Twitter Sentiment Analysis

CMPS 242 Project Report

Shachi H Kumar
University of California Santa Cruz
Computer Science
shachihkumar@soe.ucsc.edu

ABSTRACT

Twitter is a micro-blogging website that allows people to share and express their views about topics, or post messages. There has been a lot of work in the Sentiment Analysis of twitter data. This project involves classification of tweets into two main sentiments: positive and negative. In this project, the use of features such as unigram, bigram, POS tagging, and effects of data pre-processing like stemming is observed. Naive Bayes, Support Vector Machines(SVM) and Maximum Entropy(MaxEnt) are used as the main classifiers. As we shall see in the sections below, SVM with features of unigrams, bigrams and stemming, outperforms Naive Bayes.

1. INTRODUCTION

Sentiment analysis is the task of finding the opinions and affinity of people towards specific topics of interest. Be it a product or a movie, opinions of people matter, and it affects the decision-making process of people. The first thing a person does when he or she wants to buy a product online, is to see the kind of reviews and opinions that people have written. Social media such as Facebook, blogs, twitter have become a place where people post their opinions on certain topics. The sentiment of the tweets of a particular subject has multiple usage, including stock market analysis of a company, movie reviews, in psychology to analyze the mood of people that has a variety of applications, and so on.

Sentiments of tweets can be categorized into many categories like positive, negative, neutral, extremely positive, extremely negative, and so on. The two types of sentiments considered in this classification experiment are positive and negative sentiments. The data, being labeled by humans, has a lot of noise, and its hard to achieve good accuracy. Currently, the best results are obtained by Support Vector Machine(SVM) for a feature set containing stemming, and Bigram that gives an accuracy of 82.55. The main algorithms used in this project are SVM and Naive bayes and we would be comparing these in the upcoming sections.
The report is organized in the following way. Section 2 talks about the related work done in this area. Section 3 describes the twitter data that was used in this project. Description of the data include the statistical details, as well as the datasets used for testing and training.
Section 4 is a detailed description about the methodology used. The four main important topics include data pre-processing, the machine learning algorithms used, the tools required to execute the project as well as the feature extraction techniques along with features used. Section 5 reports the results thus far obtained based on the data processing performed and the algorithms used. The section 5 is based on two sets of data, one is the smaller set of data on which all the feature set was tested. The other set of data includes the entire dataset of 1.5million tweets. Only a few details on this dataset is reported as the data is very huge to be run even on high memory machines. Section 6 talks about the future work to be done in this area. The appendix contains few of the tests performed which did not add much value to the classification results, including numeric features and tf-idf.

2. RELATED WORK

Research work in the area of Sentiment analysis are numerous. Some of the early results on Sentiment Analysis of twitter data are by Go et al. who used distant learning to acquire sentiment data. They used tweets with positive emoticons like ":)" and ";)" as positive, and tweets with negative emoticons like ":(" as negative. Sentiment Analysis on twitter data has been done previously by Go et al. where they have built the model using Naive Bayes, MaxEnt and SVM classifiers, where they report SVM is better than all other classifiers. On the features, they have used Unigram, Bigram, along with Part-of-speech (POS) tagging. They note that unigram feature outperforms all other models and also mention that bigrams and POS tagging does not help. They also perform some pre-processing of the data that was used in modeling the pre-processing techniques used in this project. The text processing they perform includes removal of URLs, username references and repeated characters in words.

A survey report from Pang and Lee on Opinion mining and sentiment analysis [4] gives a comprehensive study in the area with respect to sentiment analysis of blogs, reviews etc. Algorithms used in the survey include Maxium Entropy , SVM and Naive Bayes.

As the twitter data is noisy with lot of slang and short words some of the pre-processing techniques using the slang dictionary is mentioned in the paper Apoorva et al., along with it removal of the stop words. They also use the emoticon dictionary which has been implemented in this project to be used in numeric features. They also implement a prior polarity scoring which scores many English words between 1(Negative) to 3(Positive). From the algorithm point of view they provide a tree kernel and feature based models. Unigram baseline model is combined with other features and modeled in this paper. Different combination of the features are selected. Part-of-speech tagging and emoticons list from wikipedia are used for the features.

