Sentiment Analysis on Twitter Data using KNN and SVM

Mohammad Rezwanul Huq
Dept. of Computer Science and Engineering
East West University
Dhaka, Bangladesh

Ahmad Ali
Dept. of Computer Science and Engineering
East West University
Dhaka, Bangladesh

Anika Rahman
Dept. of Computer Science and Engineering
East West University
Dhaka, Bangladesh

***Abstract*—Millions of users share opinions on various topics using micro-blogging every day. Twitter is a very popular micro-blogging site where users are allowed a limit of 140 characters; this kind of restriction makes the users be concise as well as expressive at the same time. For that reason, it becomes a rich source for sentiment analysis and belief mining. The aim of this paper is to develop such a functional classifier which can correctly and automatically classify the sentiment of an unknown tweet. In our work, we propose techniques to classify the sentiment label accurately. We introduce two methods: one of the methods is known as sentiment classification algorithm (SCA) based on k-nearest neighbor (KNN) and the other one is based on support vector machine (SVM). We also evaluate their performance based on real tweets.**

Keywords—Support Vector Machine (SVM); k-nearest neighbor (KNN); Grid Search; Confusion matrix; ROC graph; Hyperplane; Social data analysis

I. Introduction

These days social networks, blogs, and other media produce a huge amount of data on the World Wide Web. This huge amount of data contains crucial opinion related information that can be used to benefit for businesses and other aspects of commercial and scientific industries. Manual tracking and extracting this useful information from this massive amount of data is almost impossible. Sentiment analysis of user posts is required to help taking business decisions. It is a process which extracts sentiments or opinions from reviews which are given by users over a particular subject, area or product in online. We can categorize the sentiment into two types: 1) positive or 2) negative that determine the general attitude of the people to a particular topic. Our principal goal is to correctly detect sentiment of tweets as more as possible. This paper has two main parts: the first one is to classify sentiment of tweets by using some feature and in the second one we use machine learning algorithm SVM [1]. In both the cases, we use five-fold cross validation method to determine the accuracy. We propose two approaches for sentiment analysis. One of the technique facilitates KNN and the other uses SVM. Both techniques work with same dataset and same features. For both SCA and SVM we calculate weights based on different features. Then in SCA, we build a pair of tweets by using different features. From that pair, we measure the Euclidian distance for every tweet with its counterpart. From those distance we only consider nearest eight tweets label to classify that tweet. On the other hand in SVM, build a matrix from the calculated weights based on different features and by applying PCA (principal component analysis) [2], we try to find k eigenvector with the largest Eigen values. From this transformed sample dataset we try to find the best c and best gamma by using grid search technique [3] to use in SVM. Finally, we apply SVM to assign the sentiment label of each tweet in the test dataset. In both algorithms, we use confusion matrix [4] to calculate the accuracy.

Later, we compare our two techniques in respect to an accuracy level of detecting the sentiment accurately. We found that Sentiment Classifier Algorithm (SCA) performs better than SVM.

II. Related Work

Researchers have paid attention to this problem to some extent. In this paper [5], authors looking at popular micro-blogging Twitter, here the authors build models for two classifying tasks. These are a binary task of classifying sentiment into positive, negative classes and three-way task means to classify sentiment into positive, negative and neutral classes. They also performed an experiment with unigram model, a feature based model, and a tree kernel-based model. They were combining unigrams with their features and features with the tree kernel. In this paper, they presented extensive feature analysis of the 100 features they propose.

In this work [6], authors proposed an approach to automatically detect sentiment on Twitter messages (Tweets) and also proposed two-step sentiment analysis classification method for Twitter. First, they classified messages as a subjective and objective category and further distinguishes the subjective tweets as positive or negative. For creating these classifiers, instead of using manually annotated data to compose the training data as regularly supervised approaches, they leverage sources of noisy labels as their training data. These noisy labels were provided by a few sentiment detection websites over Twitter data. For better utilizing these sources, it is important to verify the potential value of using and combining them. A more robust feature set that captures a more abstract representation of tweets and it is composed by meta-information associated to words and specific characteristics of how tweets are written is also proposed by the authors. In Meta-features, they map a given word in a tweet to its part-of-speech using a part-of-speech dictionary as POS tags are good indicators for sentiment tagging. An effective and robust sentiment detection approach for Twitter messages is presented by this paper which uses biased and noisy labels as input to build its models. The main limitations of our approach are the cases of sentences that contain antagonistic sentiments.

