative expression precedes the word. For tweet syntax features, they use #(hashtag, @(reply), RT(retweet), link, punctuations, emoticons, capital-

ized words, etc. They create a feature set from both the features and experiment with machine learning technique available in WEKA. SVM performs best. For test data, 1000 tweets were manually annotated as positive, negative, and neutral. The highest accuracy obtained was 81.9% on subjectivity detection followed by 81.3% on polarity detection.

A very related study to this thesis was done by Pak and Paroubek (2010). They did a three-classed (positive, negative, neutral) sentiment analysis on Twitter posts. They collected negative and positive classes using emoticons (for positive: ":-)", ":)", "=)", ":D" ,etc and for negative: ":- (", ":(", "=(", ";(", etc.). For the neutral class, they took posts from Twitter accounts of popular news outlets (the assumption is news headlines are neutral).

After the data collection, they did some linguistic analysis on the dataset. They POS tagged it and looked for any differences between subjective (positive and negative) and objective sentences. They note that there are differences between the POS tags of subjective and objective Twitter posts. They also note that there are difference in the POS tags of positive and negative posts. Then they cleaned the data by removing URL links, user names (those that are marked by @), RT (for retweet), the emoticons, and stop words. Finally they tokenized the dataset and constructed n-grams. Then they experimented with several classifiers including SVM, but Naive Bayes was found to give the best result. They trained two Naive Bayes Classifiers. One of them uses n-gram presence, and the other, POS tag presence. The probability of a sentiment (positive, negative, neutral) of a Twitter post is obtained as the sum of the summation of the probabilities of n-gram presence and the summation of the probabilities of n-gram POS tags. Namely,

$$L(s|M) \quad = \quad \sum_{g \in N} log(P(g|s)) + \sum_{t \in T} log(P(t|s))$$

where G is a set of n-grams of the tweet, T is the set of POS tags of the n-grams, M is the tweet and s is the sentiment (one of positive, negative, and neutral). The sentiment with highest likelihood ($L(s|M)$) becomes the sentiment of the new tweet.

Pak and Paroubek (2010) achieved best result (highest accuracy) with bi-gram presence. Their explanation for this is that bi-grams provide a good balance between coverage (uni-grams) and capturing sentiment expression patterns (tri-grams) (Pak and Paroubek, 2010). Negation('not' and 'no') is handled by attaching it to the words that precede and follow it during tokenization. The handling of negation is found to improve accuracy. Moreover, they report that removing n-grams that are evenly distributed in the sentiment classes improves accuracy. Evaluation was done on the same test data used by Go et al. (2009). However, they do not explicitly put their accuracy in number other than showing it in a graph (in which it seems to approach 1) and stating it in words saying "a very high accuracy".

2.2 Assumptions and Approaches

2.2.1 Interpreting the review of literature and the way forward

In the preceding sections, an overview of general sentiment analysis and Twitter-specific sentiment analysis has been presented. The main difference between general sentiment analysis and Twitter sentiment analysis comes from the fact that Twitter posts are short, usually one sentence composed of at most 140 characters. Because of this mere fact, some classification approaches and feature selection used in general sentiment analysis may not be important to Twitter sentiment analysis. Thus discourse structure approach is easily out of competition. Similarly, relationship-based approach becomes irrelevant because there is no such thing as whole-component relationship in tweets. The two possible approaches are language models and knowledge-based approaches. By the way, almost all the sentiment analysis studies use either knowledge-based or language-model approaches. The others are merely theoretical approaches and not so much has been done with them in practice.

The choice of knowledge-based or language model approaches dictates what techniques to use. Obviously, knowledge-based approach calls for the use of unsupervised techniques, and language model calls for supervised machine learning techniques. All the Twitter-specific sentiment analyses reviewed above used supervised techniques or achieved better results with them. Although it has bee shown that supervised techniques outperform unsupervised techniques (Pang et al., 2002), in this work, for the sake of comparison, both of them will be used.

Now selecting potential approaches and techniques to use for the task at hand is not what is all there to be done. Potentially useful features and algorithms need to be selected too. There is not much to be done for unsupervised techniques except selecting and building lexicon dictionaries, which in this case are obtained from a publicly available source. However, for supervised techniques, there are several features to choose from. These are n-grams, POS tags, syntax and negation, or any combination of them.

POS can be used to disambiguate sense and to guide feature selection (Turney, 2002). But Turney himself shows that uni-gram features selected with the help of POS performed less than features selected on the basis of frequency. This diminishes the enthusiasm for using POS as features. But there is another way to use them like what Pak and Paroubek (2010) did - they used them to train Naive Bayes classifier in such a way that the classifier contributes to the overall sentiment classification. However, other than that they note differences between the POS tags of the sentiment classes, Pak and Paroubek (2010) do not state how much the classifier trained on POS tags contribute to the overall classification performance.

Syntax is reported to give n-gram equivalent results for sentence level classification (Pang and Lee, 2008). Pak and Paroubek (2010) also note that there is difference between the POS tags of subjective and objective tweets, and also between the POS tags of negative and positive tweets. This shows that there is

syntactic difference between the different sentiment classes. Therefore, it should be important to incorporate syntax into the feature vector. Likewise, incorporating explicit negation ('no' and 'not' or abbreviated form) into the feature vector might be good as it has also been shown to improve performance (Pak and Paroubek, 2010, Pang and Lee, 2008, Na et al., 2004)).

The other feature is n-gram. This is the most important and the most used feature. All sentiment classification discussed so far is done using n-grams[1]. But which n-gram (uni-gram, bi-gram, tri-gram or other higher order n-gram) gives best results? Pang et al. (2002) achieved best results with uni-grams for movie review; Dave et al. (2003) achieved better results using bi-grams and tri-grams than using uni-grams for product review; Go et al. (2009) achieved best results with uni-grams for a two-classed Twitter sentiment analysis; Pak and Paroubek (2010) achieved best results with bi-grams combined with POS tags for a three-classed Twitter sentiment analysis.

Now as can be seen from the results, there is no consensus on whether uni-grams or bi-grams (tri-grams seem to be out) is the best feature to use. They are both good competitors. However, both are capturing sentiment in different ways -uni-grams are best at capturing coverage of data, and bi-grams are good at capturing sentiment patterns (Pak and Paroubek, 2010) So, wouldn't it be good to combine them to profit from both coverages? That is exactly the approach adopted in this thesis. But, this thesis also went further to see if tri-grams can also contribute to the result. But that is not all to this approach- there is more. By combining the uni-grams and bi-grams, syntax and negation are also handled. The use of bi-grams captures collocations, and handle explicit negation by combining it to the word preceding it or following it (what Pak and Paroubek (2010) exactly did, albeit in a different way). Since POS tags are one way of expressing syntax, they are also assumed to have been handled. The combination of uni-gram and bi-gram can also handle sarcasm and irony to some extent.

A question now is whether to use uni-gram+bi-gram presence or frequency. Almost all the reviewed literature, general or sentiment specific, used n-gram presence(absence) as a binary feature following a finding by Pang et al. (2002) that, in sentiment analysis of movie reviews, term presence performed better than term frequency. Pang et al. (2002) points out that this can be one indicator of the difference between sentiment analysis and text classification. Depending on this finding, Pak and Paroubek (2010) adds that they do not believe that sentiment may be expressed through repeated use of a term. Taking into account that in IR and "traditional" text classification (as opposed to classification of sentiment) term frequency is preferred over term presence and that movie reviews are a different domain (where sentiment is usually expressed at the end of the whole review), it makes sense to experiment with term frequency and compare it with term presence. Term frequency can also be converted to TF-IDF (term frequency - inverse document frequency) to take into account

[1]Only Barbosa and Feng (2010) did not use it and their reason is because they do not believe n-grams could be a good choice for short texts as Twitter messages. But since their results are not that good, their features are not used here

term's importance to a tweet.

Another question is which supervised learning algorithms are best suited for sentiment analysis. The most common algorithms that are used in the reviewed literature are Multinomial Naive Bayes, Naive Bayes, support vector machines (SVM) and maximum entropy. The algorithms that achieved best results are SVM by Pang et al. (2002), Multinomial Naive Bayes by Go et al. (2009), Naive Bayes by Pak and Paroubek (2010) and Naive Bayes Multinomial by **?**. There will be experimentation with three of them.

A final thing about n-grams is the basis or criteria by which they will be selected. This is important because not all uni-grams or bi-grams are equally important. There are different ways to select the best n-gram features. Some of them are mutual information score (MI), chi-square, frequency, etc.

In the literature, first separating subjective and objective posts, then applying sentiment analysis to the subjective posts only has been shown to give better results in some cases Barbosa and Feng (2010), Pang and Lee (2008). Such an approach is good especially if the interest is in separating objective and subjective tweets. This was handled in a different way in this thesis - by combining the negative and positive tweets to obtain the subjective tweets. Such an approach is good because there is no need to train two classifiers (one for subjectivity detection, another for polarity detection), one classifier does both. Therefore this approach becomes hitting two birds with the same stone.

In the Twitter-specific sentiment studies, different preprocessing of the data has been used. Some prefer to delete user names, URLs, RT, hashtags. Some prefer to replace them with Equivalent classes. It will be nice to see if they make any difference.

2.2.2 Assumptions

All the Twitter sentiment analyses we saw above are done out of context, that is the posts are extracted based on a query term and analyzed for their sentiment (negative, positive or neutral) irrespective of their context. However, subjectivity and sentiment can be context and domain dependent (Pang and Lee, 2008). This thesis examines, empirically, if sentiment depends on context (the content of the news articles) for the domain of tweets about news. It investigates the extent to which it is possible to focus on the tweets about the news without getting involved with the actual news stories themselves. In other words, how much knowledge of the news stories does one need in order to understand the sentiment of the tweets about news. It is also important to look at whether tweets about news involve third party sentiments such as "I don't trust the article's support for Hamas". This is a subjective negative sentiment about a subjective positive sentiment of the original author.

There are some assumptions to be made here before moving to the next chapter. In sentiment analysis of tweets about news, news headlines are obviously assumed to be neutral. This is so because the objective of this thesis is to detect sentiment of people posting messages about news on Twitter. If somebody just posts the news headline on Twitter, s/he is not expressing her/his

sentiment. Thus tweets about news from Twitter accounts of news outlets are the best place to obtain neutral tweets. (Pak and Paroubek, 2010) also assumed and used Twitter posts from Twitter accounts of major news outlets as neutral tweets in their sentiment analysis of tweets containing a query. This assumption makes more sense in this thesis than any where else. Wilson et al. (2005) showed that the use of emoticons for training a classifier is effective. Pak and Paroubek (2010) also used them. Thus following this, tweets that contain positive emoticons and tweets that contain negative emoticons will be assumed to be positive and negative tweets respectively. Those three types of tweets (positive, negative, and neutral) will be used to train a classifier that will be applied to do sentiment classification of tweets about news. But before training and experimentation, manual annotation and study of tweets about news is done to see the effect of context, the news story in this case, on the determination of the sentiment of the tweets about news. This is done in section 3.2 of chapter 3.

2.2.3 Evaluatiom

Different papers on sentiment analysis have been reviewed, and their success of performance measured. Success of a classifier or any NLP system is measured by improvement in performance at an application task(Manning and Schütze, 1999). Now the question is how is success in performance measured? There are different ways of measuring success depending on what is required and the type of problem that one is trying to solve. The following metrics are used, among others: accuracy, precision, recall or F-measure. In information retrieval (IR) and some classification tasks, precision and recall are prefered. However, in the papers that are reviewed above accuracy was used to measure performance of classifiers and to compare them with each other. Later in chapter 5, all of the above measures will be defined and it will be explained why accuracy is a preferred measure of performance, especially when comparing performance of classifiers.

Chapter 3

Data Collection and Preprocessing

Data collection for the research is not as simple as it may seem at first thought. There are assumptions and decisions to be made. There are three differently collected datasets: test data, subjective training data, and objective (neutral) training data. Before discussing them, Twitter API will be discussed.

3.1 Twitter API

Twitter provides two APIs: REST and Streaming[1]. REST API consists of two APIs:one just called the REST API and another called Search API (whose difference is entirely due to their history of development). The difference between Streaming API and REST APIs are: Streaming API supports long-lived connection and provides data in almost real-time. The REST APIs support short-lived connections and are rate-limited (one can download a certain amount of data but not more per day). REST APIs allow access to Twitter data such as status updates and user info regardless of time. However, Twitter does not make data older than a week or so available. Thus REST access is limited to data Twittered not before more than a week. Therefore, while REST API allows access to these accumulated data, Streaming API enables access to data as it is being Twittered.