3. DATA

Tweets are short length messages and have a maximum length of 140 characters. This limits the amount of information that the user can share with every message. Due to this reason, users use a lot of acronyms, hashtags, emoticons, slang and special characters. Acronyms and slang such as *2moro* for *tomorrow* and so on are used to keep sentences within the word limit. People also refer to other users using the @ operator. Users also post URLs of webpages to share information. Emoticons are a great way to express emotions without having to say much. More details on these are explained in the next section.
The data used for this project is based out of Sentiment140 and contains about 1.5 million classified tweets, each row is marked as 1 for positive sentiment and 0 for negative sentiment.

More details about the data are as below:

Table 2: Data Statistics

Type	***Count***
Positive tweets	790185
Negative tweets	788440
Positive Emoticons	14727
Negative Emoticons	6275
Total words	20952530
Total words without stop-words	13363438
Stop words	7589092

Along with the twitter data, the project also required other datasets like stopwords[1] , a dictionary of negative and positive words[2], an emoticon dictionary[3] and an acronym dictionary for twitter slang words[4]. The use of these are described in the next section.

Dictionary of negative and positive words.

The dictionary of negative and positive words is a dataset containing around 6800 negative and positive words. This dataset is used to determine the numeric features of number of negative and positive words in the tweets, based on which sentiment classification is done. The process of stemming, as explained below is also performed on this dataset, so that it maps to the training and test dataset. Some negative and positive words from the dataset are shown as below :

[1]nltk.corpus.stopwords

[2]`http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html`

[3]`http://en.wikipedia.org/wiki/List_of_emoticons`

[4]`http://www.noslang.com`

Table 3: Negative and positive words dataset

Type	***Count***
abnormal	Negative
bothered	Negative
dangerous	Negative
dejection	Negative
aspirations	Positive
excited	Positive
fun	Positive
genuine	Positive
happiness	Positive

Emoticons.

Emoticons are a great way to express emotions, especially given the restriction on the length of tweets. Emoticons also form an effective way in determining the sentiment of the tweet. In this project, the emoticons are used as numeric features - positive and negative emoticons. Some of the positive and negative emoticons are shown in the table below.

Table 4: Negative and positive emoticons

Type	***Emoticons***
Negative Emoticons	:-/ : :'(:[= :/ :@ :'-(:c ;(=/ v.v
Positive Emoticons	:-| =p :] :-P ;) :p :3 =] :b :-) 8) ø/ :') ;-) :-p :S

The dataset was used in parts and in stages. In the beginning stage, a training set of size 60000 tweets and test set of size 40000 tweets was used. This enabled validation, as well as helped in tuning parameters for the algorithm. For example, while using Linear SVM, the parameter C was tuned to get maximum accuracy. The default value of C was 1 and the value of this parameter giving maximum accuracy was found to be 0.032.
For the final experiment, the entire data set of 1.5 million tweets was used with 75% of the data was used for training and 25% for testing. At different steps of pre processing, the data was tested using machine learning algorithms such as Naive Bayes, SVM and MaxEnt, the results of which are discussed in Section 5.

4. METHODOLOGY

The main approach involved in this project are the various data pre-processing steps, the machine learning classifiers and feature extraction. The main machine learning algorithms used are Naive Bayes, Support Vector Machines(SVM) and Maximum Entropy(MaxEnt). The main data pre-processing steps include URL and username filtering, twitter slang removal, stopwords removal and stemming. Feature extraction includes POS tagging, unigram, bigram (all the above in various combinations) and numeric features all of which are described below.

4.1 DATA PRE-PROCESSING

4.1.1 Filtering[2]

Item ID	*Sentiment*	*Sentiment Source*	*Sentiment Text*
106	0	Sentiment140	really wanted Safina to pull out a win; to lose like that...
166	1	Sentiment140	..... hot choco is the best!
107	0	Sentiment140	”RIP, David Eddings.”
174	1	Sentiment140	” :-D))).. What an amazin night! Miss u guys!”

Table 1: Data

URLs.
People use twitter not only for expressing their opinions but also for sharing information with others. Given the short maximum length of tweets, one way of sharing is using links. Tweets include various links or URLs and these do not contribute to the sentiment of the tweet. The URLs in the data used in this project are of the form *http://plurk.com/p/116r50*. These do not contribute to the sentiment of the tweet. Hence these were parsed and replaced by a common word, URL.

Usernames.
Tweets often refer to other users and such references begin with the @ symbol. These again do not contribute to the sentiment and hence are replaced by the generic word USERNAME.

Duplicates or repeated characters.
People use a lot of casual language on twitter. For example, ’happy’ is used in the form of ’haaaaaaappy’. Though this implies the same word ’happy’, the classifiers consider these as two different words.