By investigating the utility of linguistic features for detecting the sentiment of Twitter messages, the author of this paper [7] evaluate the usefulness of existing lexical resources as well as features to capture information about the informal and creative language used in micro-blogging. For building training data they take a supervised approach but leverage existing hashtags in Twitter data. In their experiment, they use three different corpora of Twitter messages. For development and training, they use hashtagged dataset (HASH) and the emoticon dataset (EMOT). There are three steps for preprocessing the dataset. They are: 1) tokenization, 2) normalization and 3) parts-of-speech (POS) tagging and they also use a variety of features for their classification and experiment. They use unigrams and bigrams for the baseline and they also include features typically used in sentiment analysis such as sentiment lexicon and POS features. The features are n-gram, lexicon features, part-of-speech features, micro-blogging features.

Their experiments [8] show that part-of-speech features may not be useful for sentiment analysis in the micro-blogging domain. Features from an existing sentiment lexicon were somewhat useful in conjunction with micro-blogging features. For automatically classifying the sentiment of Twitter messages, in this paper, the authors introduce a novel approach and they classified these messages as either positive or negative with respect to a query term. For this reason, they build a framework that treats classifiers and feature extractors as two distinct component. This framework allows them to easily try out different combinations of classifiers and features extractors and then normalizing the effect of query terms along with corresponding tweets. Their assumption is that users prefer to perform sentiment analysis on a product and not of a product. By Stripping out the emoticons causes the classifier to learn from other features such as unigrams and bigrams present in the tweet. An interesting side effect of their feature is that they use non-emoticon to determine the sentiment. They consider emoticons as noisy labels because they are not perfect while defining the correct sentiment of a tweet. For example, @BATMANNN :(I love chutney....... Without the emoticon, most people would probably consider this tweet to be positive. Tweets with these types of mismatched emoticons are used to train our classifiers because they are difficult to filter out from our training data. The Twitter language model has many unique properties such as using rnemes and links and taking advantage of the following properties to reduce the feature space. They also use support vector machine classification technique. Here their output data are two sets of vectors of size m. Each entry in the vector corresponds to the presence of feature. In this paper, they show that machine learning algorithms (Naive Bayes, maximum entropy classification, and support vector machines) can achieve high accuracy for classifying sentiment when using this method.

III. Working Procedures

We use a variety of features for our classification experiments. For the baseline, we use word feature, n-gram feature, pattern feature and punctuation feature [9]. Finally, we also include another key based feature. After calculating weight based on features for each tweet, we use our techniques.

In **Word Feature,** each word in tweet considered as a binary feature [10]. For counting the word feature of a tweet, it compares with a dictionary which is used to detect which words are stop words and which are not. Stop words [11], [12] are out of consideration. Otherwise, every word is considered for word feature. Moreover, if we encounter sequences of two or more punctuation symbols inside a tweet, we consider them as word features [13]. Additionally, the common word RT, which means “retweet”, does not constitute a feature because it may appear in the majority of tweets inside the dataset. When we calculate word feature weight we calculate is as $ws = \frac{Nf}{coun(f)}$ where Nf represents the number of feature present in the tweet and count (f) represent the number of total feature present in the whole dataset. We use this formula for all the features of our experiment.

In **N-Gram Feature,** a sequence of 2-5 consecutive words in a sentence is regarded as a binary n-gram feature. The tweet which contains more rare words that have a higher weight than which contain common words and it has made a greater effect on the classification task.