The Streaming API and the Search REST API were used during data collection for this thesis. The streaming API was used to collect training data while the Search REST API was used to collect test data. The two datasets, training data and test data, had to be collected in different ways. The reason why the Streaming API was used to collect training data is because collecting a large amount of tweets (training data is a large data) needs a non-rate-limited long-lived connection. The Test data had to be collected using the Search REST

[1] http://dev.Twitter.com/doc

API for reasons that will be clear soon.

Both the streaming API and the Search REST API have a language parameter that can be set to a language code, eg. 'en' to collect English data. But the collected data still contained tweets in other languages making the data very noisy. I therefore decided to collect tweets that contain some specific emoticons without regard to language and to later separate the data into English and non-English data. Below I discuss how the datasets were collected.

3.2 Test Data

3.2.1 Collection

Because the objective of the thesis is to analyze the sentiment of Twitter messages posted in reaction to news, only tweets about news should be collected. However, this is not a simple task. There seems to be no way of obtaining all and only tweets that are posted in reaction to news articles. To overcome this problem, some sort of bootstrapping approach was used. In this approach, first tweets that contain words of the news headline above some threshold were collected. The threshold used is 3 words and the news headlines are obtained from the news feed of the Denver Post. Once tweets are collected this way, the links from these tweets are extracted. A script uses these extracted links to go to Twitter and to fetch more tweets that contain these links.

There are two assumptions in the bootstrapping approach used above. The first is that Twitter posts about a particular news articles will always contain a link to the tweet unless the news has been circulated enough to be public knowledge. It does not make sense for somebody to post a reaction to a news article without a link to it. This assumption has a problem because the same news and therefore the same news headline can be used by a different news outlet. Therefore the tweet may not be , for example, a reaction to the news article posted by the Denver Post, but instead it is for the same news article posted by New York Times. However this does not affect the sentiment analysis task. What it affects is if somebody wants to know how many people reacted and posted in Twitter, for example, to a news article posted in the Denver Post. This can be solved by deciphering the short URLs to their real URLs. That is if a tweet contains a short URL, and on deciphering it, it gives a real URL that belongs to Denver Post, obviously the Twitter post is meant to be to the news article in The Denver Post.

The second assumption, the reason for going to Twitter to fetch more tweets that contain the link from the initial tweets, is that there will be posts that will not contain words from the news headline but that are still posted in reaction to the news. Such posts will be the types of posts that reflect the user's sentiment (negative or positive), unlike the posts that contain words from the headline, which are usually neutral ones. (). This assumption is itself based on another assumption which says that a single real URL will have the same short URL no matter how many people post and repost it on Twitter. However, this is

not true for two reasons. One reason is that there are many URL shortening services (186 according to Wikipedia) and thus the short URL will not be the same as a user may use any of them. The second reason is even for one URL shortening service, there can be more than one short URL for a give real URL because many of the URL shortening services allow users to customize their URLs. Had it not been for this two reasons, it would be possible to fetch all tweets posted in reaction to news article by a certain news outlet, for example, the Denver Post.

3.2.2 News content, Tweets about news and manual annotation

In order to see this, a corpus of 1000 tweets about news was sampled from the test data and manually examined and annotated as negative, positive and neutral. A web interface was built to aid the manual annotation. Where the sentiment of the news was possible to understand from the Twitter posts only, the sentiment was provided for it, where it was difficult to determine its sentiment, i.e. where it needed a context to assign it a sentiment, it was annotated with 'context'. Here is the procedure I followed in annotating the sentiment of the news :

- A Twitter post that, on reading, sounds to be a news headline is annotated as neutral. This does not mean it does not contain words indicating sentiment
- A Twitter post that contains subordinating conjunctions was annotated the sentiment of the main clause
- A Twitter post that contains subtleties and sarcasms was annotated as one of the three sentiment classes only if it was clearly determinable
- A Twitter post that were difficult to give a sentiment was annotated 'context' (this is a tweet that needs context to determine its sentiment).
- A sentiment expressed on the content or presentation is taken to be the same, i.e they get the same annotation.

Out of 1000 Twitter posts about news that have been annotated following the above procedure, 31 Twitter posts needed context to understand and determine their sentiment. Thus 96.9% of the test data did not require context to determine their sentiment. So, if context is ignored and if tweets about news are assumed to be context-independent, the accuracy of the assumption becomes 96.9%. This is an assumption worth taking for it is high. But it is important to note here that annotating Twitter posts about news was not easy. Many times, it was difficult to determine the sentiment. Some Twitter posts seemed both negative and positive. Neutral Twitter posts are difficult to differentiate from either negative or positive Twitter posts. This is because neutral posts can have both negative and positive sentiments. The only litmus for recognizing neutral

Twitter posts is to see if they can appear as news headline. Thus determining the sentiment of a Twitter post is not as straightforward as it may seem.

Determining the sentiment without the context of the news is made difficult by other factors too. One factor is sarcasm. What clearly seems to be positive or negative tweet may turn out to be otherwise when seen against the content of the news article. Moreover, the sentiment expressed may be on the news content or the presentation of the news. Twitter posts that contain question marks tend to require context to understand them. For example, "What is Sociology For? Doom, Gloom And Despair http://dlvr.it/DhDss ". This tweet requires reading the content of the news provided in the link to say if it is negative or positive or neutral. A related thing that I tried to examine was whether Twitter posts about news involve third party opinion such as "I do not like the article's support for Hamas". Out of 1000 tweets about news, I did not find a single tweet that involve third party opinion. So, here again, it is safe to assume that tweets about news do not involve third-party opinions.

The high accuracy (97%) above means that context can be ignored in the domain of tweets about news. Thus the whole work of sentiment analysis on tweets about news will assume that the sentiment of a tweet about news is universal and not specific to the content of that particular news. In other words, it is assumed that the sentiment of a tweet about news can be understood without the contents of the news.

Now that context was found not to matter so much in the tweets about news, the sentiment analysis task will be carried out.The annotated data minus the 31 context-requiring tweets will be used as standard test data for different learning algorithms that will be trained later in Chapter 4.

3.3 Training Data

There are two datasets that are used for the training of a classifier: subjective data and neutral data. Subjective data is data that involves positive and/or negative sentiment while neutral data is data that does not show sentiment. The following data was collected to be used to train a classifier.

3.3.1 Subjective Data

3.3.1.1 Collection

Subjective data in this context is data that contains negative and/or positive emoticons. While it is possible to collect enough negative and positive data in one or two consecutive days, the subjective data was collected in four non-consecutive days to enable randomness. Also I decided to collect negative and positive tweets at the same time instead of collecting them in different days. Collecting them at the same time was considered good because it can capture some subtle differences between negative and positive Twitter posts for some similar event. The emoticons used to collect these data are in Table 3.1. Both the negative and positive emoticons were combined to be used at the same time. This

positive	negative
:-), :), :o3, :c, :},8), :D	:-D, :-(, :(, :c, :<, :[, :{, D:, D;

Table 3.1: emoticons

positive	negative	both	total
1481679	362602	57932	1902213

Table 3.2: subjective data

means a tweet that contains positive emoticon(s) or negative emoticon(s) or both qualifies to be a subjective data. A CPAN module called Net::Twitter::Stream was used to accomplish the data collection.

A total of 1,902,213 subjective Twitter posts were collected using a combined set of the negative and positive emoticons in Table 3.1. Once data was collected, it was separated into tweets that contain only negative emotions, tweets that contain only positive emoticons and tweets that contain both emoticons. The separated results are presented in table 3.2

As can be seen from the table, a total of 1902213 tweets were found to have 1481679 only positives, 362602 only negatives, and 57932 both negatives and positives. The fact that there are more positive tweets than negative tweets shows that more people use positive emoticons than negative emoticons. It is also noteworthy that there are substantial amount of tweets that contain both negative and positive emoticons. Tweets that contain both negative and positive emoticons are confusing because they contain both sentiments.

3.3.1.2 Removing non-English tweets

Until now, we have not made sure that the negative and positive tweets, the datasets that will be used for training, are English and English only. Both positive and negative data were separated into English and non-English data. This was possible by using Google's language detection web service[2]. The Google's language detection web service requires a reference website against which strings are compared to determine their language. The strings in this case are the tweets. The web service enables one to specify a confidence level that ranges from 0 to 1. Since Twitter data contains a lot of slang and misspellings, I set the confidence level at 0 not to get rid of many English tweets. Even with confidence level set at 0, it removes some tweets which, if seen manually, are English tweets. However, since there is enough data, it is not a big problem. Since for each string, google's language detection web service had to determine its language on the fly, it takes a long time. The result of the language detection service showed that there are more tweets in other languages than in English. The results of the language detection is given in Table 3.3

The table shows how much of the data is non-English. These had to be

[2]http://code.google.com/apis/language/translate/v1/using_rest_langdetect.html

	English	non-English	%English
positive	1066447	415232	72
negative	62777	299825	17.3
Total	1481679	715057	67.5

Table 3.3: English and non-English subjective data

class	training data	test data
negative	62777	102
positive	1066447	93
neutral	1481679	771

Table 3.4: total data per class

removed from the data so they will not confuse the machine learning algorithm that will be used later. Another thing to observe from the data table is that the percentage of English and non English in both negative and positive tweets. Approximately 72% of the positive tweets are English, which suggests that the English Twitter users use positive emoticons more than the non-English Twitter users. Approximately 17% of the negative tweets are English, which suggests most non-English Twitter users posted negative tweets. If this is any indicator, it might mean relatively higher dissatisfaction of the non-English speaking people, at least for the few day the data was collected in. The overall percentage of English tweets is 67.5%, which says that more than half of all tweets are in English.

3.3.2 Neutral tweets

For neutral tweets, I collected tweets from Twitter accounts of 20 major newspapers such as New York Times, the guardian, the Washington Post, CNN News Room and BBC global News. The assumption here is that a news headline is a neutral post. The neutral tweets were collected from Twitter streaming API. They were collected on days different from when the subjective data was collected. Collecting neutral data also took time because there is no enough posts from only 20 news accounts per day. A total of 183683 neutral tweets were collected over about 30 days. The total, however, contains a lot of repetition as news headlines are retweeted many times. The duplicates will be removed later. The tweets were also not refined with Google language detection because it is known that the news outlets serve English news.

3.3.3 Total initial data

The total number of training data and test data per class is presented in 3.4 . This is after non-English tweets are removed from subjective data, and the 'context' tweets removed from the test data.

I experimented with these data, but since this is small (will be even smaller as duplicates are removed later) and since more training data is needed later on to study the effect of increasing training data on accuracy, more training data was collected and added. All data that was collected after this was also cleaned the same way the data in Table 3.4 was cleaned. Therefore, in the training data that will be used later, there will be more data for each class of the training data. Only the test data will not change. Some further preprocessing operation will be conducted under subsection 4.2.1 of chapter 4 because I wanted to see how varying them will affect performance.

Chapter 4

Experimentation and Results

Experimentation with both keyword-based and supervised techniques was done. The keyword-based approach does not require any training data. The work is only on the test data. It also does not require the test data to be represented in a special way. It can be done by string matching operations. On the other hand the supervised approach requires a lot of preprocessing on the training data. It also requires both the training data and the test data to be represented in some way. The representation can be either bag-of-words representation or feature-based representation, or both. Obviously, supervised approaches are computationally complex. Both approaches are discussed below.The keyword-based approach was included for the sake of comparison.

4.1 Keyword-based unsupervised Approach

The first task here is to build discriminatory-word lexicons. Two discriminatory-word lexicons, one containing words indicating positive sentiment and another containing words indicating negative sentiment, were built. The discriminatory word lexicons were obtained from twitrratr.com[1]. The negative-keyword lexicon contains 186 words and the positive-keyword lexicons contains 174 words. The words are listed in appendix A.

Here is how the keyword-based sentiment classification algorithm works: for each tweet, the number of positive and negative keywords found in it are counted; this is done by cross referencing the words in the tweet with the respective keyword-lexicons. If a tweet has more positive keywords than negative keywords, then it is a positive tweet; if it has more negative keywords than positive keywords, then it is a negative tweet; if it has equal number of positive and negative keywords, then it is a tie; if it contains neither negative nor positive keywords, it is neutral. This classifier was run on the test data. The test data contains 966 tweets of which 102 are negative, 93 are positive and 771 are

[1] http://twitrratr.com is a website that provides keyword-based sentiment analysis. The website makes its discriminatory-word lexicons publicly available

negative	positive	neutral		tie
15	12	70	negative	5
1	52	39	positive	1
13	42	715	neutral	1

Table 4.1: keyword-based classification on test data

neutral. This data is the standard test data. The results are presented in Table 4.1 in the form of a confusion matrix.