Table 5 :Data Filtering

Tweets containing	***Replaced by***
http://plurk.com/p/116r50	URL
@reeta	USERNAME
cooooooooool	cool
baaaaaad	baad

To improve this and make words more similar to generic words, such sets of repeated letters are replaced by two occurrences. Thus haaaaappy would be replaced by haappy.

4.1.2 Twitter slang removal

As mentioned in the previous statement, tweets contain a lot of casual language. Also, given that the maximum length of a tweet is 140 characters, people tend to use abbreviations or some short forms for words. These short words are replaced by the actual words that they represent to improve performance of the learning algorithms.

Twitter Slang	*Actual word*
2gethr	Together
bff	best friend forever
1dering	Wondering
2moro	Tomorrow
2morrow	Tomorrow
tomo	Tomorrow
tmoro	Tomorrow
lol	laugh out loud

The advantage of doing this is evident from the above table. The word *Tomorrow* is used by people using many short forms like 2moro, 2morrow, tomo, tmoro and so on. If these are not mapped to the common original word, then training on them would not produce good accuracy and may also cause overfitting, as these might not be found in the test data.

4.1.3 Stop-words removal

In information retrieval, there exists many words that are added as conjunctions in sentences. For example, words like *the, and, before, while*, and so on do not contribute to the sentiment of the tweet. Also these words do not help in classifying the tweets as they appear in all classes of tweets. These words are removed from the data so as to avoid using them as features.

The stopwords corpus was obtained from NLTK. Some modifications were required to this as the corpus also had some negative words such as *nor, not, neither* which are important in identifying negative sentiments and should not be removed.

4.1.4 Stemming

In information retrieval, stemming is the process of reducing a word to its root form. For example, *walking, walker, walked* all these words are derived from the root word *walk*. Hence, the stemmed form of all the above words is *walk*.
NLTK provides various packages for stemming such as the PorterStemmer, LancasterStemmer and so on. The PorterStemmer was used in this project which uses various rules for suffix stripping. In addition to stemming the train and test data, the positive and negative word corpus was also stemmed. Stemming reduces the feature space as many derived words are reduced to the same root form. Multiple features now point to the same word and hence it increases the probability of the word.

Table 7: Stemming

Original words	*Stemmed word*
amazed	amaze
amazing	amaze
amazement	amaze

As we will see in the results section, stemming gives a good increase in accuracy. By stemming, different derived words are mapped to their root words and this allows more matching between the tweets in the test and training set.

4.2 MACHINE LEARNING ALGORITHMS USED

4.2.1 Baseline

This experiment uses Naive Bayes with Unigrams as a baseline.

4.2.2 Naive Bayes

The Naive Bayes classifier [5] is one of the basic text classification algorithms. It is a simple classifier based on Bayes theorem and makes naive independence assumptions of the feature variables. Despite this very naive assumption, it is seen to perform very well in many real-world problems.
Mathematical representation: Consider attributes $X_1, X_2....X_n$ to be conditionally independent of each other given a class Y. This assumption gives us,

$$P(X_1....X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)$$

By Bayes theorem, we have,

$$P(Y|X_i) = \frac{P(X_i|Y)P(Y)}{P(X_i)}$$

Using Bayes theorem in the previous equation, we can find the problability of predicting the class Y given the features X_i. The class that gives the maximum probability that the given features predict it, is the class that the tweet will belong to.

In this experiment, the NaiveBayesClassifier from NLTK was used to train and test the data.

4.2.3 SVM

Support Vector Machines [6] is another popular classification technique. A support vector machine constructs a hyperplane or set of hyperplanes in a high-dimensional space such that the separation is maximum. This is the reason the SVM is also called the maximum margin classifier. The hyperplane identifies certain examples close to the plane which are called as support vectors. LinearSVC from sci-kit learn, which is a python package, is used to classify the tweets.

