# Sentiment Analysis of Twitter Statuses

Devansh Manu
Center of Exact Humanities
devansh.manu@research.iiit.ac.in

***Abstract :*** **In this paper we present results of using different tools for sentiment analysis of Twitter statuses. Sentiment analysis is the task of identifying whether the opinion expressed in a text is positive or negative or neutral in general, or about a given topic. We have utilized methods like Naive Bayes Classifier, Maximum Entropy techniques and Linear SVM models to classify the tweets into positive, negative or neutral sets, over a set of features that were extracted from the texts using techniques taken from the field of natural language processing (NLP). We present experimental evaluation of our dataset and classification results derived from different techniques with respective comparisons. Much future work remains, but the results indicate that Linear SVM model is a promising technique for sentiment analysis.**

***Keywords—sentiment analysis, opinion mining, classification, machine learning***

## II. INTRODUCTION

Social networks have revolutionized the way in which people communicate. Information available from social networks is beneficial for analysis of user opinion, for example measuring the feedback on a recently released product, looking at the response to policy change or the enjoyment of an ongoing event. Manually sifting through this data is tedious and potentially expensive. Nowadays, when the micro-blogging platforms, such as Twitter, are commonly used, the task of sentiment analysis becomes even more interesting. Microblogging introduces a new way of communication, where people are forced to use short texts to deliver their messages, hence containing new acronyms, abbreviations, and grammatical mistakes that were generated intentionally.

Sentiment analysis is a relatively new area, which deals with extracting user opinion automatically. An example of a positive sentiment is, “I am so happy today.” alternatively, a negative sentiment is “It has been such a horrible day altogether.”. Objective or neutral texts are deemed not to be expressing any sentiment, such as news headlines, for example “Company shelves wind sector plans”. There are many ways in which social network data can be leveraged to give a better understanding of user opinion such problems are at the heart of natural language processing (NLP) and data mining research.

A variety of techniques for supervised learning algorithms have demonstrated reasonable performance for sentiment analysis; a non-exhaustive list includes Naive Bayes [Lewis, 1998; McCallum and Nigam, 1998; Sahami, 1996], k-nearest neighbor [Yang, 1999], support vector machines [Joachims, 1998; Dumais et al., 1998], boosting [Schapire and Singer, 1999] and rule learning algorithms [Cohen and Singer, 1996; Slattery and Craven, 1998].

Among these, we have selected Naive Bayes Classifier, Maximum Entropy techniques and Linear SVM model to carry out the sentiment analysis for our dataset, and hence compared their respective values as well as their accuracy and performance for twitter feeds sentiment analysis.

## II. RELATED WORK

The increase in social media networks has made sentiment analysis a popular research area, in recent years. In Turney[4] reviews are classified by calculating the summation of polarity of the adjectives and adverbs contained within text. This study utilised movie and car reviews, where thumbs up and thumbs down ratings indicate positive and negative sentiment respectively. A discrepancy between the accuracy of the movie and car reviews was observed with the car reviews getting a higher accuracy. This was attributed to the fact that movie reviews, whilst being positive, can have a lot of adjectives and adverbs that do not fully relate to the overall enjoyment of the film and can actually be more a description of the scenes within the film itself. The PMI-IR (Pointwise Mutual Information - Informations Retrieval ) algorithm was used to classify documents. This algorithm works by taking the relevant bigrams from the document then using the near function on a search engine to see how many times this bigram appears near a word that expresses strong positive or negative sentiment, a large number of matches indicates a stronger polarity.

Pang[1] consider word presence vs frequency where word presence is found to be more effective than word frequency for sentiment analysis. Word position within a given sentence can also be effective, where such information can be used to decide if a particular word has more strength at the beginning or the end of a given sentence.

Go[2] train sentiment classifier on Twitter data. This itself presents a new challenge as there is no explicit rating system such as star rating or thumbs rating like in previous work. This issue is negated through the use of Twitter’s search functionality by searching for emoticons like :) :( representing positive and negative sentiment respectively. This system is highly limited as it is restricted to binary classification and

does not take into account objective texts. This work explored the use of several different classifiers across different n-grams with and without the use of POS tags. A combination of using Unigrams and Bigrams give the best results across all classifiers. The inclusion of POS tags with unigrams had a negative impact across all classifiers however this still performed better than using bigrams.

