Efficient Sentiment Classification of Twitter Feeds

Nicholas Chamansingh and Patrick Hosein
Department of Computer Science
The University of the West Indies, St. Augustine, Trinidad
e-mail: nicholas.chamansingh@gmail.com, patrick.hosein@sta.uwi.edu

***Abstract*—Sentiment Analysis encompasses the use of Natural Language Processing together with statistics and machine learning methods for the identification, extraction and characterization of sentiment elements from a body of text. Micro-blog platforms, such as Twitter, allows for the sharing of real-time comments and opinions from millions of users on various topics. This research presents an experiment to determine an efficient sentiment classifier of real-time Twitter feeds. Naive Bayes, Support Vector Machine (SVM) and Maximum Entropy (MaxEnt) classification methods were compared. For each approach we used the same pre-processing and feature selection methods. Chi-Square feature selection was used to determine the smallest feature set and training data size needed for a classifier with a given accuracy level, storage requirements and classification time. Results show that, when compared to previous work, a significant reduction in data input and processing can be achieve while maintaining an acceptable level of accuracy.**

Keywords-sentiment analysis; sentiment classification; twitter; data analytics

I. Introduction

Opinions provide the views or feeling about specific topics and events. Social media has presented a range of possibilities for the interaction of people who may never meet or interact with each other in person. Platforms which allow real-time sharing of comments and opinions are called micro-blogs. One of the most popular micro blogging platforms is Twitter which has millions of users who share millions of opinions and feelings about daily topics and events [1].

Sentiment generally falls into the categories of feelings, attitudes, emotions and opinions and is subjective, that is, not based on facts. Binary oppositions can also be thought of as Sentiment such as positive/negative, like/dislike and good/bad. Sentiment Analysis can be used for various purposes such as determining if product reviews are positive or negative, if bloggers attitudes have changed since an election or even identifying (in)appropriate content for an advertisement placement. In this paper we describe experiments that were performed to obtain an efficient sentiment classification of Twitter feeds on events based on a two-class classifier, positive/negative. As events happen, social media users may want to know how the population feels about that event, when it happened and how opinions on the topic progressed over time.

Sentiment Analysis is typically performed by first using some data for feature extraction. These features are used together with a classification algorithm to analyze the stream of data to determine the sentiment of a particular topic. The first step is compute intensive and involves the preprocessing and classification of a tweet while the second step may take some time to obtain a definitive decision on the topic sentiment. Our objective is to develop a system that will dynamically adapt over time (i.e., continuously perform the preprocessing function and classification) and be able to provide sentiment information in near real-time. We therefore seek to determine the minimum amount of processing and training data required to obtain results at a certain level of confidence. This will determine the level of latency in a near-real time system. Such a real-time system is in development and will be provided in a subsequent paper.

The rest of this paper is organized as follows. Section 2 presents related work followed by a description of the sample dataset in Section 3. Section 4 describes the methodology used and the preprocessing of the data. Section 5 explains the efficiency experiments which were performed and Section 6 presents the results of the experiments. Section 7 presents the discussion. The conclusions of the research are provided in Section 8.

II. Related Work

In [2], models were built for two classification tasks, a binary task which only classifies positive and negative sentiment and a 3-way task which classifies sentiment into positive, negative and neutral. Experiments were conducted with a unigram model (i.e. each term assumed to be independent of others), a feature based model and tree kernel model. A unigram model was used as a baseline to compare the results of the experiments conducted with results showing that, for both classification tasks, the unigram model achieved a performance gain of over 20% . The feature based model only used 100 features and achieved an accuracy similar to the unigram model which uses over 10,000 features. Both models were outperformed by the tree kernel based model. Experiments were also conducted by combining unigrams with the 100 features as well as a combination of the 100 features with the tree kernel. In both experiments the baseline unigram was outperformed by over 4%. From the experiments conducted it was found that Twitter specific features (emoticons, hashtags, etc.) added little value to the classifier. Additionally, prior polarity of words with parts-of-speech tags features was beneficial for both classification tasks.