4.2.4 MaxEnt

The Max Entropy classifier is a discriminative classifier commonly used in Natural Language Processing, Speech and Information Retrieval problems. The max entropy classifier uses a model very similar to the Naive bayes model but it does not make any independence assumption, unlike Naive Bayes. The max Ent classifier is based on the principle of maximum entropy and from all the models, chooses the once which has the maixmum entropy. The goal is to classify the text(tweet, document, reviews) to a particular class, given unigrams, bigrams or others as features. If $w_1, w_2....w_m$ are the words that can appear in a document, according to bag-of-words model, each document can be represented by 1s and 0s indicating if the word w_i is present in the document or not.
The parametric form of the MaxEnt model can be represented as below:

$$P(c|d, \lambda) = \frac{exp[\sum_i \lambda_i f_i(c, d)]}{\sum_c [\sum_i \lambda_i f_i(c, d)]}$$

Here, c is the class to be predicted, d is the tweet, and λ is the weight vector. The weight vector defines the importance of a feature. Higher weight means that the feature is a strong indicator for the class c. The parameters are chosen by iterative optimization, and for the same reason, this classifier takes a long time to learn when training size, features are large.

4.3 FEATURE EXTRACTION

4.3.1 Unigram

Unigrams are the simplest features that can be used for learning tweets. The bag-of-words model is a powerful technique in sentiment analysis. This technique involves collecting all words in the document and using them as features. The features can either be the frequency of words, or simply 0s and 1s to indicate if the word is present in the document or not. In this project, 0s and 1s are used to indicate the absence or presence of a word in the tweet.

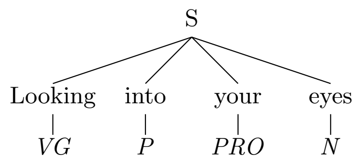
4.3.2 Bigram

Bigrams are features consisting of sets of two adjacent words in a sentence. Unigram sometimes cannot capture phrases and multi-word expressions, effectively disregarding any word order dependence. For example, words like 'not happy', 'not good' clearly say that the sentiment is negative, but a unigram might fail to identify this. In such cases, bigrams help in recognizing the correct sentiment of the tweet.

4.3.3 POS tagging

Part-of-speech tagging in linguistics and information retrieval is the process of tagging each word in a sentence to a particular part of speech. There are many parts of speech such as noun, adjective, pronoun, preposition, adverb, and so on. A word can take different meanings in different sentences, i.e a word can act as a noun in one sentence, and as an adjective in another.
For this project, the tagger model $maxent_treebank_pos_tagger$ provided by NLTK was used. Below tree shows a POS-tagging.

NLTK POS tagging:



4.4 TOOLS

4.4.1 Natural Language Toolkit

The NLTK is platform for building python programs to work with text data. It provides a variety of corpora and resources and various libraries for text classification, tagging, stemming, tokenization and parsing. In this project, NLTK was used extensively for tokenizing (tokenizing the tweets), POS tagging, the tagger model being maxent_treebank_pos_tagger, stemming (as described above, it used the PorterStemmer of NLTK), and classification.

The NLTK classifiers used were NaiveBayesClassifier and the MaxentClassifier.

4.4.2 *IPython*

Ipython is a command shell for interactive computing mainly for Python. Few of its main features are its input history across sessions, tab completion, support for visualization and use of GUI tool kits. The IPython also offers a rich text web interface called the IPython notebook. This project used IPython and IPython notebook extensively for data processing, learning, analysis and visualization, the results of which are discussed in the next section.

4.4.3 *Amazon Elastic Compute Cloud*

The Amazon Elastic Compute Cloud or the EC2 is the main component of Amazon's cloud computing platform, Amazon Web Services(AWS)[5]. It is a web service that allows users to rent virtual machines to run their applications. To use the large amount of data available for this sentiment analysis task, a high memory, high CPU system was required which led to the need for virtual machines on EC2. For this project, the machine with the following configuration was used extensively: m2.2xlarge: 34.2 GiB of memory, 13 EC2 Compute Units (4 virtual cores with 3.25 EC2 Compute Units each), 850 GB of local instance storage, 64-bit platform.

4.4.4 *Pandas*

Pandas is a software library written for Python and is used for data analysis and manipulation[6]. Pandas is an open source library and it also interoperates with the IPython and other Python libraries.

4.4.5 *Sci-kit Learn*

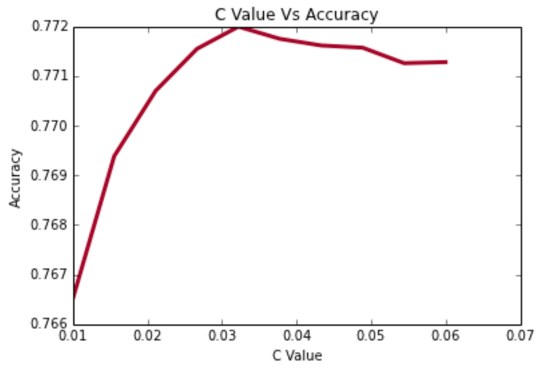
Scikit-Learn is an open source machine learning library for the Python programming language[7].It features various classification, regression and clustering algorithms including support vector machines, logistic regression, naive Bayes, random forests, gradient boosting and k-means, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. In this project, Sci-kit Learn was mainly used for the SVM classifier, in particular, the Linear SVC.