In the confusion matrix, the sum of the cells in a row makes the total of the actual class. For example, total number of positives is 1+52+39+1. The number at the intersection of positive column and positive row is the number that is correctly classified as positive. For positive, 52 out of 93 are classified correctly. The rest are classified incorrectly as 1 negative, 39 neutral and 1 tie. Thus, the per class accuracies are:

$$accuracy_{neg} = 15/102 = 14.7\%$$

$$accuracy_{pos} = 52/93 = 55.9\%$$

$$accuracy_{neu} = 715/771 = 92.7\%$$

What the confusion matrix and the accuracies show is that the classifier performs low in recognizing negative tweets. However, it does very well in recognizing neutral tweets. The result on positive tweets is not that bad too, 55.8%. The over all accuracy is the percentage of tweets that are classified correctly. The sum of the three cells on the diagonal from top left to the bottom right is what is correctly classified.

$$accuracy = 781/966 = 80.9\%$$

Just out of curiosity, I ran it on the training data that contains emoticons. I thought that this time the accuracy will improve even better because the keyword lexicons also involve emoticons. The result is seen in Table 4.1.

Here below are the accuracies per class (recalls) for keyword-based classification on training data.

$$accuracy_{neg} = 1936/15000 = 12.9\%$$

$$accuracy_{pos} = 5356/15000 = 35.7\%$$

neg	pos	neu		tie
1936	2219	10519	neg	326
382	5356	9098	pos	164
507	1226	13204	neu	63

Table 4.2: Keyword-based classification on training data

$$accuracy_{neu} = 13204/15000 = 88\%$$

And the overall accuracy is

$$accuracy = \frac{1936 + 5356 + 13204}{3 * 15000} = 45.5\%$$

The basis for using tweets containing emoticons for training data (emoticons will be used in supervised approaches below) lies in the assumption that the emoticon reflects the sentiment of the entire Twitter post. This assumption itself is based on the fact that Twitter posts are short (140 characters) and usually one sentence. Because of this, the emoticon is assumed to be showing the sentiment of the entire Twitter post rather than the sentiment of parts of it[2]. According to this, keyword-based sentiment classification fails. As indicated above, it does well with recognizing neutral tweets, but it has difficulty recognizing positive and negative tweets. Especially, it has more difficulty with negative tweets. Like in the experiment on the test data above, the classifier can be biased in favor of the negative class, the class that it has difficulty recognizing by classifying the 'ties' as negative. The overall accuracy improves to 46.77%, but it is still so low.

The keyword-based classifier has achieved very good accuracy, 80.9%, on the test data, but low accuracy, 45.5%, on the training data containing emoticons. There are two issues in this two contradictory results. One is if the keyword-based classifier is taken to be reliable, then the use of tweets containing emoticons for training data in supervised approaches is wrong because it says emoticons alone do not show the true sentiment of the tweet. The other is if we take emoticons alone as true indicators of sentiment, then the keyword classifier is not a good choice. But, the high accuracy achievement of the classifier on the test data is not really high if examined closely against a baseline accuracy. The test data contains a total of 966 tweets of which 102 are negatives, 93 are positives and 771 are neutrals. If the classifier just assigns a neutral class to every tweet, then the baseline accuracy is 79.8%, which is almost the same as 80.9%, the accuracy achieved by the keyword-based classifier on the test data. This

[2]Although usually true, it is possible that a Twitter post will show sentiment for part of the post. This is, for example, the reason that there are tweets that contain both negative and positive emoticons.

casts doubt on the reliability of the keyword-based classifier because the small improvement over the baseline accuracy maybe due to chance. Further, since tweets containing emoticons have been shown to be effective for training data by Read (2005) and have been used in many other sentiment analysis studies as has been shown in the review in chapter 2, it makes sense to to take emoticons to be reliable indicators of sentiment. Had the improvement of accuracy over the baseline accuracy been so big as to indicate that it really is not by chance, it would be important to examine the extent to which different keywords(including emoticons) contribute to the improvement and to make further analysis.

One can argue that the performance of the keyword-based classifier is as good as the lexicons it uses as input. Although in the context determination it has been shown that sentiment of tweets can be computed without context suggesting that tweets about news are similar with other general tweets at least in context independence, there is still a posibility that tweets about news may use limitted lexicon. Thus it is possible that the results of the keyword-based approach may be improved with news-specific lexicon, but building a news-specific lexicon is not done here.

4.2 Supervised Approaches

This is where most of the time and effort was spent. Under this section, different supervised machine learning approaches were used. To enable experimentation with different machine learning algorithms and to enable the identification of factors that affect results, a three-step process was used. The first step is preprocessing the training data. The second is feature extraction and data representation. The third is classifier training and testing with different machine learning algorithms. This approach helps to experiment with different possibilities by varying the preprocessing operations, feature extraction and then the machine learning algorithms. Each of them are explained in the following subsections.

4.2.1 Preprocessing Training Data

Cleaning the data: Since tweets contain several syntactic features that may not be useful for machine learning, the data needs to be cleaned. The cleaning is done in such a way that it confirms to WEKA's data representation format. A module that enables choice of different cleaning operations was developed. The module provides these functions:

- remove quotes - provides the user to choose to remove quotes (“”) from the text
- remove @ - provides choice of removing the @ symbol, removing the @ along with the user name, or replace the @ and the user name with a word 'USERNAME'
- remove # - removes the hashtag

class	total of a class	after duplicates removed	Duplicates	% of duplicates
positive	122112	111192	10920	8.9
negative	104353	97953	6400	6.1
neutral	183683	15596	168087	91.5
total data	410148	224741	185407	45.2

Table 4.3: Removal of Duplicates

- Remove URL - provides choices of removing URLs or replacing them with 'URL' word
- Remove RT - removes the word RT from tweets
- Remove frownsmiles- removes tweets that contain both emoticons

For start, the following cleaning operations have been done on the training data. All URLs have been removed from tweets. The hashtag(#), a symbol used to indicate nouns, has been removed. Tweets that contain both negative and positive emoticons were removed from the dataset to avoid confusing the machine learning algorithm.

Removing Duplicates: Duplicates were removed from all training data.Only exact duplicates were removed. The results of the removal of duplicates is given in Table 4.3.

As can be seen from the table, there are duplicates in each class. The number of duplicates in neutral is overwhelming, but expected. This is because the same headline is twittered and retweeted. Also the fact that there are more percentage of duplicates in positive tweets than there are in negative tweets is expected because people will tend to retweet something they are happy with. The total percentage of duplicates is 45, close to half of all tweets.

4.2.2 Feature Extraction and Instance Representation

One question in machine learning is how to represent the data. Both the training and test data must be represented in some way in order for a machine learning algorithm to learn and build a model. Some of the ways that data can be represented are feature-based or bag-of-words representation. By features, it is meant that some attributes that are thought to capture the pattern of the data are first selected and the entire dataset must be represented in terms of them before it is fed to a machine learning algorithm. Different features such as n-gram presence or n-gram frequency, POS tags, syntactic features, or semantic features can be used. For example, one can use the keyword lexicons that we saw above as features. Then the dataset can be represented by these features using either their presence or frequency.

In bag-of-words representation, a tweet is represented in terms of the words or phrases it contains. The difference between bag-of-words representation of

a Twitter post and the set-of-words of the same Twitter post is that, in bag-of-words representation, if a word occurs twice, it will be also present twice in the bag-of-words representation while in set of words, it will only be present once regardless of how many times it is found in the Twitter post. In this work, features and bag-of-words were used at different steps for different purposes. The bag-of-words representation was used in attribute selection, and feature representation was used in data representation.

4.2.2.1 Attribute Selection

Attribute selection is the process of extracting features by which the data will be represented before any machine learning training takes place. Attribute selection is the first task when one intends to represent instances for machine learning. Once the attributes are selected, the data will be represented using the attributes. So attributes are the features. For the work at hand, the attributes were selected from the data itself. First the entire dataset was converted into bag-of-ngrams. By bag-of-ngrams, it is meant either bag-of-uni-grams or bag-of-bigrams or bag-of-trigrams or a combination of any of them. Bag-of-uni-grams is just bag-of-words. The others are also like bag-of-words except that they are made up of phrases. How many attributes should be selected as features and what criteria to use in selecting competitive features will be discussed below. When attributes are selected, it is also important to think about which words to exclude from being selected as attributes. Articles, for example, may not be important. This is done by using stop words.

4.2.2.2 Instance Representation.

Once features are selected, the data must be represented in some way in terms of the features. A choice of whether to use uni-gram presence or uni-gram frequency, bi-gram presence or bi-gram frequency, tri-gram presence or tri-gram frequency, or a combination of uni-gram+bi-gram presence or frequency, etc. has to be made. Although we used the entire data set in our selection of attributes, the representation of the data must be done on a per instance (Twitter post) basis. It will be clear soon with an example.

4.2.3 Machine-learning algorithms

According to the literature, multinomial Bayes classifier and Support vector machines are found to give better accuracy. Two sets of algorithms were experimented with in this thesis. Bayes classifiers and Support Vector machines.

4.2.3.1 Bayes classifiers

All Bayesian models are derivatives of the well-known Bayes Rule. Bayes Rule says the probability of a hypothesis given a certain evidence, i.e. the posterior probability of a hypothesis, can be obtained in terms of the prior probability

of the evidence, the prior probability of the hypothesis and the conditional probability of the evidence given the hypothesis. Mathematically,

$$P(H|E) = \frac{P(H)P(E|H)P}{P(E)}$$

where
$P(H|E)$- posterior probability of the hypothesis.
$P(H)$- prior probability of hypothesis
$P(E)$- prior probability of Evidence
$P(E/H)$- conditional probability of Evidence given Hypothesis.

In our case, we would have three hypothesis and the one that has the highest probability would be chosen as a class of the tweet whose sentiment is being predicted. For this work the mathematical formulas of obtaining the probability of a tweet becoming neutral is :

$$P(s|E) = \frac{P(s) * P(E|s)}{P(E)}$$

s stands for sentiment. The E (evidence) stands for the new tweet whose class is being predicted. P(s) and P(E|s) are obtained during training. s can be either of the three classes. This mathematical formula is used to compute the probability of a tweet being neutral, positive or negative. It is saying: what is the probability of E (the new Twitter post about news, in our case) being one of the classes (neutral, negative, positive) given the values of P(s) and P(E|s). The only problem is what the value of P(E) is. However, as will be clear below with example, P(E) does not need to be computed. Thus probability of the the new tweet being one of the classes can be computed from $P(s)$ and $P(E|s)$. The class that has the highest probability is chosen as the sentiment of the Twitter post. Naive Bayes, multinomial Naive Bayes are some algorithms that use the standard Bayes rule or a variation of it. These algorithms will be explained below. But, first an important concept called multinomial experiment will be discussed because it is a basis for all of the algorithms.

4.2.3.2 Multinomial Experiment

An experiment is a procedure that has three things: each procedure can have more than one outcome, each outcome is known in advance, there is uncertainty in each outcome. Tossing a coin is an experiment because it has more than one outcome (head and tail), head and tail are known as outcomes before the experiment, and whether it will be head or tail is entirely dependent on chance.

A multinomial experiment is an experiment that has four properties.

- the experiment is repeated n times (n trials) - throwing dice 20 times
- Each trial can result in a discrete number of outcomes - 1 through 6

Suppose a multinomial experiment consists of n trials, and each trial can result in any of k possible outcomes: E_1, E_2, . . . , E_k. Also suppose that each possible outcome can occur with probabilities p_1, p_2, . . . , p_k. Then, the probability (P) that E_1 occurs n_1 times, E_2 occurs n_2 times, . . . , and E_k occurs n_k times is

$$P = [\frac{n!}{n1! * n2! * ...nk!}] * (p_1^{n_1} * p_2^{n_2} * ... * p_k^{n_k})$$

where $n = n_1 + n_2 + ... + n_k$ rearranging, it becomes

$$P = n! \prod_{i=1}^{k} \frac{P_i^{n_i}}{n_i!}$$

Where k is the number of outcomes

Figure 4.1: multinomial formula

- The probability of any outcome is constant - probability of getting 1, 2, 3, 4, 5 or 6 is 1/6 any time the dice is thrown
- The trials are independent; that is, getting a particular outcome on one trial does not affect the outcome on other trials -getting 2 in trial one does not have any effect in getting 2 or any of the other outcomes in subsequent experiments.