Similarly, Bakliwal et. al. [3] conducted sentiment analysis experiments with a three-class positive/negative/neutral classification on the topic of the Irish General Elections of February 2012. The dataset used in the experiments were doubly annotated as positive, negative and neutral. Sentiment baselines were established for tweets that were marked sarcastic. They explored two approaches. The first utilized subjectivity lexicon as the primary source of information to determine the sentiment (positive, negative or neutral), of a

978-1-5090-3469-7/16/$31.00 ©2016 IEEE

political party or party leader. The overall best version of that method had an accuracy of 58.9% which outperformed the majority baseline by 4%. The second method combined bag-of-words features, some Twitter specific features and lexicon scores and these additions increased the accuracy to 61.6%.

Go et. al. [4] proposed a method which automatically extracts positive or negative sentiment from tweets and can aggregate feedback without manual intervention. The training data consisted of tweets with emoticons and utilized distant supervision. Emoticons, such as ☺ for positive and ☹ for negative, were used as noisy labels. Emoticon data were used to train classifiers which were tested against a set of tweets. A list consisting of 174 positive words and 185 negative words were used as a baseline, and the number of negative and positive keywords that appeared in each tweet were counted. The polarity with the higher count was returned by the classifier and a positive polarity was returned in the case of a tie. The machine learning methods used were Naive Bayes, Maximum Entropy (MaxEnt) and Support Vector Machine (SVM). The keyword baseline was outperformed by the machine learning algorithm with accuracy results of 81.0% for Naive Bayes, 80.4% for MaxEnt and 82.9% for SVM. A combination of unigrams and bigrams were also used as features with improvements for Naive Bayes (from 81.3% to 82.7%) and MaxEnt (from 80.5% to 82.7%) and a decline for SVM (from 82.2% to 81.6%). They also noted that Parts of Speech (POS) tags did not improve the accuracy of the results.

Additionally, Go et. al. [5] conducted experiments to accurately classify Twitter messages. Tweets which consisted of emoticons, such as ☺ and ☹, were collected as training data. In total, 75 negative tweets and 108 positive tweets were manually annotated. The experiments used different classifiers including Naive Bayes, Maximum Entropy and Support Vector Machine. From their results the Naive Bayes classifier together with Mutual Information feature selection provided the best accuracy of 84.15% and Naive Bayes together with Chi-square feature selection had an accuracy of 80.32%. The Maximum Entropy classifier performed similarly to the Naive Bayes classifier but it took very long to train and test. The Support Vector Machine had an accuracy of 79.91%. Different feature extraction methods were used together with each classifier. These included Unigrams, Bigrams, Negates as features and Parts of Speech tagging. It was found that the unigram feature extraction model provided the best results because of the nature of tweeted messages.

From the previous work as well as work included in [3], [6], [7] and [8], one can conclude that the Naive Bayes classification model was the least computationally expensive and provided a reasonable accuracy when compared to Maximum Entropy and Support Vector Machine. Furthermore, the Chi-Square approach selects appropriate features which improves the accuracy of the classifier. From these results, experiments was conducted using the Nave Bayes, MaxEnt and SVM classifiers with Chi-square feature selection to compare and derive an efficient Sentiment classifier.

III. Data Description

Tweets are the messages posted on the social networking and micro-blogging service called Twitter. The service allows users to post their thoughts, feelings and generally anything in real time. Restricted to 140 characters in length, tweets are short messages. Because of the length of the tweets, acronyms as well as emoticons and other characters are used to express feelings. In addition to text, tweets may also include the following:

- Emoticons: These are facial expressions represented by punctuation and letters, for example ":)" for a smile and ":(" for a frown. Emoticons are used to express a user's mood.
- The "@" symbol: Users refer to other users on Twitter using this symbol (e.g. @John).
- The hashtag symbol"#": These are used to mark topics. This is done to increase the visibility of tweets (e.g, #worldcup).

The data for this research, which was used for training and testing, was taken from [9]. The dataset was streamed from the Twitter Search API by using keywords such as "google" and "visa". The dataset contained 1.6 million tweets and was automatically annotated. The entire process is described in [4]. The process parsed tweets and those that contained the emoticon ":)" were classed as positive while those that contained the emoticon ":(" were classed as negative. The following were included for each tweet: the polarity of the tweet (0 = negative, 4 = positive); the id of the tweet (eg. 2087); the date of the tweet (eg. Sat May 16 23:58:44 UTC 2009); the query (eg. lyx); the user that tweeted (eg. robotickilldozr); and the text of the tweet (eg. Lyx is cool).