5. RESULTS

The first set of results show the analysis of the feature set and algorithms for the smaller dataset of 60000 training tweets and 40000 test tweets. When the entire dataset is used, the accuracy's are scaled up to a great extent. In general terms, below is the comparison obtained for the smaller dataset.

5.1 Parameter tuning

The Linear SVC classifier has various parameters that can be tuned depending on the noise in the data. The default value of C is 1. The tuning of parameter C is done while training with smaller set of the data. This was done by running the LinearSVC classifier for multiple C values, the output of which is shown below:



We see that the maximum accuracy is obtained when C =0.032. Hence, all the results that are shown in this section use the value of C as 0.032.

5.2 Results with smaller dataset

The following are the results obtained by data processing, analysis and visualization on a smaller portion of the data, i.e using 60900 tweets for training and 44000 tweets for testing.

5.2.1 Result Analysis using Naive Bayes:

Baseline.

The Baseline, i.e Naive Bayes with Unigram gives more details on the most informative features after the classifier was run. The below table shows these details.

Table 9 Naive Bayes (Unigram) most informative features

throat	neg : pos = 52.5 : 1.0
sad.	neg : pos = 47.0 : 1.0
sad!	neg : pos = 27.4 : 1.0
welcome!	pos : neg = 25.2 : 1.0
rip	neg : pos = 24.8 : 1.0
followfriday	pos : neg = 22.5 : 1.0
cancelled	neg : pos = 21.5 : 1.0
sad	neg : pos = 20.1 : 1.0
congratulations	pos : neg = 19.8 : 1.0
:'(	neg : pos = 17.6 : 1.0

Effect of Stopwords:.

The algorithms were first run on the dataset without any data preprocessing. When Naive Bayes was run, it gave an accuracy of 73.45 percent, which is considered as the baseline result. The next thing used was stopword removal. When stopwords were removed and Naive Bayes was run, it gave an accuracy of 73.67 percent.

Table 10: Effect of Stop words

Algorithm	*Accuracy*
Naive Bayes Unigram	73.45
Naive Bayes Stopwords removed	**73.67**

The results are almost identical, this was the case even with Linear SVC. This shows that stopwords do not really affect

[5]http://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud
[6]http://en.wikipedia.org/wiki/Pandas_(software)
[7]http://en.wikipedia.org/wiki/Scikit-learn

Table 8: Results with small dataset

Algorithm	**Unigram**	**Unigram and Bigram**	**Stemming Unigram**	**Stemming Bigram**	**POS Unigram**	**POS bigram**	**cleaned Data and Slang removal**	**Slang removal, stemming and Bigram**
Naive Bayes	73.45	75.12	75.35	77.10	75.10	76.88	74.38	77.01
SVM	77.13	78.29	77.29	78.95	77.68	78.63	77.62	**79.63**

the predictions much. An intuition to this can be obtained from the fact that given the short length of tweets, people generally avoid the use of stopwords such as and, while, before, after and so on. Thus removal of stopwords does not make a lot of difference to the accuracy.

Effect of Bigram as a feature:.
Bigram uses a combination of two words as a feature. Bigram effectively captures some features in the data that unigram fails to capture. For example, words like 'not happy', 'not good' clearly say that the sentiment is negative. This effect can be clearly seen from the increase in accuracy from 73.45(Unigram) to 75.12 percent which is almost a 2% increase. The below table gives the most informative features for Naive Bayes with Bigrams as features.