Multinomial Formula Multinomial Formula is the formula by which a probability of an outcome is computed. The probability of an outcome is obtained by the formula in Figure 4.1 as defined in stattrek website(Unknown) .

Suppose we throw a dice 20 times, what is the probability that 1 occurs 2 times, 2 occurs 2 times 3 occurs 3 times, 4 occurs 6 times, 5 occurs 5 time, and 6 occurs 2 times ?

Experiment -throwing dice 20 times
trial results in any of 1 through 6 outcome
probability of getting 1, 2, 3, 4, 5 or 6 is constant. each has 1/6 probability.
the trials are independent.
thus:

$$p(1) = p(2) = p(3) = p(4) = p(5) = p(6) = 1/6$$

and $n_1 = 2$, $n_2 = 2$, $n_3 = 3$, $n_4 = 6$, $n_5 = 5$, $n_6 = 2$. Plugging the values in the multinomial formula,

$$P = [\frac{20!}{2! * 2! * 3! * 6! * 5! * 2!}] * ((\frac{1}{6})^2 * (\frac{1}{6})^2 * (\frac{1}{6})^3 * (\frac{1}{6})^6 * (\frac{1}{6})^5 * (\frac{1}{6})^2)$$

$$P = 1.6e^{-10}$$

Let's define a variable for each outcome. Let x refer to '1 occurs n_1 times'. The values of variable x depend on how many times the dice is thrown, that is on n. x is called a random variable. There are similar other 5 variables for '2 occurs n_2 times', '3 occurs n_3 times', '4 occurs n_4 times', '5 occurs n_5 times', and '6 occurs n_6 times'. So there are k random variables. Now lets say we are interested only in presence, and not in how many times 1, 2, 3, 4, 5 or 6 occur in throwing a dice. That means $n_1 = n_2 = n_3 = n_4 = n_5 = n_6 = 1$. Thus the formula reduces to

$$P = n! \prod_{i=1}^{k} \frac{P_i^{n_i}}{n_i!}$$

$$P = n! \prod_{i=1}^{k} \frac{P_i^{1}}{n_i!}$$

$$P = n! \prod_{i=1}^{k} \frac{P_i}{n_i!}$$

Since there are more than one number of variables(k number of variables), the formula is multi-variate. Also since the variables can have only two values success (present) or failure (absent), it is binomial. A function, graph, or table that links the values of the random variable(s) to their corresponding probability is called probability distribution(Manning and Schütze, 1999). A probability distribution which tends to cluster around a mean is called a normal probability distribution(Manning and Schütze, 1999).

Below are algorithms as implemented in WEKA. The Naive Bayes algorithms are based on multinomial experiment discussed above. Therefore, indepdence of attributes is assumed.

Multinomial Naive Bayes: Multinomial Naive Bayes uses the Multinomial distribution we saw above. Thus the probability of a Twitter post being a certain class is

$$P = n! \prod_{i=1}^{k} \frac{P_i^{n_i}}{n_i!}$$

In our case, n is the total number of attributes used (the total trial), k is the number of outcomes of the experiment (the attributes found in a single Twitter post about news), P_i is the probability of i^{th} outcome (an attribute found in the Twitter post) and n_i is the number of times the i^{th} attribute occurs in the

Twitter post. Since an attribute can occur 0, 1, 2, 3 etc. times, the experiment is Multinomial. This formula is used to compute the probability of each sentiment class. During the computation of whether a certain Twitter post about news is positive, negative or neutral, only the value of P_i changes, all others remain the same. P_i changes because the probability of an attribute in the neutral, positive or negative classes are different. The factorials of n and n_i help to account for the fact that order of occurrence of the attributes in the Twitter post does not matter. But since factorials of n and n_i are the same for each class and thus do not a help in the comparison of the probability of a Twitter post being a certain class, they can be dropped simplifying the formula to

$$P = \prod_{i=1}^{k} P_i^{n_i}$$

This is simply the multiplication rule of probabilities applied to probabilities of each attribute raised to the number of times the attribute occurs. The formula is a modification of the standard Bayes rule to accommodate frequency of occurrence of an attribute in a certain Twitter post. Basically, what it means is that if an attribute occurs twice, it will be taken into account by multiplying its probability twice.

The probability of each attribute in each sentiment class is obtained during the training. For example, if we have an attribute, say 'few' in a certain Twitter post, it will have three probabilities, one for each of the sentiment classes of neutral, positive, and negative. Given a tweet, the evidence(E), the formulas for the probability for a Twitter post being neutral becomes

$$P(neu|E) = \prod_{i=1}^{k} Pneu_i^{n_i}$$

The same formula are obtained for negative and positive classes by replacing *neu* by *pos* and *neg*. After the probabilities of a certain Twitter post being produced by each of the three classes is computed, the one that produces it with the highest probability becomes its sentiment class. This assumes the three sentiment classes have the same probability of existence. If they do not have, the formula becomes

$$P(neu|E) = (\prod_{i=1}^{k} Pnew_i^{n_i}) * P(neu)$$

The result is multiplied by the probability of the sentiment class.

Naive Bayes: Naive Bayes is like multinomial Naive Bayes. The only difference is that it does not take frequency of an attribute into account. Naive Bayes

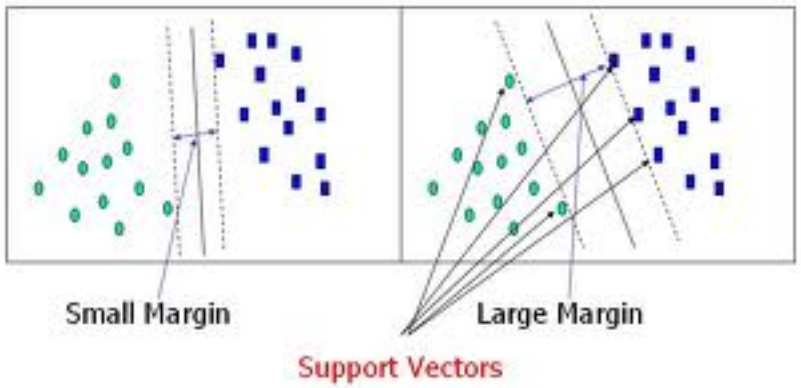
uses presence or absence of an attribute rather than frequency. This means the formula becomes:

$$P(neu|E) = (\prod_{i=1}^{k} Pneu_i) * P(neu)$$

Naive Bayes normalizes frequencies for nominal attributes into probabilities. For numeric attributes, it uses the Gaussian or normal distribution to compute the probability density function. Where the normal assumption is grossly wrong, Naive Bayes uses kernel density estimators to improve performanceWitten and Frank (2005).

4.2.3.3 Support Vector Machines

This algorithm is different from the above ones because it works in a completely different way. While missing values are all ignored (only, present values are taken into account) in all the above Bayes algorithms, Support Vector Machine (SMO in WEKA terms) replaces them globally. Nominal attributes are discretized (changed into binary ones) and normalized. It works to find a line or plane that divides the data into two. The line should be as far away as possible from the two data sets it divides. This line is called maximum hyperplane.



In a a three-classed classification (as we have here), there will be three pair-wise classification. That is positive-negative, negative-neutral, and positive-neutral. Support vector machines find the maximum hyperplane that divides the training space into classes as far apart from each other as possible. SVM is computationally expensive since it involves all the discretization, normalization and repetitive dot products operations.

4.2.3.4 WEKA

WEKA(Witten and Frank, 2005) is an excellent machine learning toolkit developed by University of Wakaito. WEKA comes with many machine learning

algorithms and filtering tools. WEKA runs in many platforms including windows, Linux and Macintosh. It is written in Java and is available under GNU General Public License. It provides both data filtering and machine learning algorithms.

WEKA can be used for three purposes: to apply it to a dataset with the purpose of learning more about the dataset itself, to use learned models to predict new instances, and to compare different learning algorithms in order to select the best performing one for the task at hand(Witten and Frank, 2005). WEKA's algorithms can also be embedded in other applications where machine learning is a part of some larger task thus saving time and energy in coding algorithms afresh.

All the tasks of attribute selection, representation and learning are done with WEKA. WEKA's StringToWordVector enables one to change parameters and do many filtering tasks. It enables both attribute selection and representation. WEKA is easy to use and provides different tunable parameters for filtering and training learning algorithms, thus enabling one to change parameters and to see if they improve performance.

To show how the attribute selection, instance representation, training classifier and testing works in WEKA, an example showing the steps and processes is given below. The example is simplified to avoid getting into technical difficulties. It is just to highlight the important steps and processes. It aims to show how attribute selection, representation, learning and prediction are made using WEKA.

Training Data: Let's say we have the training data in Table 4.4. The example has 13 tweets that have been extracted from the actual training data randomly. This is after some preprocessing has been conducted on the data. The tweets are on the first column and their classes annotations are on the second column.

Attribute Selection and representation of instances: Once we have the preprocessed training data, the next task is to select attributes. To keep the explanation simple, let's focus on extracting uni-gram attributes. To do so, all the above Twitter posts are converted into bag of words representation. All the words in the bag-of-words were converted into lowercase. Then out of all the words, some words must be chosen as attributes. The choice is made on the basis of frequency. However, before doing so, stop words were used to exclude words such as articles that have nothing to do with sentiment. The stop words used are:

that was we what where you your I rt a and are he is it me my that the them this to u s so.

Lowercasing and selecting the frequent words as attributes is done by specifying parameters in WEKA's StringToWordVector filtering tool. It is possible to specify the number of attributes to keep for each class. This may mean that each class will have its own attributes. It is also possible to specify how frequently a word must occur in the entire bag-of-words to be chosen as an attribute. This

tweets	class
I love how in the Never Say Never music video, Justin is eating twizzlers. :)	positive
any more :(I made a video of her with clips from the Wellington show but it doesn't have any effects or anything...	negative
OH can't win in arcades. So he does this.. :)	positive
MEET ME ON NOTHERN LIGHTS DOCK I WILL DO SOMETHING REALLY REALLY SERIOUS ITS ABOUT MY LIFE AND Twitter :(NOW	negative
RT : 500 federally-protected vultures are wintering in a small Virginia town – to the disgust of residents .	neutral
off to sleeps :) love you all xx	positive
RT : @c_bence that pic was.... Disgusting. My poor eyes // ughhh and then I opened it :(	negative
RT : Couple floating on inflatable dolls rescued from Melbourne's Yarra River - news.com.au	neutral
LMFAO I turned your pixel-art into a Galaga ship without thinking about it even once :D I guess barbs == UNIVERSALLY AWESOME	positive
:-(Feel better soon, SP!	negative
RT : Israeli court indicts suspected leader of a cell of neo-Nazis group - haaretz.com	neutral
pleez add mesoon never stopped lovin your music and talent :) @Breeydoesit	positive
NHS trusts in north-west ban 57 types of surgery	neutral

Table 4.4: Example training data

```
@attribute class {negative,positive,neutral}
@attribute any numeric
@attribute com numeric
@attribute from numeric
@attribute in numeric
@attribute love numeric
@attribute music numeric
@attribute never numeric
@attribute of numeric
@attribute on numeric
@attribute really numeric
@attribute soon numeric
@attribute t numeric
@attribute video numeric
```

Figure 4.2: Attributes

```
positive,0,0,0,0,1,1,1,1,0,0,0,0,0,1
negative,0,1,0,1,0,0,0,0,1,0,0,0,1,1
positive,0,0,0,0,1,0,0,0,0,0,0,0,1,0
negative,1,0,0,0,0,0,0,0,0,1,1,0,0,0
neutral,0,0,0,0,1,0,0,0,1,0,0,0,0,0
positive,0,0,0,0,0,1,0,0,0,0,0,0,0,0
negative,0,0,0,0,0,0,0,0,0,0,0,0,0,0
neutral,0,0,1,1,0,0,0,0,0,1,0,0,0,0
positive,1,0,0,0,0,0,0,0,0,0,0,0,0,0
negative,0,0,0,0,0,0,0,0,0,0,0,1,0,0
neutral,0,0,1,0,0,0,0,0,1,0,0,0,0,0
positive,0,0,0,0,0,0,0,1,1,0,0,0,1,0,0
neutral,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0
```

Figure 4.3: Intance representation

was the one used in this example. The 13 most frequent words were chosen as attributes. The first attribute is the class attribute. The attributes and intance representations are presented in Figures 4.2 and 4.3 exactly as they appear in WEKA. The attributes in our case are all numeric except the class attribute which is nominal.