IV. Methodology and Tools

For the classification of tweets, a supervised learning approach was taken. The experiments were conducted comparing Naive Bayes, Support Vector Machine and Maximum Entropy algorithms with unigrams as the model. Features were selected using the Chi-square test.

A. Naive Bayes

For this work, Multinomial Naive Bayes was used. Multinomial Naive Bayes works on the assumption that, given the class, all features are conditionally independent and hence the following applies:

$$P(c|t) = P(c)P(t|c)/p(t) \tag{1}$$

where t is the text to classify and c is a particular class. $P(c)$ and $P(t)$ represent the probabilities of the class and text independently. $P(t|c)$ represents the probability of text t appearing given class c.

The aim of the algorithm is to choose a value of c which maximizes $P(c|t)$. For text t, w_i represents the i-th feature in it. Therefore $P(w_i|c)$ represents the probability of that feature appearing if class c is present. Parameters $P(c)$ and $P(w_i|c)$ need to be trained. Using the Naive Bayes method, these parameters are obtained by using their maximum likelihood estimation (MLE). In order to make a prediction on a new sentence t, the log likelihood of different classes are calculated and the class with the highest log likelihood is taken as the prediction. The log likelihood is calculated using the following formula:

$$\log P(c) + \sum_i i \log P(w_i|c) \tag{2}$$

B. Maximum Entropy

Maximum Entropy (MaxEnt) models are feature-based models where the idea is to choose the most unigram models, from all models which suit the training data, which conform to a specic constraint. Unlike Naive Bayes, Maximum Entropy

does not make any inference that the features are conditionally independent of each other [10].

In the case of our experiments, the aim of MaxEnt was to use unigrams, which are contextual information from the twitter feeds, in order to categorize the feeds into a given class (positive or negative). A stochastic model which correctly represents the behavior of the random process, where a tweet x would be taken as input and a value y would be produced as the output, first needs to be created. The model is built using statistics from the training data. Consider the feature function shown below:

$$f_j(x,y) = \begin{cases} 4 & \text{if } y = \text{positive and x contains } word_k \\ 0 & \text{if } y = \text{ negative and x contains } word_k \end{cases} \tag{3}$$

Hence, the following equation represents the MaxEnt Models:

$$p^*(y|x) = \frac{\exp\left(\sum_i \lambda_i f_i(x,y)\right)}{\sum_i \exp\left(\lambda_i f_i(x,y)\right)} \tag{4}$$

In classification, the importance of a feature is decided upon by weight vectors. The higher the weight, the stronger the feature is as an indicator for the class. This weight vector is calculated by optimizing over Lagrange Multipliers numerically in order to maximize the conditional probability.

C. Support Vector Machines

Another common classification method is Support Vector Machine (SVM) [11]. SVM is a type of supervised technique, where in sentiment analysis, involves training a sentiment classifier via the frequency of various words appearing in a tweet. The input data can be in a numerical form, such as a word index number and a weight. This numerical data will create values and patterns which can be used in the testing phase to label processed tweets. The optimal hyper-plane of the SVM can be calculated using the following formula:

$$f : w \cdot x + b = 0 \tag{5}$$

where w represents the hyper-plane parameters; x represents the SVM data; b represents the bias hyper-plane parameters; and f represents the function of the hyper-plane.

D. Chi-Square Feature Selection

For the Chi-square feature selection, a score is used to measure if a feature and a class are independent of each other. This score is calculated using the Chi-square test which is a statistical method to determine if two events are independent of each other. The method first assumes that the feature and class are independent and then calculates a Chi-square score. The larger the value of the score, the more dependent they are on each other. Therefore, for each class, the aim is to select the features with the highest Chi-square scores. The Chi-square score is calculated using the following formula:

$$\chi^2(F,C) = \frac{N(N_{11}N_{00} - N_{10}N_{01})^2}{(N_{11}+N_{01})(N_{11}+N_{10})(N_{10}+N_{00})(N_{01}+N_{00})} \tag{6}$$

where N represents the total number of training sentences, N_{11} is the number of times feature F occurred in class C, N_{10} represents the number of sentences which are in class C and contain feature F, N_{01} represents the number of sentences which do not contain feature F but are in class C and N_{00} is the number of sentences which do not contain feature F and are not in class C.