Table 11: Naive Bayes (Bigram)

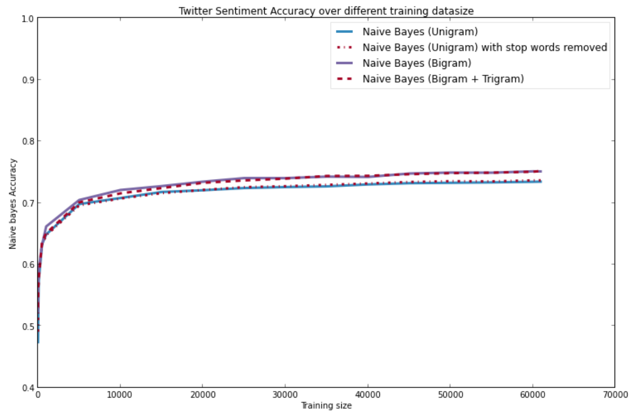
throat	neg : pos = 52.5 : 1.0
sad.	neg : pos = 47.0 : 1.0
('that', 'sucks')	neg : pos = 37.3 : 1.0
('lost', 'my')	neg : pos = 36.0 : 1.0
('once', 'you')	pos : neg = 36.0 : 1.0
('you', 'add')	pos : neg = 34.6 : 1.0
('so', 'sad')	neg : pos = 32.0 : 1.0
("i'm", 'stuck')	neg : pos = 30.1 : 1.0
sad!	neg : pos = 27.4 : 1.0
welcome!	pos : neg = 25.2 : 1.0

Effect of using Trigrams:.
Running Naive Bayes using Trigrams bigrams and unigrams together gave an accuracy of 75.19 percent which is almost the same as the accuracy obtained when Bigrams were used as a feature. Also this feature combination bloats up the feature space exponentially and the execution becomes extremely slow. Hence for further analysis, the trigrams are not considered as they do not have a noticeable impact on the accuracy.

Table 12: Effect of Trigram

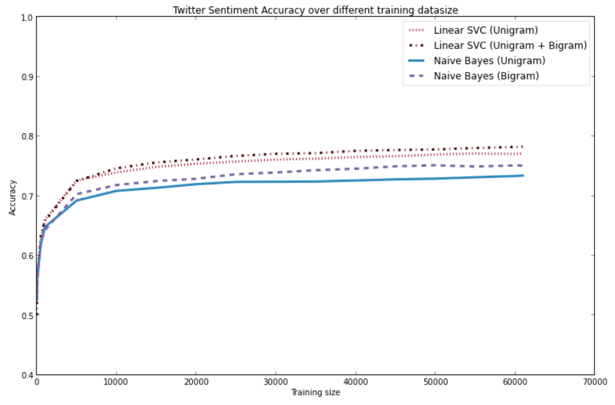
Algorithm	*Accuracy*
Naive Bayes Bigram	75.12
Naive Bayes Trigram,Bigram	**75.19**

The below graph shows very clearly the effect on accuracy when stopwords and trigrams are considered. The dashed line shows the bigram, trigram combination which is almost close to the accuracy for bigram. The dash-dot line shows the effect of stop words on the accuracy. As in the figure, this almost aligns with the Naive Bayes unigram accuracy.



Effect of Stemming:.
As can be seen in the Results table, stemming has a remarkable effect on the features used. When Unigram is used as a feature with stemmed words, the accuracy of Naive Bayes increases from 73.45 percent to 75.35 percent, an increase of almost 2 percent. On similar lines, for bigram features, stemming increases the accuracy from 75.12 percent to 77.10 percent. The reason, as mentioned above, is that stemming reduces the feature space as many derived words are reduced to the same root form and multiple features point to the same word, thus increases the probability of the word. In case of SVM, though the increase is not substantial with stemming, the accuracy does improve by a small value.

The below graph shows a comparison of accuracies of Naive Bayes and SVM for the different feature set combinations.



Data filtering and slang removal as an independent feature

Table 13: Results with full data

Algorithm	Unigram	Stopwords removed	Unigram and Bigram	Stemming Unigram and Bigram
Naive Bayes	76.39	76.98	78.65	79.46
SVM	80.02	79.22	82.07	**82.55**

does give a small improvement in accuracy with an increase from 73.45 percent to 74.3 percent in case of Naive Bayes and increase from 77.13 percent to 77.62 percent in SVM.

With all the features considered, the results show that SVM outperforms Naive Bayes in all cases. In particular, the feature combination of Slang removal, stemming and Bigram gives the maximum accuracy of 79.63 with SVM.

Maximum Entropy model gives an accuracy consistently in-between Naive Bayes and SVM. Also it runs iteratively and takes a large amount of time to run. Hence MaxEnt was not used for all the feature combinations.