Note that the attributes are selected from the entire training corpus on the basis of frequency. Thus if a word occurs more frequently in all training corpus than another, it will be selected first. Only when the more frequently occurring words are finished, a less frequent word gets chosen. Once the attributes are selected, each instance is represented in terms of the attributes. According to uni-gram presence representation, the first instance becomes

positive,0,0,0,0,1,1,1,1,0,0,0,0,0,1

The 0's show that the attributes that are not present in the instance. Ac-

Attribute	negative	positive	neutral	total
about	1	1	0	2
any	1	0	0	1
com	0	0	2	2
from	1	0	1	2
in	0	2	2	4
love	0	2	0	2
music	0	2	0	2
never	0	2	0	2
of	1	0	3	4
on	1	0	1	2
really	1	0	0	1
soon	1	1	0	2
t	1	1	0	2
video	1	1	0	2

Table 4.5: Attribute counts in each class

cording to uni-gram frequency representation, the first instance would be:

positive,0,0,0,0,1,1,1,2,0,0,0,0,0,1

The 9^{th} attribute has value 2 now because 'never' occurs twice in the instance.

Learning a model: Once attributes are selected and the training data is represented in terms of the attributes, training starts. Training is learning a model according to which prediction on test data will be made. During the training, the presence of each attribute value in each of the classes (negative, positive, and neutral) is counted as in Table 4.5.

From the attribute counts in Table 4.5, the observed probabilities of attributes in each class(negative, positive, or neutral) are computed. The results are in Table 4.6.

The probabilities of the classes are also computed. This count is about how many of the instances are negative, positive and neutral.

The number of tweets for each class value are counted separately and divided by the number of instance. The results are shown in Table 4.7.

The above fractions in tables 4.6 and 4.7 are called observed probabilities. In the first table P(neg) stands for probability of the attributes in the first column to be negative. P(pos) and P(neu) also stand for probability of the attributes in the first column to be positive and neutral respectively. However, in the second table, L(neg) stands for probability of the negative class in general and not of a particular attribute. L(pos) and L(neu) are for probabilities of positive class and neutral class respectively..

Attribute	P(neg)	P(pos)	P(neu)
about	1/2	1/2	0/2
any	1/1	0/1	0/1
com	0/2	0/2	2/2
from	1/2	0/2	1/2
in	0/4	2/4	2/4
love	0/2	2/2	0/2
music	0/2	2/2	0/2
never	0/2	2/2	0/2
of	1/4	0/4	3/4
on	1/2	0/2	1/2
really	1/1	0/2	0/2
soon	1/2	1/2	0/2
t	1/2	1/2	0/2
vodeo	1/2	1/2	0/2

Table 4.6: Observed probabilities of attributes

L(neg)	L(pos)	L(neu)
4/13	5/13	4/13

Table 4.7: Class probabilities

Prediction: Now, let's say we want to determine the sentiment of the following new Twitter post.

'Egyptians from all walks in Cairo are vowing to drive Egypt President Hosni Mubarak from office'

First, a learning algorithm will look for the presence of any of the attributes above in the new tweet. 'in' and 'from' are found. After that, the probabilities of each of the hypotheses are computed from the two attributes. There are three hypotheses to be tested. They are: it is neutral, it is positive and it is negative. Let's represent them by P(neu|E), P(pos|E) and P(neg|E) respectively. The given new Twitter post is the evidence (E). Thus when we say P(pos|E), we are saying what is the probability of this new Twitter post being positive. Likewise, P(neg|E) and P(neu|E) are the probabilities of the new Twitter post being negative and neutral respectively. Now let's compute the probabilities of the three hypotheses in Naive Bayes. The probability that the new Twitter post is neutral is :

$$P(neu|E) \quad = \quad (\prod_{i=1}^{k} Pneu_i) * P(neu)$$

Since we have only two attributes, $k = 2$. Thus the formula becomes

$$P(neu|E) = (Pneu_{in} * Pneu_{from})) * P(neu)$$

$Pneu_{in}$ and $Pneu_{from}$ are the probabilities of 'in' and 'from' in the neutral class. and $P(neu)$ is the probability of the neutral class. We know all the values of this from the above learned model. $Pneu_{in} = 2/4$, $Pneu_{from} = 1/2$ and $P(neu) = 4/13$. With these values

$$\begin{aligned} P(neu|E) &= (2/4 * 1/2)) * 4/13 \\ P(neu|E) &= 8/104 \end{aligned}$$

The probability of the new Twitter post being negative and positive is computed in a similar manner. The probabilities of 'in'; and 'from' for positive and negative are also obtained from the learned model above. Plugging them in the formulas for each class

$$\begin{aligned} P(pos|E) &= (Ppos_{in} * Ppos_{from})) * P(pos) \\ P(pos|E) &= (2/4 * 0) * 5/13 \\ P(pos|E) &= 0 \end{aligned}$$

Similarly

$$\begin{aligned} P(neg|E) &= (Pneg_{in} * Pneg_{from})) * P(neg) \\ P(neg|E) &= (0 * 1/2) * 4/13 \\ P(neg|E) &= 0 \end{aligned}$$

Now that we know the probability of Twitter post being generated by each of the sentiment classes, we can decide what its sentiment is. Clearly, this Twitter post is neutral. This is essentially how Naive Bayes works. Naive Bayes does not care about how frequent an attribute occurs in the new Twitter post whose sentiment is being computed. All it cares is if the attribute is present. Note that 'from' occurs twice in the Twitter post. If one wants to take into account the frequency of occurrence of an attribute, one must use multinomial Naive Bayes. If multinomial Naive Bayes is used, the probability of being in the neutral class becomes:

$$\begin{aligned} P(neu|E) &= (Pneu_{in} * Pneu_{from}^2)) * P(neu) \\ P(neu|E) &= (2/4 * (1/2)^2)) * 4/13 \\ P(neu|E) &= 2/208 \end{aligned}$$

Since the probabilities of negative and positive still remain 0 (because of the multiplication by 0), the new Twitter post is predicted neutral in multinomial

Naive Bayes too. However, the multiplication by 0, which gives attributes whose probability is zero a veto power in the decision making is something to be concerned about. As we can see from the above examples, if one of the attributes has zero probability, the entire result is rendered zero. This can be avoided by adding some number, an operation called smoothing. Smoothing gives some probability mass to non-occurring attributes(Manning and Schütze, 1999). For example, If 1 is added to each numerator and 3 to each denominator in the fractions of table 1, we can obtain the results to be different as shown below for Naive Bayes. The add 1 smoothing operation is called Laplace estimator.

$$\begin{aligned} P(neu|E) &= (4/13 * 3/7 * 2/5) = 24/455 \\ P(pos|E) &= (5/13 * 3/7 * 1/5) = 15/455 \\ P(neu|E) &= (4/13 * 1/7 * 2/5) = 12/455 \end{aligned}$$

Although still this result shows that the tweet is neutral, it is not difficult to observe that the result can change with such kind of adjustment resulting in a different prediction altogether.

The example above showed the important processes involved in predicting the sentiment of a Twitter post using a few training data and one test data. Each Twitter post was represented in terms of the attributes presence. Missing values of attributes were represented by 0's. So far it is fine, but what about when the training data is huge and so the attributes are also huge in number. Remember in reality we have to deal with huge number of training data and a number of attributes. In that case the representation can be changed to sparse format. The sparse format only represent the attributes that are not missing. For example, the first instance would be represented as below. Only attributes that have presence appear.

{0 positive,5 1,6 1,7 1,8 1,14 1}

The above example used uni-gram attributes only. It is possible to to use bi-grams and tri-grams or a combination of two or three of them. WEKA's StringToWordVector tool enables one to choose uni-gram, bi-gram or tri-gram or a combination of them. However, since uni-grams are most frequent, only if the number of attributes is high will bi-grams and tri-grams be selected as attributes.

There are two things to note here: One is that Naive Bayes and multinomial Naive Bayes work only with attributes that have presence in the Twitter post; attributes with missing values are ignored. Another is that the probabilities can be computed differently. For example in the above, we computed the probabilities based on presence. It is also possible that the probabilities be computed from the number of times the attributes occur in the training data, the frequency of the attributes. Once the probabilities are computed in one of the two ways, the learning algorithm can be applied.

In the above demonstration example, Naive Bayes and multinomial Naive Bayes were presented as if they are directly handling the multi-valued class, that is the class attribute. However, in WEKA's implementation, they do not

handle multi-valued attributes directly. Instead, WEKA uses an approach called pair-Wise classification. In pair-wise classification, a classifier for each pair of classes that can be derived from the multi-Valued class is developed, using only the instances that belong to the two classes (Witten and Frank, 2005). For the three-class classification that we have here, there will be three pair-wise classes: positive-negative, positive-neutral, and negative-neutral. Out of the three pairs , the one that receives more votes is selected.

However, there is another way to make algorithms such as Naive Bayes and multinomial Naive Bayes handle multi-valued classes such as the one we have here. Algorithms such as Naive Bayes, multinomial Naive Bayes, or SVM that do not directly multi-class classification can be wrapped in other more powerful algorithms that handle multi-class classification . In this case, the FilteredClassifier algorithm of WEKA was used to do that. The FilteredClassifier algorithm also helps to enable the processing of test data in the same way the training data was processed. In FilteredClassifier, the learning algorithm (Naive Bayes, Naive Bayes Multinomial or SVM) and filtering tools (StringToWordVector) are provided as parameters.

4.2.3.5 Presence, frequency(count) and TF-IDF

In the demonstration above, the instances were represented in terms of attribute presence.That is just one way of representation. Following the proposed approach in section 2.2.1, there will be experiments with presence, frequency (or its FT-ID) form. The difference between term presence and term frequency is that in presence only 1 or 0 is registered while in frequency, it is how many times the attribute occurs (1, 2, 3, etc.) that is registered. The use of presence(absence) or count(frequency) affects the probabilities during learning the model. And the effect on probabilities affects the prediction of the learning algorithm used.

One other thing to consider is whether every attribute has the same importance for a Twitter post. For example, does 'the' have the same importance as 'disgusting' for a given Twitter post? There is a way of measuring this importance called TF-IDF (Term Frequency, Inverse Document Frequency). The TF-IDF (let's call it I) of an attribute a in Twitter post t is obtained by:

$$I = f_{at} * \log \frac{(T)}{T_a}$$

Where f_{at} is the attribute's (a) frequency in a Twitter post (t), T is total number of Twitter posts used in the training, T_a is total number of Twitter posts that include attribute t. Attributes are the words selected from the tweets. This gives most frequent words less weight, while it gives rare occurring words much weight. In the demonstration example discussed above, the attributes were treated as binary ones, and the probabilities were obtained from counting. Here, the probability will not be obtained by counting. The attributes will be treated as numeric ones and mean(x) and standard deviation(σ) will be computed for

each attribute for each class, and the probability density (assuming normal distribution) of that attribute in a class will be computed as:

$$f(x) = \frac{1}{\sqrt{2\Pi}\sigma} - e^{\frac{(x-u)^2}{2\sigma^2}}$$

In case the normal assumption is grossly wrong, WEKA uses other density estimators(Witten and Frank, 2005). If this probability density formula was applied to the demonstration example, the probabilities show minor differences, but the prediction is the same.

In this thesis, three things have been done. One is experimentation with presence, frequency and TF-IDF for data representation. second is experimentation was three different machine learning algorithms (Naive Bayes, multinomial Naive Bayes, and SVM). Third is experimentation with uni-grams, bi-grams, and uni-grams+bi-grams. All the results are presented in the subsequent section. All work is done using WEKA's filtering tools and machine learning algorithms.

4.2.3.6 Results

In all the tables below, the same training data and test data is used. The total training data used is 46800 (15600 neutral, 15600 positive, and 15600 negative). This is done not to bias the classifier in favor of any of the classes. In each of the algorithms, the same number of attributes selected on the basis of frequency are used. The total number of attributes used is 10,000. Before attribute selection, training data was converted to lowercase. Stop words were used. The lower limit of frequency for an attribute to be chosen was set at 1. The results are presented tables 4.8, 4.9, and 4.10.

When we use the same number of attributes (10,000) with uni-grams, bi-grams or uni-grams+bi-grams, there are some things to note. In uni-grams, a maximum of 10,000 most frequent words will be selected as attributes. Similarly a maximum of 10,000 most frequent bi-gram words will be selected when we use bi-grams. However, since there will not be the same number of bi-grams as uni-grams, there might be a problem of data sparseness. When uni-grams+bi-grams are used, both uni-grams and bi-grams will be selected as attributes on the basis of frequency. Obviously, there will be more uni-grams than bi-grams. In place of some uni-grams, some frequent bi-grams will be chosen. For example, if the word 'Mubarak' occurred 4 times in the entire training corpus and was chosen as an attribute in the uni-gram, it will be left out and 'step down' will be chosen instead in the uni-grams + bi-grams if 'step down' occurred more frequently in the entire training corpus.