E. Tools

In order to carry out the classifications of the tweets we used the Natural Language Toolkit (NLTK) [12] and Scikit-Learn [13] both of which are python libraries. Both NLTK and Scikit-Learn are among the leading platforms for implementing Python programs for classification and language processing. Scikit-Lean provides many classification algorithms, some of which were used in the experiments in this paper. NLTK provides a user friendly interface which provides access to more than 50 corpora and lexical resources (e.g. WordNet). NLTK also provides a collection of text processing libraries which provide services such as tokenization, classification, semantic reasoning, tagging , parsing, stemming and an active discussion forum.

F. Preprocessing

The dataset mentioned above (see Section III) underwent preprocessing before classification and testing. The purpose of preprocessing the data was to ensure that all unnecessary attributes of the tweets were removed and normalized.

For the preprocessing stage the data was first split into two sets, one containing all positive tweets (800000 positive tweets) and another containing all negative tweets (800000 negative tweets). For each set, each tweet was decoded to "utf-8" format then converted to lowercase. This was followed by removing urls and embedded links and replacing them with empty strings. The next step involved removing the "@username" tag which was used when tagging a user and was replaced with "AT_USER". All additional white spaces were removed. The final step normalized words with two or more repetitions of a character and replaced it with the character itself. Once each tweet in the set was preprocessed, it is added to a new set which only contained the processed tweet of the set and its associated sentiment.

After the preprocessing of the tweets, the Chi-square feature selection method was used to generate a word score for each word in the tweet. The total number of unique features (words) found in the dataset was 244946 features, each with an associated word score ranging from most informative to least informative.

V. Experiments

In order to efficiently classify tweets in real time and to test the accuracy of the classifier, each classifier underwent two sets of testing, accuracy testing and classification time testing. We also compared the file sizes of the classifiers and the respective feature set of each of the top classifiers. This section starts by describing the accuracy testing which was performed to determine the best trade-off between accuracy and the number of features of the classifier. It then explains the classification time testing. Classification time testing was performed to determine the trade-off between the training set and the time taken to classify incoming tweets.

A. Accuracy Tests

Data was selected from a set of 1.6 million tweets (see Section III). A different subset of tweets was used for each test and for all tests the data was split into negative and positive sets for extraction and processing. For each test conducted, the training was done with three-quarters (1200000 tweets) of the dataset while testing was performed with one-quarter (400000 tweets).

TABLE I. DATA SETS USED FOR TESTING

Training Size	Test Size	Feature Size	Percentage
1200000	400000	24494	10%
1200000	400000	48989	20%
1200000	400000	73483	30%
1200000	400000	97978	40%
1200000	400000	122369	50%
1200000	400000	146842	60%
1200000	400000	171316	70%
1200000	400000	195790	80%
1200000	400000	220264	90%
1200000	400000	244946	100%

Each classifier was created using percentages of the total number of features and then tested for accuracy. For the first test, 10% of the features were used to created the classifier. For every subsequent test, the percentage was increased by 10% until the classifier was created with the full 100% of the features (244946 features). Table I shows the number of features used for each test conducted on the different classifiers. These tests were conducted in order to determine the most efficient classifier. A classifier was considered to be efficient based on its accuracy, file size and time taken to classify tweets.

B. Classification Runtime

The top classifiers obtained for each algorithm were tested against each other using the test data (400000 tweets) to determine the time taken to classify tweets. Since we are emulating the length of time it takes to classify tweets in a near real-time system, classification time testing involved determining the time taken to preprocess the 400000 tweets and classify them. Once the preprocessing and classification steps were completed, the total time taken was divided by the number of tweets in order to obtain the average time taken to classify one tweet.

C. File Size Comparison

The file sizes of each classifier which was needed to obtain a reasonable level of accuracy within a short period of time, were compared to determine the amount of memory required to load the classifiers for classification in a near real time situation. These files include the classifier file and the feature set file.