## III. DATA COLLECTION AND PREPROCESSING

Twitter API was used for the data extraction process. Negative, positive and objective texts were collected from the following dataset: https://www.cs.york.ac.uk/semeval-%C2%AD%E2%80%902013/task2/index.php?id=data. The steps followed included the removal of any urls and usernames (usernames follow the @symbol replacing them with a generic term AT_USER) and removal any characters that repeat more than twice turning a phrase such as OOMMMGGG to OOMMGG, which is applied by a regular expression. Finally, the stopset words were removed from the data. The stopset is the set of words such as “a”, “and”, “an”, “the”, these are words that do not have any meaning on their own. The second phase is associated with the stemming for each word. The nltk library was used for stemming and the extraction of unigrams .

| Before Preprocessing | After Preprocessing |
|---|---|
| @Peter I heard about the contest! Congrats girl !!! | AT_USER heard about contest! Congrats girl!!! |

## IV. CLASSIFIERS

In our first set of experiments, we are trying to see how good each classifier performs on the datasets, with different features settings. All measurements are done on the average accuracy of the same 10-fold cross validation split. The following are the types of classifiers, thus used and evaluated.

**Naive-Bayes Classifier :** We decided to use a classification strategy based on Naive Bayes (NB) because it is a simple and intuitive method whose performance is similar to other approaches. NB combines efficiency (optimal time performance) with reasonable accuracy. The main theoretical drawback of NB methods is that it assumes conditional independence among the linguistic features. If the main features are the tokens extracted from texts, it is evident that they cannot be considered as independent, since words co-occuring in a text are somehow linked by different types of syntactic and semantic dependencies. However, even if NB produces an oversimplified model, its classification decisions are surprinsingly accurate (Manning et al., 2008).

A Naive bayes classifier is a simple probabilistic model based on the Bayes rule along with a strong independence assumption. The Naïve Bayes model involves a simplifying conditional independence assumption. That is given a class (positive or negative), the words are conditionally independent of each other. This assumption does not affect the accuracy in text classification by much but makes really fast classification algorithms applicable for the problem.

$$P(C|m) = P(C)\prod_{i=1}^{n} P(f_i|C)$$

where C is the class positive, negative or objective sets, $m$ is the twitter message and $f$ is a feature. In our experiments the features are POS tags, unigrams or bigrams.

**Maximum Entropy Technique :** The Maximum Entropy (MaxEnt) classifier is closely related to a Naive Bayes classifier, except that, rather than allowing each feature to have its say independently, the model uses search-based optimization to find weights for the features that maximize the likelihood of the training data.

The features you define for a Naive Bayes classifier are easily ported to a MaxEnt setting, but the MaxEnt model can also handle mixtures of boolean, integer, and real-valued features.

We have briefly sketched a general recipe for building a MaxEnt classifier, assuming that the only features are word-level features:

1. For each word $w$ and class $c \in C$, define a joint feature $f(w, c) = N$ where $N$ is the number of times that $w$ occurs in a document in class $c$. ($N$ could also be boolean, registering presence vs. absence.)

2. Via iterative optimization, assign a weight to each joint feature so as to maximize the log-likelihood of the training data.

3. The probability of class $c$ given a document $d$ and weights $\lambda$ is

$$P(c|d,\lambda) \stackrel{def}{=} \frac{\exp\sum_i \lambda_i f_i(c,d)}{\sum_{c' \in C} \exp\sum_i \lambda_i f_i(c',d)}$$

Because of the search procedures involved in step 2, MaxEnt models are more difficult to implement than Naive Bayes model, but this needn't be an obstacle to using them, since there are excellent software packages available.

The features for a MaxEnt model can be correlated. The model will do a good job of distributing the weight between correlated features. (This is not to say, though, that you should be indifferent to correlated features. They can make the model hard to interpret and reduce its portability.)

**Suppport Vector Machines :** A support vector machine (SVM) is an algorithm that uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane. A hyperplane is a “decision boundary” separating the tuples of one class from another. With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane.

$$\langle w, \nu(x)\rangle = \sum_{i=1}^{n} z_i y_i \langle \nu(x_i), \nu(x)\rangle \qquad = \sum_{i=1}^{n} z_i y_i \kappa(x_i, x)$$

The SVM finds this hyperplane using support vectors (“essential” training tuples) and margins (defined by the support vectors). The basic idea behind support vector machine is illustrated with the example shown in Figure 1. In this example the data is assumed to be linearly separable. Therefore, there exist a linear hyperplane (or decision boundary) that separates the points into two different classes. In the two-dimensional case, the hyperplane is simply a straight line. In principle, there are infinitely many hyperplanes that can separate the training data.
Figure 1 shows two such hyperplanes, B1 and B2. Both hyperplanes can divide the training examples into their respective classes without committing any misclassification errors. Although the training time of even the fastest SVMs can be extremely slow, they are highly accurate, owing to their ability to model complex nonlinear decision boundaries. They are much less prone to over fitting than other methods.