The tables show performance of three different machine learning algorithms under three choices of representations. The representations are presence, frequency and TF-IDF. Under each representation, uni-grams, bi-grams and uni-grams+bi-grams are used. Naive Bayes achieves best result with bi-grams for each representation. Its best result is an accuracy of 81.47% and is obtained

	uni-gram	bi-gram	uni-gram+bi-gram
Naive Bayes	75.88	78.78	77.02
multinomial Naive Bayes	85.82	64.80	86.44
Support vector machine	77.02	77.74	79.09

Table 4.8: Accuracies using presence representation

Classifier	uni-gram	bi-gram	uni-gram+bi-gram
Naive Bayes	75.98	81.47	76.92
Multinomial Naive Bayes	84.56	64.70	86.23
Support vector machine	78.68	76.40	81.37

Table 4.9: Accuracies using frequency representation

with frequency or TF-IDF representation (both representations give the same results for Naive Bayes). Multinomial Naive Bayes achieves best results with uni-grams+bi-grams for each representation. The best result for multinomial Naive Bayes is 86.44% and is achieved with uni-grams+bi-grams presence. SVM also achieves best result (an accuracy of 81.37%) with uni-grams+bi-grams using frequency and TF-IDF representations. Of all the three algorithms, multinomial Naive Bayes achieves the best result.

A closer look at the tables shows uniform behaviors for multinomial Naive Bayes. One uniform behavior is that for each representation, multinomial Naive Bayes achieves higher accuracy with uni-grams than bi-grams. Also, for each representation uni-grams+bi-grams achieves even higher accuracy than uni-grams or bi-grams in isolation. I attribute the achievement of higher accuracy with uni-grams+bi-grams to the contribution from both uni-grams and bi-grams. While uni-grams provide a good coverage, bi-grams capture collocations, syntax, negation, etc. This uniform behavior means that uni-gram+bi-gram is the best feature when using multinomial Naive Bayes for sentiment analysis of tweets. Another uniform behavior is that for each n-gram (uni-gram, bi-gram, uni-gram+bi-gram), multinomial Naives Bayes achieves better results under presence than under frequency or under TF-IDF. This second uniform behavior means that presence is the best representation when using multinomial Naive Bayes for sentiment analysis of tweets. Both observations lead to the conclusion that uni-gram+bi-gram presence is the best feature when using multinomial Naive Bayes for sentiment analysis of tweets.

	uni-gram	bi-gram	uni-gram+bi-gram
Naive Bayes	75.98	81.47	76.92
Multinomial Naive Bayes	84.58	63.35	85.61
Support vector machine	78.57	76.29	81.37

Table 4.10: Accuracies using tf-idf representation

Encouraged by the improved accuracy when using uni-gram+bi-gram as features, I wondered if tri-gram might also contribute to the performance of multinomial Naive Bayes. To see this, I experimented with uni-gram+bi-gram+tri-gram presence. An accuracy of 86.23% was obtained, which is a bit lower than 86.43, the accuracy obtained with uni-gram+bi-gram presence. I also experimented with frequency and TF-IDF representation, but it did not improve the accuracy. Therefore, it seems uni-gram+bi-gram is the best feature.

Now the best algorithm, the best representation format, and the best feature for sentiment analysis of tweets are known. The best algorithm is multinomial Naive Bayes, the best representation format is presence and the best feature is uni-grams+bi-grams. A question here is to conclude so, are the results statistically reliable? This is an hard-to-solve question because to check statistical reliability for each representation format for each algorithm for each attribute selection (uni-gram, bi-gram, uni-gram+bi-gram), one has to experiment with a number of different attributes and a number of different training data. This is practically very difficult since there are many possible combinations and since SVM and Naive Bayes take a long time to train and test. However, it can be seen that SVM has low accuracy in comparison to Naive Bayes or Multinomial Naive Bayes in all representations and features. Moreover, it can be seen that from the consistent behavior of multinomial Naive Bayes shown with respect to each representation, and the graph in Figure 4.4, bi-grams alone are far from being competitive. Uni-grams and uni-gram+bi-gram as features and Naive Bayes and multinomial Naive Bayes as algorithms seem to have competitive results. However, again the fact that uni-grams are outperformed consistently by uni-grams and uni-gram+bigrams in both Naive Bayes and multinomial Naive Bayes compels one to believe that uni-grams+bi-grams is indeed a better feature. The same consistency of multinomial Naive Bayes outperforming Naive Bayes also compels one to conclude that Multinomial Naive Bayes is indeed better. I also did sporadic runs by varying the number of uni-grams and uni-gram+bi-grams under Naive Bayes and multinomial Naive Bayes which confirmed that Multinomial Naive Bayes using uni-gram+bi-grams under presence outperformed others. So not all possible combinations were exhaustively tried, but sporadic runs and the consistency of the results in the tables show the above conclusions are reliable.

all the results in the Tables 4.8, 4.9, 4.10 are done with 45000 training data (15000 tweets for each class) and 10,000 attributes. It is interesting to see how varying the number of attributes and training data while keeping one of them constant may affect the accuracy. Both of them are dealt differently below. But this time, the experimentation is done with multinomial Naive Bayes only since it is the one that gives the best result.

4.2.3.7 Varying number of attributes

In all the results reported in the previous section, the number of attributes was 10, 000. This was a common sense choice. It is interesting to vary the number of attributes keeping the training data constant and see if that improves perfor-

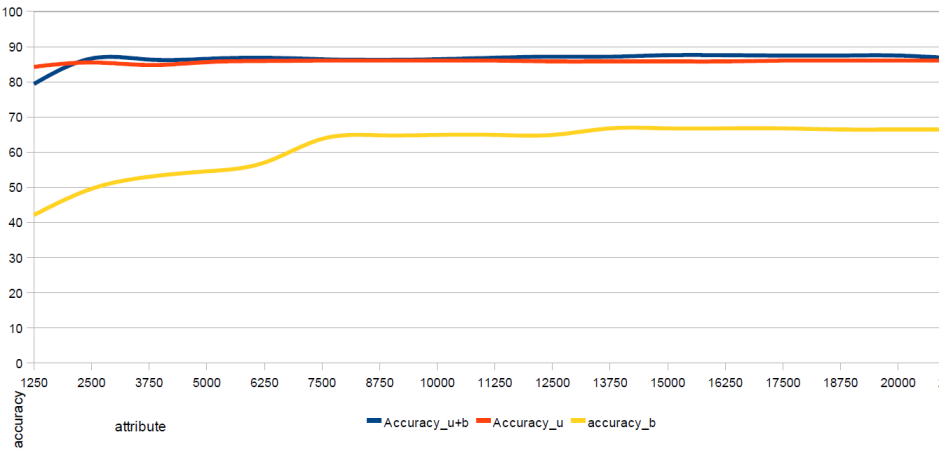


Figure 4.4: The effect of varying attributes for uni-gram, bi-grams and uni-grams+bi-grams

mance. The experiment showed that performance is not linearly proportional to the number of attributes. However, it showed that the best result out of the attributes seen in the graph was to be found at attribute numbers of 15,000 and 16,250. At both attribute numbers, an accuracy of 87.58% is recorded. The result of varying attributes is presented in the graph in Figure 4.4. In order to show and compare how uni-grams or bi-grams in isolation perform, they are presented in the graph. It was found that at almost any point in the graph, uni-grams or bi-grams in isolation performed less than uni-grams+bi-grams. The only time uni-grams outperformed uni-gram+bi-grams is at attribute number 1250. However that is not when the best result for uni-grams+bi-grams are recorded. This further confirms the claim that uni-grams+bi-grams combine the best of uni-grams(coverage) and bi-grams (sentiment pattern), and is the best feature for Twitter sentiment analysis.

The graph also shows that uni-grams outperform bi-grams at any point. The highest accuracy achieved with uni-grams is 86.02% , while the highest accuracy obtained with bi-grams is 66.77%. The highest accuracy achieved with uni-grams+bi-grams is 87.58%. The accuracy for uni-grams, bi-grams and uni-grams+bi-grams starts to decrease slowly after attribute number of 20,000.

4.2.3.8 Increasing the number of training data

we saw above that varying the number of attributes varies the accuracy. Initially, a total of 45000 tweets were collected to be used as training data. As the experimentation was being done, some more training data was collected. Now, it might be interesting to see how varying the number of training data affects accuracy. This was done keeping the number of attributes constant at 16250 (at which one of the best results was obtained). Increasing the number of training

data did lead to some increment of accuracy, although not in a linear fashion. An accuracy of 87.78% was obtained using 51,000 number of training data, but it started to decrease. Normally, the learning algorithm is expected to achieve better results as the number of training data is increased. But is accuracy going to increase indefinitely as the number of training data is increased? If yes, then we will reach 100% accuracy. This is a theoretical assumption, and it may not happen in reality nor are we going to prove it here. The interest here is to obtain as high accuracy as possible with the number of training examples that had been collected.

The maximum number of training data collected was 82491 (a sum of equal number of each class). This is after all the preprocessing operations including duplicate removal have been done. I took this training data and experimented with it by varying the number of attributes. However, there was no improvement in performance. The best performance in terms of accuracy remains the one achieved using 51000 number of training data and 16250 number of attributes, that is an accuracy of 87.78. The number

Chapter 5

Analysis and Conclusion

5.1 Analysis

5.1.1 Evaluation

Evaluation is a key by which credibility and improvement is measured. There are two ways of evaluating a classifier: cross-validation and test data. Cross-validation is used in case there is no adequate data for both training and testing or in the absence of a gold standard dataset for testing. In such a situation, some amount of data is held out for testing and the remaining is used for training. To ensure representativeness of training and test data, a method called stratification is used. Also, the cross-validation can be repeated, usually 10 times. This type of evaluation is called stratified 10-fold cross-validation(Witten and Frank, 2005). In this work, however, the situation is different: the training data and the test data are different in that the test data is not guaranteed to be of the type of tweets that are used in the training data. What is being done is this: train a classifier on tweets containing emoticons and use that classifier to determine sentiment of tweets about news. Moreover, there is no such shortage of data too.

5.1.2 Evaluation metrics

The diagram is taken from Manning and Schütze (1999) and is used to illustrate how the different metrics are defined in information retrieval (IR). In this diagram, there is the total data (the test data), let's say a collection of documents, represented by the rectangle. Let's say the IR system wants to retrieve a set of documents that fulfill some requirement. These set of documents are the 'target' (tp+fn). But the system actually retrieves the set of documents that are in the circle named 'selected' (fp+tp) in the diagram. tp stands for true positives, fp stands for false negatives and fn stands for false negatives. Those that are neither in the 'selected' nor in the targeted are true negatives (tn). From here, accuracy, precision, recall and F measure are computed as follows.

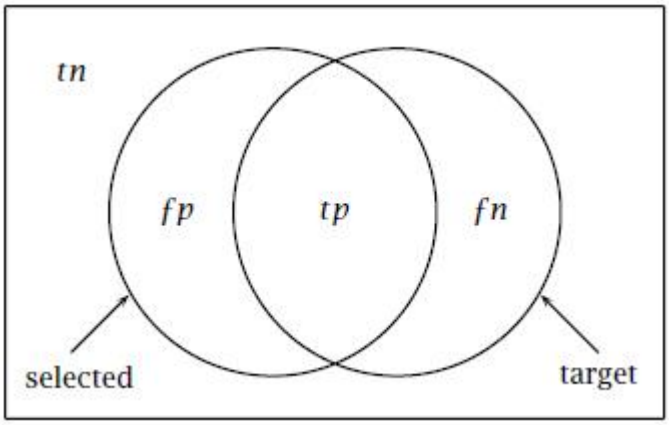


Figure 5.1: diagram for evaluation definition

$$accuracy = \frac{tp + tn}{tp + fp + fn + tn}$$

$$precision = \frac{tp}{tp + fp}$$

$$recall = \frac{tp}{tp + fn}$$

$$F = \frac{2 * precision * recall}{precision + recall}$$

So what do they each mean? Accuracy measures the proportion of documents that are correctly obtained, precision measures the exactness, that is, the percentage of selected documents that are the targeted documents, and recall measures the completeness, that is the proportion of targeted documents that the system selected. F measure is geometric mean of precision and recall. The measures of precision and recall are important performance measures when the dataset (set of documents) can be seen as relevant (the target) and irrelevant (outside the target). They are also useful measures to talk about the performance of a system when tn is so big (Manning and Schütze, 1999). There can be a trade off between precision and recall since one may be increased at the expense of the other. Depending on the nature of a NLP task, one may look for higher precision, recall, F measure or accuracy or some mixture of them all.