VI. RESULTS

This section provides a detailed account of the results from the testing phase. It first presents the results from the classifier accuracy testing for each algorithm. It then compares the time taken to classify tweets. Finally it goes on to compare the file sizes of each classifier and their associated feature sizes.

A. Accuracy Results

Figure 1 illustrates the results from the accuracy classification testing. The plot displays a graph with the results from testing all classifiers with the different number of features and their respective accuracy. From the graph it can be seen that when the number features was approximately 45000 (20% of the total features), the accuracy was the highest, 77.02% to 78.22%. When the number of features increased from 45000, the accuracy decreased and remained constant, 76.95% to 78.50%. This trend continued until the number of features were approximately 20000 (90% of the total features) where the accuracy dropped to 76.44% to 77.04%. The accuracy was the lowest for each algorithm when the total number of features were used. This accuracy ranged between 75.74% to 77.39%.

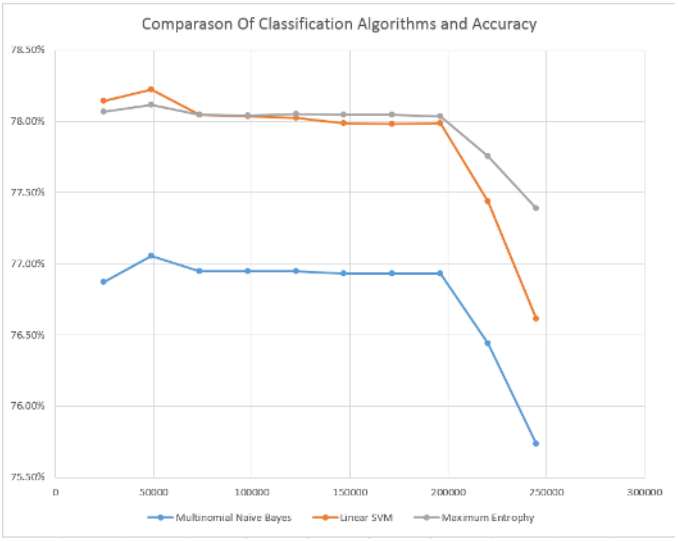

Figure 1. Classification Algorithms Comparison

TABLE II. TIME TAKEN TO CLASSIFY 400000 TWEETS

Classifier	Runtime (s)	Time/Tweet (ms)
Multinomial Naive Bayes	75	187
Linear SVM	259	647
Maximum Entropy	141	352

B. Classification Runtime Comparison

The time taken to classify 400000 tweets using the top performing classifiers were recorded and shown in Table II. Each algorithm had the highest accuracy when 20% of the top performing features were used (48989). These classifiers were tested with 400000 tweets. The Multinomial Naive Bayes classifier took the shortest time to classify the total test set, requiring 187ms to process and classify a tweet. The second fastest classifier was the Maximum Entropy classifier which took 352ms to process and classify a tweet. The Support Vector Machine classifier took the longest with a time of 647ms to process and classify a tweet.

C. File Size Comparison

For each of the top classifiers we recorded classifier file sizes and feature set file sizes. These are presented in Table III. Each algorithm was tested with 400000 tweets and used 20% (48989) of the features. From the results we find that each classification algorithm had the same file size for the feature set. Support Vector Machine and the Maximum Entropy tied for having the smallest classifier file size. The largest classifier file belonged to the Multinomial Naive Bayes classifier.

Table III. COMPARISON CLASSIFIER FILE SIZES

Classifier Size	Test Size (MB)	Feature Size (KB)
Multinomial Naive Bayes	14.1	819.9
Linear SVMs	2.6	819.9
Maximum Entropy	2.6	819.9

VII. DISCUSSION

From the results each classifier performed the best in terms of accuracy when 20% (48989) of the total number of features were used. The highest accuracy of 78.14% belonged to the SVM classifier, followed by 78.07% obtained from the Maximum Entropy classifier and finally 77.06% which belonged to the Naive Bayes classifier.