5.3 Results with entire dataset

As the table shows, when the processing, analysis was done on the bigger dataset, the accuracy scaled up to a great extent. Naive Bayes baseline scaled up to 76.39 and SVM scaled up to 80.02 percent. The best result tested thus far, was obtained when SVM was used on a feature set of a combination of Unigram, Bigram with stemming, giving an accuracy of 82.55. MaxEnt also performed well and gave an accuracy of 77.18 when stopwords was removed.

6. FUTURE WORK

6.1 Multi-class classification

Till now, I have only dealt with binary classification of tweets, either as positive or negative sentiment. There are many tweets, for instance, those with URL's which do not have any sentiment, or, are neutral. These tweets are mainly for sharing some useful information with people, and not necessarily for raising an opinion. As a part of my future work, I would like to explore multi-class classification into various levels of sentiment such as Extremely positive, positive, neutral, negative and extremely negative.

6.2 More numeric features

The numeric features that were used in this experiment include number of negative and positive words, emoticons, length of tweets and number of special characters such as exclamations, hashtags and so on. The numeric features did not yield good accuracy and gave around 63 percent accuracy. Hence, as a part of my future work on this, I would like to generate more as well as smarter numeric features.

6.3 Use more classifiers

In this project, Naive Bayes, SVM and MaxEnt were used extensively. I would also like to explore other machine learning algorithms like Artificial Neural networks. Also generation of more numeric features will allow me to use more binary classifiers such as logistic regression and so on.

6.4 Use Hadoop Framework

As we have seen, using more data scales up the accuracy. But despite using the amazon EC2, the data still turned out to be very memory and CPU intensive and I was unable to run the SVM and Naive Bayes with POS Tagging and also unable to run MaxEnt algorithms on it. As a part of the future work, I would like to make use of Hadoop for processing large data of this kind.

6.5 Use of Twitter based tagger

I have made use of the tagger model provided by NLTK in this project. There is also a twitter-specific POS tagger which is the Penn Treebank-style tagset for Twitter[8]. The tagger was trained from a fixed version of Ritter et al. EMNLP 2011's annotated data. It has been updated as recently as June 2013[1].
I would like to make use of this tagger and analyse how different it is from the NLTK tagger, and also see how it might improve accuracy in the classification.

APPENDIX

A. ADDITIONAL WORK

A.1 Github Link

The code for the Sentiment Analysis of Twitter can be found at the following link:
https://github.com/shachi04/TwitterSA

A.2 Numeric features

The numeric features that were used in this experiment include number of negative and positive words, emoticons, length of tweets and number of special characters such as exclamations, hashtags and so on. More details on few of these features is given in the Data section. The numeric features did not yield good accuracy and gave around 63 percent accuracy.

A.3 TF-IDF

Term frequency-Inverse document frequency (tf-idf) is a numeric value that tells how important a word is in a document. It is a very popular feature generation technique used in information retrieval[3].

Term frequency:
In document classification each term is assigned a weight, depending on the number of times it occurs in the document. This is known as the term frequency and is denoted by $tf_{t,d}$. The term weight does not depend on the order of its occurrence but only the frequency.

Inverse Document frequency:
Let the document frequency , defined as the number of docu-

[8]`http://www.ark.cs.cmu.edu/TweetNLP/`

ments in the collection that contain the term t, be df_t. If the total number of documents is N, then the inverse document frequency of the term is given by :

$$idf_t = log\frac{N}{df_t}$$

Thus, the idf of a rare term is high, whereas the idf of a frequent term like 'the' is low.

Tf-idf weighting:
The tf-idf of a word t in a document d is given by :

$$tf - idf_{t,d} = tf_{t,d} \times idf_t$$

Tf-idf gets a high score when the term t appears in less number of documents, but more frequently. Its score is low when the term appears in most documents which would mean that the term cannot be used to distinguish the document.

Tf-idf is very popular in text classification. In this view, it was used as one of the features in this project. But tf-idf gave a low accuracy. This could be attributed to the fact that tweets are of very short length and considering these as documents does not add any value.

B. REFERENCES

[1] Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. *Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments*

[2] Alec Go, Richa Bhayani and Lei Huang, *Twitter Sentiment Classification using Distant Supervision*

[3] Christopher D.Manning, Prabhakar Raghavan and Hinrich Schutze, *Introduction to Information Retrieval*

[4] Bo Pang and Lillian Lee, Opinion mining and sentiment analysis

[5] Tom M. Mitchell, generative and discriminative classifiers: Naive Bayes and Logistic Regression

[6] Christopher M. Bishop, Pattern Recognition and Machine Learning