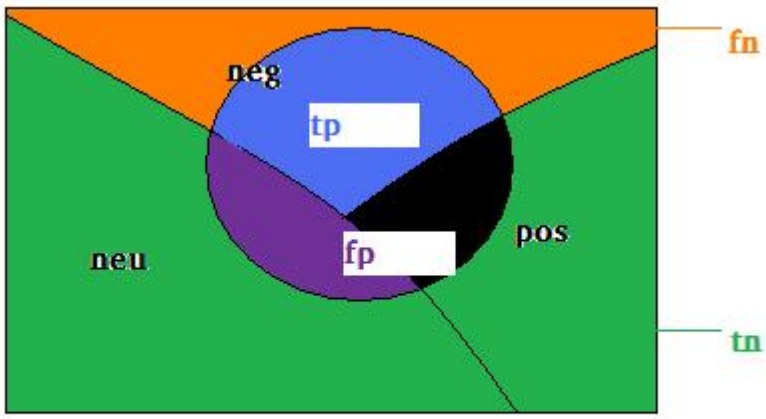


Figure 5.2: diagram for evaluation definition on classification

Now how do these performance measures apply in classification task, especially in a three-way or two-way classification that we have here. Again let's see a diagram with three classes neutral, negative and positive.

The rectangle (the whole test data) is divided into *neg*, *pos*, and *neu*. Let's say the neg class is now the target. The circle shows the selected. The circle does not exactly match the *neg*; it misses some part of the *neg*, and takes some parts from *neu* and *pos*. The part that is missed from *neg* is fn and the parts that are taken from *neu* and *pos* are fp. The parts that are outside the selected (the circle) and neg are tn. This is when we target the negative class. But, in three-way classification, there are three targeted classes. Thus there will be similar results for positive and neutral classes with their own tp, fn, fp and tn. The definitions of precision, recall, accuracy and F measure will be exactly like we saw above in terms of the tp, fp, fn, and tn. For a three-way classification, the results can be given in the form of a confusion matrix which is also called contingency table.

a	b	c	<− classified as
a:a	a:b	a:c	| a = negative
b:a	b:b	b:c	| b = positive
c:a	c:b	c:c	| c = neutral

The confusion matrix shows the actual and predicted classes. The rows show the actual class. To make it clear and to relate it with the diagram above, let's see for the negative class. There are $a:a + a:b + a:c$ total number of actual negatives. Out of them, the classifier got only $a:a$ correctly. It missed $a:b$ and $a:c$. Instead it classified $b:a$ and $c:a$ as negatives, which actually are positive and neutral respectively. $a:a$ is the tp in the diagram, $a:b$ and $a:c$

constitute fn, and $b:a$ and $c:a$ constitute fp. The rest, that is $b:b$, $c:c$, $b:c$, $c:b$ and $c:c$ constitute tn. Similarly, $b:b$ is the number of positives that the classifier got right, and $c:c$ is the number of neutrals that the classifier got right. Given this, precision, recall and accuracy can be redefined as below:

$$precision = \frac{number-of-correctly-classified-instances-of-x}{number-of-predicted-as-x}$$

$$recall = \frac{number-of-correctly-classified-instances-of-x}{number-of-actual-intances-x}$$

$$accuracy = \frac{numbe-of-correctly-classified-instances}{number-of-intances}$$

For the negative class, it means:

$$precision = \frac{a:a}{a:a+b:c+c:a}$$

$$recall = \frac{a:a}{a:a+a:b+a:c}$$

There will be precisions and recall for the positive and negative classes as well. Thus in a three-classed classification, there will be three different measures of precision, recall and F measure because each class is a different target. However, there is only one accuracy measure of the overall classification:

$$accuracy = \frac{a:a+b:b+c:c}{total}$$

where *total* is the sum of all cells in the confusion matrix. What this means is that we can not use precision and recall (and therefore F measure) to compare different classifiers. This is because we are trying to classify the dataset into three equally important classes and not into two relevant and irrelevant ones. Even when classifying the dataset into two equally important classes such as subjective and objective, the measures of precision and recall are still per individual class. That is why, I think, all of the papers reviewed in chapter 2 used accuracy for measuring performance and comparing different classifiers.

It might be possible to combine the recall and precision measures of each class to come up with an overall measure of recall and precision for the classifier. I have not come across any such measure in the papers I reviewed. However, precision and recall can tell us about some aspects of a classifier. Precision tells the exactness of the classifier with respect to each class and recall tells the completeness of the classifier with respect to each class. Recall enables

to identify the class with respect to which the classifier is having difficulty predicting and to use this info to tip the classifier in favor of that class.

Accuracy is the measure by which all the results of the above algorithms were compared. Error is the other way of talking about it. So if an algorithm has 80% accuracy, it means it has 20% error. Below is a result from WEKA for multinomial Naive Bayes on 51000 amount of training data and 966 test data. The accuracy is on the test data and this is the classification at which the highest accuracy was achieved.

=== Error on test data ===
Correctly Classified Instances 848 87.7847 %
Incorrectly Classified Instances 118 12.2153 %
Kappa statistic 0.6501
Total Number of Instances 966
=== Confusion Matrix ===

a	b	c	<− classified as
57	17	28	| a = negative
12	69	12	| b = positive
27	22	722	| c = neutral

The rows show the actual class. Thus, there are 102 negatives, 93 positives, and 771 neutrals in the actual classes. On the other hand, the columns show the predicted classes. Thus, there are 96 negatives, 98 positives and 762 neutrals in the predicted classes. For a perfect classifier (100% accuracy), there must be equal number of negatives, that is 102 in the first column and first row. The same with positives and neutrals. But here, we are dealing with reality and that is far from happening. The number of predicted class is different from the actual class. We need to measure this. This is measured by accuracy. The accuracy is computed as below.

In the table above, if every thing was perfect, the diagonal from top left to the bottom right would be 102, 93, 771. This is the perfect scenario. The values out of the diagonal are errors. Thus to compute accuracy, the number in the diagonals are added and divided by the total number of test instances (the total of all cells). Accordingly

$$accuracy \quad = \quad \frac{57 + 69 + 722}{966} = 848/966 = 87.7847\%$$

The accuracy is the same as the result reported in the 'correctly classified instances' in the WEKA output above. The accuracy is high, but let's put it into perspective. The baseline for comparison, it was also used in the keyword-based classifier, is the situation in which the classifier just predicts the most frequent class for every tweet in the test data. The most common class in the test data is neutral. There are 771 neutral tweets. So if the classifier assigns neutral class to every tweet, it will have an accuracy of 79.8%. Taking this into account, multinomial Naive Bayes classifier has done well above the baseline, achieving an improvement of about 8%. The same classifier was evaluated on 10-fold cross-validation and the accuracy obtained is 78.27%. It does not say

much about the accuracy on the test data since the test data is different from the training data.

There is more that we can learn from the confusion matrix. We can learn where the classifier is having difficulty by computing recall. the c:c cell (the cell at the bottom right) is where all the neutrals should be in a perfect scenario. Thus the cells a:c and b:c are telling us what the classifier got wrong. It is saying it wrongly predicted 27 neutrals as negative and 22 neutrals as positive. From this we can compute the per-class errors (the other way of talking about recall) as opposed to the overall accuracy computed above. Computing this error rate for each will tell us where the most difficulty of prediction lies.

$$\begin{aligned} error_{neg} &= \frac{17+28}{102} = 44.12\% \\ error_{pos} &= \frac{12+12}{93} = 25.80\% \\ error_{neu} &= \frac{27+22}{771} = 6.36\% \end{aligned}$$

For this particular example, the classifier has difficulty predicting negatives followed by positives. It is very successful with predicting neutral tweets.

5.1.3 Objective-subjective classification

Now, let's say we want to do a two-classed classification, that is classifying a tweet as either subjective or objective. An objective tweet, in this case, is the a neutral tweet, and a subjective tweet is either a positive or a negative tweet. In this thesis, the two-classed classification is done from the results of the three-classed classification. Namely, the tweets that are predicted to be negative or positive are counted to form a subjective tweet and the class that is neutral is the objective tweet. The confusion matrix for the high-achieving accuracy we saw above is reproduced in Table 5.1. The values that are in solid-boundary cells are the ones that have been classified belonging to the subjective class correctly. Table 5.2 shows the two-classed confusion matrix after regrouping and reducing the three-classed confusion matrix of Table 5.1 Then the accuracy for the two-classed matrix can be computed as:

$$\begin{aligned} accuracy &= \frac{155+722}{966} \\ accuracy &= 90.79\% \end{aligned}$$

This accuracy is higher than the accuracy of the three-classed classification. The reason is that when the three-classed confusion matrix is reduced to two-classed confusion matrix, the values of two cells that were wrong in the former become correct in the later. These two wrong values are those tweets that are actually negative but classified as positive (cell b:a in a row-column order), and those that are actually positive but classified negative (a:b in a row:column

a	b	c	<− classified as
57	17	28	| a = negative
12	69	12	| b = positive
27	22	722	| c = neutral

Table 5.1: Subjective -objective matrix a

a	b	<− classified as
155	40	| a = subjective
49	722	| b =objective

Table 5.2: Subjective-objective matrix b

order). The values are 17 and 12 respectively. These two values are responsible for the increase in the accuracy of the two-classed classification. Since $a : b+b : a \geq 0$, the accuracy of two-classed classification is always guaranteed to be greater than or equal to the accuracy of three-classed classification it is derived from. This can be generalized to mean text-classification into more general classes is easier than to detailed classes.

There is one more thing to consider when one makes two-classed classification from a three-classed classification. This is whether a high accuracy in the three-classed classification implies a high accuracy in the two-classed classification obtained from it. A high-accuracy three-classed classification is one which achieves higher values on the diagonal from top-left to bottom-right. In the two-classed classification, not only the diagonals, but also the values of cells a:b and b:a play a role in the computation of accuracy. Therefore a three-classed classification that has low accuracy can give a two-classed classification that has high accuracy because of the values of the two cells or a high-accuracy three-classed classification can give a low-accuracy two-classed classification. This mean the 90.79% accuracy achieved above might not be the highest two-classed accuracy one can get. However, because of time limitations, I did not explore in this direction.

5.1.4 Factors and challenges

5.1.4.1 Factors affecting performance

It is important to identify the factors that improve performance. Factors that affect performance are training data, attributes (features) selection, representation of instances, and choice of learning algorithm. A decision to make equal number of each class (positive, negative, and neutral) was made to avoid biasing the classifier in favor of any of the classes. Imagine the number of training instances in each class was not the same. Imagine, in particular, 2/3 of the training data is positive, and the rest 1/6 neutral and the other 1/6 negative. This means biasing the classifier in favor of positives since$P(pos) = 2/3$ and the whole formula becomes

$$P(pos|E) = \frac{P(Pos)P(E/Pos)}{P(E)} = \frac{2/3 * P(E/H)}{P(E)}$$

This doubles the probability of a new Twitter post being positive from the normal situation where all classes have probability of 1/3 when the number of training data is the same for each class. This is not to mention the effects that there will be in the probabilities of attributes during training. Surprisingly, it is also possible to use to improve accuracy. For example, if the classifier is showing weakness in predicting positive class, it can be biased to favor positives by increasing the number of positive tweets in the training data. Another way the training data can affect performance is by presence of noise. Noise can mean repetition of Twitter posts, having Twitter posts that are confusing such as those that contain both frowns and smiles, etc. That is why proper preprocessing was done before training. However, removal or replacement with equivalent class of user names, URLs, words starting with a hashtag (#), words staring with at (@) did not bring any significant change in accuracy.

Attribute Selection affects the accuracy that one can get from a classifier. One way that this can affect accuracy is in the use of uni-grams, bi-grams, uni-gram+bi-gram, or other features. Unlike uni-grams, bi-grams or other higher level n-grams capture some syntax. The bi-gram 'step down' is different from 'down steps'. The best results were obtained with uni-gram+bi-gram. The number of attributes to be selected also affects performance. Too few attributes can cause underfitting and too many attributes can cause overfitting. Underfitting is a situation where the classifier is not biased enough to make make prediction on unseen instances. Overfitting is a situation where the trained model is highly-fitted to the training data that it fails in predicting new instances successfully. Thus it is good to find a balance somewhere in between too few and too many attributes. Unfortunately, there seems to be no simple way of finding this balanced number of attributes except by trial and error.