The Naive Bayes algorithm had a classifier file size of 14.1 MB whilst both SVM and Maximum Entropy had a file size of 2.6 MB. The feature size for all of classifiers was 819.9 KB. The final comparison among each of the top classifiers from the classification algorithms was the time taken to classify tweets. Each classifier classified 400000 tweets and the total time was divided by the number of tweets in order to estimate the average time taken to classify one tweet. This resulted in classification times of 187.525ms, 646.575ms and 352.425ms for Nave Bayes, SVM and MaxEnt respectively. From the results of the experiments, the MaxEnt classifier has been shown to be the most efficient in terms of accuracy, time taken to classify tweets and file size.

VIII. CONCLUSIONS AND FUTURE WORK

This research focused on sentiment analysis of Twitter feeds, more specifically to determine an efficient sentiment classifier of real-time Twitter feeds. The experiment included the comparison of three classification algorithms namely Multinomial Naive Bayes, Linear Support Vector Machine and Maximum Entropy. In order to reduce the number of features for classification, the Chi-Square feature selection method was used to assess the performance of the features used. Testing was performed on each of the classification algorithms which used different numbers of features to derive an efficient classifier.

The work presented by [3] involved multiple steps in the preprocessing stage, such as lexicon scoring, Part-of-Speech tagging and negation. Although the combination of these methods have shown to have an accuracy of 61.62%, the computational time and the time taken for the classification phase would be greatly increased by the increased number of methods used. In addition, [2] preformed preprocessing with both an emoticon dictionary and an acronym dictionary. Their experiments also involved obtaining prior polarity scores. The combination of all of these steps would require additional computational time, although their experiments resulted in an accuracy of 75.39%. The work presented in this paper involved only the Chi-squared feature selection method and provided an accuracy of 78.07%. Therefore, the work has proven that acceptable accuracy can be achieved with fewer steps and hence would result in a lower computational time. Hence, our experiments achieved the objectives of providing a classifier which can be used in a system that can dynamically adapt over time since less computation time is required.

For future work, domain specific tweets will be used for training with a more granular annotation method than the one used to obtain training data for this paper. Using tweets which fall under a particular domain may have the possibility of yielding better performance. We also plan to deploy a real-time classifier that dynamically re-trains using recent data.

REFERENCES

[1] F. Bravo-Marquez, M. Mendoza, and B. Poblete, "Combining strengths, emotions and polarities for boosting twitter sentiment analysis," in *Proceedings of the Second International Workshop on Issues of Sentiment Discovery and Opinion Mining*. ACM, 2013, p. 2.

[2] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau, "Sentiment analysis of twitter data," in *Proceedings of the Workshop on Languages in Social Media*. Association for Computational Linguistics, 2011, pp. 30–38.

[3] A. Bakliwal, J. Foster, J. van der Puil, R. O'Brien, L. Tounsi, and M. Hughes, "Sentiment analysis of political tweets: Towards an accurate classifier." Association for Computational Linguistics, 2013.

[4] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N Project Report, Stanford*, vol. 1, p. 12, 2009.

[5] A. Go, L. Huang, and R. Bhayani, "Twitter sentiment analysis."

[6] C. Lin and Y. He, "Joint sentiment/topic model for sentiment analysis," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 375–384.

[7] E. Kouloumpis, T. Wilson, and J. D. Moore, "Twitter sentiment analysis: The good the bad and the omg!" *Icwsm*, vol. 11, pp. 538–541, 2011.

[8] X. Zhou, X. Tao, J. Yong, and Z. Yang, "Sentiment analysis on tweets for social events," in *Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on*, June 2013, pp. 557–562.

[9] Sentiment140. (2014) Sentiment140 for academics. [Online]. Available: http://help.sentiment140.com/for-students

[10] V. Vryniotis. (2013) Machine learning tutorial: The max entropy text classifier. [Online]. Available: http://blog.datumbox.com/machine-learning-tutorial-the-max-entropy-text-classifier/

[11] Tiara, M. K. Sabariah, and V. Effendy, "Sentiment analysis on twitter using the combination of lexicon-based and support vector machine for assessing the performance of a television program," in *Information and Communication Technology (ICoICT), 2015 3rd International Conference on*, May 2015, pp. 386–390.

[12] NLTK.org. (2015) Nltk 3.0 documentation: Natural language toolkit. [Online]. Available: http://www.nltk.org/

[13] scikit learn.org. (2014) scikit-learn 0.17: Documentation of scikit-learn 0.17. [Online]. Available: http://scikit-learn.org/