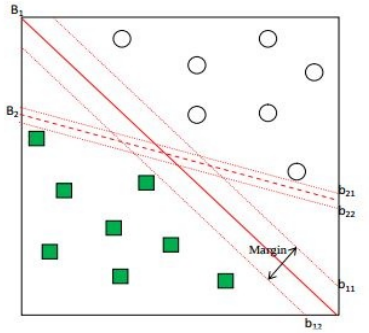



**Figure 1.** An example of a two-class problem with two separating hyperplanes, $B_1$ and $B_2$ [1]

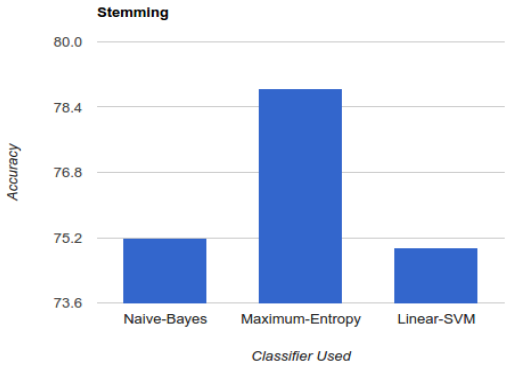
Two feature settings were used :

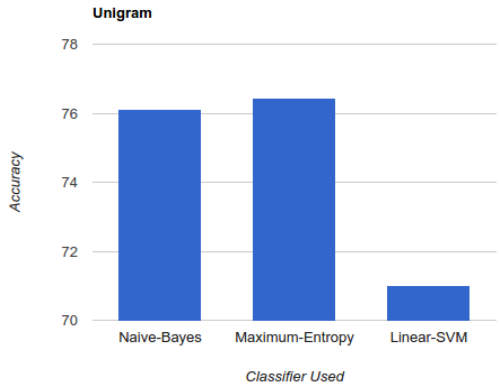
1. Unigrams - Using the word itself.
2. Stemming - Getting rid of prefixes and suffixes.

To improve the working of the classifiers.

## V. EXPERIMENTAL ANALYSIS

Having the three classifiers at hand, we now need to measure how well they perform. Upon 10-fold cross validation, we found that Maximum Entropy classifier was the best among the three then was the Naive-Bayes classifier and surprisingly the linear SVM was the last. Also stemming was found to be a better setting than unigrams as it gave a higher accuracy The difference in the accuracies were around 4% - 5%.




## VI. FUTURE WORK

We believe that more work has to be done in order to learn other reliable features ( synonyms, negations, named entities etc.), so that one can use them to improve accuracy when running on Twitter statuses. Also the comparison should be carried out with different type of SVM kernels (polynomial, gaussian etc.) and their results should be evaluated.

## VII. REFERENCES


- Bo Pang, L.L.: Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval January Volume 2 Issue 1-2, 1–94 (2008), http://www.cs. 65 Sentimentor: Sentiment Classification of Twitter Data 11 cornell.edu/home/llee/omsa/omsa.pdf

- Go, A., Bhayani, R., Huang, L.: Twitter sentiment classification using distant supervision. Processing 150(12), 1–6 (2009), http://www.stanford.edu/~alecmgo/papers/TwitterDistantSupervision09.pdf
- Pak, A., Paroubek, P.: Twitter as a corpus for sentiment analysis and opinion mining. In: Chair), N.C.C., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M., Tapias, D. (eds.) Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10). European Language Resources Association (ELRA), Valletta, Malta (may 2010)
- Tracking "Gross Community Happiness" from Tweets *by Daniele Quercia, Jonathan Ellis, Licia Capra and Jon Crowcroft.* (http://www.cs.ucl.ac.uk/fileadmin/UCL-CS/research/Research_Notes/RN_11_20.pdf)
- Russell Stuart J. and Peter Norvig. 2003. "Artificial Intelligence: A Modern Approach (2 ed.). Pearson Education". *p. 499 for reference to "idiot Bayes" as well as the general definition of the Naive Bayes model and its independence assumptions*
- *Jason Rennie, Lawrence Shih, Jaime Teevan, David Karger , Tackling the Poor Assumptions of Naïve Bayes Text Classifiers, presentation slides(http://cseweb.ucsd.edu/~elkan/254/NaiveBayesForText.pdf)*