The other factor that can affect performance is representation of instances. This includes whether to use presence(absence) or count(frequency). It also includes whether to convert count(frequency) into tf-idf to take terms importance into account. The best results for multinomial Naive Bayes were obtained using presence. Finally, the choice of algorithm for the task at hand affects performance. Multinomial Naive Bayes was found to outperform others in this sentiment analysis (classification) task.

5.1.4.2 Challenges

There are some challenges that were faced in this sentiment analysis work. One of the challenges was the collection of tweets about news. Collecting the tweets about news is a difficult task. This is because there is no easy way of doing it using Twitter API. To overcome this problem, a bootstrap approach where some seed tweets were used to obtain other more tweets was used. The seed tweets themselves were obtained by using some words from news headline. Once

this challenge was passed, another challenge was in manually annotating them. Manual annotation is challenging because it is difficult to decide whether a given tweet is positive or negative. This is made even more difficult because the tweets are about news, and therefore there is always possibility that the tweet is dependent on the context. To overcome this challenge, a common sense approach was used. So annotation is highly subjective. In this way, only few tweets were identified as requiting context to determine their sentiment. About 97% did not need context to understand them. However, it is important to note that even with the common sense appraoch, it is still difficult to differentiate between negative, positive, or neutral sentiments. This is especially so because neutral tweets contain sentiment-indicating words. Other reasons that make the annotation difficult is whether it is sarcastically said or not.

The fact that neutral tweets contain a lot of sentiment indicating words makes it difficult even for the machine learning algorithm . The basis for using data containing emoticons for training is the assumption that tweets that contain emoticons have the sentiment of the emoticons. However, it is not uncommon to find tweets that contain two opposite emoticons. Also, tweets can have more than one sentence. Both of this may mean more than one sentiment is expressed in one tweet. subordination also presents a challenge for accurate classification.

5.2 Conclusion

The thesis set out to solve a practical problem of sentiment analysis of Twitter posts about news. It started with discussions of motivations, theoretical framework and research question, importance of the research, aims and outcomes. Vast literature of general sentiment analysis, and few Twitter-specific sentiment analysis were discussed and possible approaches, techniques, features and assumptions for sentiment analysis of tweets about news were made. After this data collection were discussed. First tweets about news were collected and analyzed to see if context plays a role in determining sentiment. Then training data was collected.

After data collection, experiments with keyword-based classification and several best performing supervised machine learning techniques are done. Before experimentation with supervised techniques, different data preprocessing and feature extraction are done. Different data representations, features, number of attributes and number of training data were tried. Finally evaluation methods, subjective-objective classification, challenges and factors that affect performance are discussed.

This thesis has made some confirmations of previous finding and three main novel contributions. The confirmation are that machine learning techniques outperform keyword-based techniques and that term presence gives better results than other instance representations. The contributions are: data collection of tweets about news, empirical study of the role of context in sentiment analysis of tweets about news, and best feature selection. All training data (negative, positive, neutral) and test data were collected from Twitter using different

APIs. Negative and positive were collected from the streaming Twitter API; neutral tweets were obtained from Twitter accounts of major news outlets using streaming Twitter API. Test data was collected using Twitter Search API by a bootstrapping approach where some tweets that contain some number of words from the news headline were used to extract links and to obtain more tweets about news from Twitter. The collection of test data is where this thesis made a novel contribution.

The test data was manually inspected and annotated for its sentiment. Four classes were used to annotates it: negative, positive, neutral, and context. the 'context' class was assigned to tweets that seemed to require context to understand them. Only 3% were annotated with 'context'. The percentage of tweets requiring context to understand them are negligible in number, so sentiment analysis of Twitter posts about news can be done without regard to context. This empirical study indicates that sentiment analysis of tweets in general can be done independently without regard to their context. This is the second main contribution of this thesis for there was not any domain-specific study of sentiment of tweets before.

The third and very important contribution is in feature selection. A careful review of the literature on sentiment analysis showed that there is no one best feature vector that is suited to sentiment analysis. There are some sentiment analysis studies that achieved good results with uni-gram presence and other studies with bi-gram presence. Also there are sentiment analysis studies which indicated that incorporating syntax, negation and/or POS tags improved performance of sentiment analysis. After these observations, a novel feature vector that can combine all the features was adopted. This feature is the use of uni-gram+bi-grams. The strength of using uni-gram+bi-gram is that it captures two important features of the data: uni-grams provides better coverage of the data, and bi-grams capture sentiment expression patterns including those patterns that are captured by syntax, negation and POS tags. The use of uni-gram+bi-gram as feature for sentiment analysis of tweets proved to be indeed a better feature than other features during experimentation with different features and algorithms. Using uni-gram+bi-gram as feature gave better results than uni-gram or bi-gram in isolation.

To conclude, the thesis has shown that tweets about news can be automatically collected and successfully analyzed for their sentiment. Multinomial Naive Bayes classifier using uni-gram+bi-gram presence was found to give the highest accuracy. The highest accuracy achieved for a three-classed (negative, positive, neutral) classifier is 87.78%. The accuracy for the two-classed (subjective, objective) classifier derived from the three-classed classifier with highest accuracy is 90.79%. The accuracies of the classifier on both three-classed and two-classed classification is impressive and can be applied for practical applications dealing with sentiment analysis of Twitter posts about news in particular and Twitter posts in general.

Although the accuracies achieved with the classifier above are excellent and are higher than corresponding accuracy achievements in the literature, there might still be room for improvement. The attributes (uni-gram+bi-gram) used

are selected on the basis of frequency of occurrence in the entire training data. This means uni-grams and bi-grams compete on equal opportunity basis. However, it is obvious that uni-grams are more frequent than bi-grams and so only few very frequent bi-grams can make it to the the attributes. It would be interesting to experiment by varying the ratio of bi-grams to uni-grams in the attribute set. It would also be interesting to experiment with other attribute selection methods such as chi-square, inforgain, etc. instead of frequency. It would also be interesting to experiment with attribute selection on the basis of each class rather than from the entire training corpus. Finally, it would be interesting to see if other two-classed classifiers that can achieve accuracies higher than 90.79% can be derived. Note that the 90.78% accuracy of the two-classed classifier that was obtained from the highest-accuracy-achieving three-classed classifier might not be the highest as discussed in the objective-subjective section above.

Bibliography

L. Barbosa and J. Feng. Robust sentiment detection on Twitter from biased and noisy data. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 36–44. Association for Computational Linguistics, 2010.

M. Bautin, L. Vijayarenu, and S. Skiena. International sentiment analysis for news and blogs. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, 2008.

V.L. Corpora. Empirical Methods in Natural Language Processing.

S. Das and M. Chen. Yahoo! for Amazon: Extracting market sentiment from stock message boards. In *Proceedings of the Asia Pacific Finance Association Annual Conference (APFA)*, 2001.

K. Dave, S. Lawrence, and D.M. Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528. ACM, 2003. ISBN 1581136803.

A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 2009.

N. Godbole, M. Srinivasaiah, and S. Skiena. Large-scale sentiment analysis for news and blogs. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*. Citeseer, 2007.

V. Hatzivassiloglou and K.R. McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computa tional Linguistics*, pages 174–181. Association for Computational Linguistics, 1997.

B. Liu, X. Li, W.S. Lee, and P.S. Yu. Text classification by labeling words. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 425–430. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004.

C.D. Manning and H. Schütze. *Foundations of statistical natural language processing*, volume 59. MIT Press, 1999.

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

J.C. Na, H. Sui, C. Khoo, S. Chan, and Y. Zhou. Effectiveness of simple linguistic processing in automatic sentiment classification of product reviews. *ADVANCES IN KNOWLEDGE ORGANIZATION*, 9:49–54, 2004. ISSN 0938-5495.

A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. *Proceedings of LREC 2010*, 2010.

B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008. ISSN 1554-0669.

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

R. Prabowo and M. Thelwall. Sentiment analysis: A combined approach. *Journal of Informetrics*, 3(2):143–157, 2009. ISSN 1751-1577.

J. Read. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL Student Research Workshop*, pages 43–48. Association for Computational Linguistics, 2005.

P.D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 417–424. Association for Computational Linguistics, 2002.

Unknown. Statistics tutorial: Multinomial distribution. `http://stattrek.com/Lesson2/Multinomial.aspx`.

T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354. Association for Computational Linguistics, 2005.

I.H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann Pub, 2005. ISBN 0120884070.

A. Wright. Our sentiments, exactly. *Communications of the ACM*, 52(4):14–15, 2009. ISSN 0001-0782.

H. Yang, L. Si, and J. Callan. Knowledge transfer and opinion detection in the TREC2006 blog track. In *Proceedings of TREC*, volume 120. Citeseer, 2006.

Appendix

Keyword lexicons

positive keyword Woo, quite amazing, thks, looking forward to, damn good, frickin ruled, frickin rules, Way to go, cute , comeback, not suck, prop, kinda impressed, props, come on, congratulation, gtd, proud, thanks, can help, thanks!, pumped, integrate, really like, loves it, yay, amazing, epic flail , flail, good luck, fail, life saver, piece of cake, good thing, hawt, hawtness, highly positive, my hero, yummy, awesome, congrats, would recommend, intellectual vigor, really neat, yay, ftw, I want, best looking, imrpessive, positive, thx, thanks, thank you, endorse, clearly superior, superior, really love, woot, w00t, super, wonderful, leaning towards, rally, incredible, the best, is the best, strong, would love, rally, very quickly, very cool, absolutely love, very exceptional, so proud, funny, recommend, so proud, so great, so cool, cool, wowsers, plus, liked it, make a difference, moves me, inspired, OK, love it, LOL, :), ;), :-), ;-), :D, ;], :], :p, ;p, voting for , great, agreeable, amused, brave, calm, charming, cheerful, comfortable, cooperative, courageous, delightful, determined, eager, elated, enchanting, encouraging, energetic, enthusiastic, excited, exuberant, excellent, I like, fine, fair, faithful, fantastic, fine, friendly, fun , funny, gentle, glorious, good, pretty good, happy, healthy, helpful, high, agile, responsive, hilarious, jolly, joyous, kind, lively, lovely, lucky, nice, nicely, obedient, perfect,pleasant, proud, relieved, silly, smiling, splendid, successful, thankful, thoughtful, victorious, vivacious, witty, wonderful, zealous, zany, rocks, comeback, pleasantly surprised, pleasantly, surprised, love, glad, yum, interesting

Negative Keywords FTL, irritating , not that good, suck, lying, duplicity, angered, dumbfounding, dumbifying, not as good, not impressed, stomach it, pw, pwns, pwnd, pwning, in a bad way, horrifying, wrong, flailing, failing, fallen way behind, fallen behind, lose, fallen, self-deprecating, hunker down, duh, get killed by, got killed by, hated us, only works in safari, must have ie, fuming and frothing, heavy, buggy, unusable, nothing is, is great until, don't support, despise , pos, hindrance, sucks, problems, not working, fuming, annoying , frothing, poorly, headache, completely wrong, sad news, didn't last, lame, pet peeves, pet peeve, can't send, bullshit, fail, so terrible, negative, anooying, an issue, drop dead, trouble, brainwashed, smear, commie, communist, anti-women, WTF,

anxiety, STING, nobody spoke, yell, Damn, aren't , anti, i hate, hate, dissapointing, doesn't recommend, the worst, worst, expensive, crap, socialist, won't, wont, :(, :-(, Thanks, smartass, don't like, too bad, frickin, snooty, knee jerk, jerk, reactionist, MUST DIE, no more, hypocrisy, ugly, too slow, not reliable, noise, crappy, horrible, bad quality, angry, annoyed, anxious, arrogant, ashamed, awful, bad, bewildered, blues, bored, clumsy, combative, condemned, confused, crazy, flipped-out, creepy, cruel, dangerous, defeated, defiant, depressed, disgusted, disturbed, dizzy, dull, embarrassed, envious, evil, fierce, foolish, frantic, frightened, grieving, grumpy, helpless, homeless, hungry, hurt, ill, itchy, jealous, jittery, lazy, lonely, mysterious, nasty, rape, naughty, nervous, nutty, obnoxious, outrageous, panicky, fucking up, repulsive, scary, selfish, sore, tense, terrible, testy, thoughtless, tired, troubled, upset, uptight, weary, wicked, worried, is a fool, painful, pain, gross