In **Pattern Features,** the words are divided into three categories. They are high-frequency words (HFWs), content words (CWs) and regular words (RWs). When a word frequency is considered as *f* which frequency in the dataset is represent as *frf* and it will be considered as a high-frequency word if *frf* > *FH*. On the other hand, if *frf* < *FC*, then *f* is considered to be a content word and the rest of the words are considered as regular words. The word frequency is calculated from all the words of the dataset and it is estimated from the training set rather than from an external corpus. We treat as HFWs all consecutive sequences of punctuation characters as well as URL, REF, TAG and RT meta-words for pattern extraction as they play an important role in pattern detection. A pattern is an ordered sequence of HFWs and slots for content words. The upper bound for FC is set to 1000 words per million and the lower bound for FH is set to 10 words per million. FH is set to 100 words per million and we provide a smaller lower bound as the experimental evaluation produced better results. Observing that the FH and FC bounds allow overlap between some HFWs and CWs. By addressing this issue, we follow a simple strategy as described next. If f_{rf} € $\left(FH, FH + \frac{FC}{2}\right)$ the word is classified as HFW; otherwise, f_{rf} € $\left[FH + \frac{FC}{2}, FC\right)$ the word is classified as CW. We seek for patterns containing 2-6 HFWs and 1-5 slots for CWs. Moreover, we require patterns to start and to end with HFWs, thus a minimal pattern is of the form (HFW)(CW slot)(HFW).

TABLE. I. ACCURACY OF ALL ALGORITHMS OVER 1000 TWEETS

Method	Number of tweets	Precision	Recall	F-Score	TPR	FPR	Accuracy
Algorithm [9]	1000	0.81	0.76	0.78	0.79	0.13	79.99%
KNN with normalization (4 features)	1000	0.83	0.75	0.79	0.81	0.14	80.80%
KNN with normalization and keyword base(5 features)	1000	0.85	0.81	0.83	0.68	0.17	84.32%
SVM with (4 features	1000	0.56	0.69	0.61	0.78	0.49	58.79%
SVM with normalization (4 features)	1000	0.55	0.73	0.62	0.80	0.52	58.39%
SVM with normalization and keyword base (5 features)	1000	0.62	0.79	0.70	0.79	0.50	67.03%
SVM with normalization and keyword base (5 features) with grid search	1000	0.72	0.89	0.80	0.70	0.30	77.97%

In **Punctuation Feature,** the last feature type is divided into five generic features as: 1) tweet length in words, 2) number of exclamation mark characters in the tweet, 3) number of question mark characters in the tweet, 4) number of quotes in the tweet, and 5) number of capital/capitalized words in the tweet. The weight *wp* of a punctuation feature p is defined as $wp = (3 * Np)/(Mp * (Mw + Mng + Mpa))$, where *Mw*; *Mng*; *Mpa* declare the maximal values for word, n-gram and pattern feature groups, respectively. So, *wp* is normalized by averaging the maximal weights of the other feature types.

In the **Key-based feature,** we use a list [14] where there are 18000 words with its sentiment strength whose range falls within 1 to -1. Based on these words strength, we calculate the key based feature weight [15].

IV. EXPERIMENTAL RESULT & PERFORMANCE EVALUATION

In this section, based on the result of Sentiment Classification Algorithm (SCA) and Support Vector Machine we try to evaluate the performance of different algorithms. Our key parameters to evaluate performance are Accuracy, Recall, and Precision, etc. We have used a few open source machine learning library [16]-[18] during performance evaluation.

Here we give the accuracy of all algorithms on 1000 tweets. From the result, there are four attributes whose are precession, recall, F-Score, and Accuracy. Precision (also called positive predictive value) is the fraction of retrieved instances that are relevant while recall (also known as sensitivity) is the fraction of relevant instances that are retrieved. Both precision and recall are therefore based on an understanding and measure of relevance. F-score is calculated from precision and recall. According to the result, we can get an overview of the performance of the algorithms for different size of the dataset but for better analysis, we should compare the results based on a different parameter which is easily represented via graphs.

A. Performance Evaluation Based on Accuracy

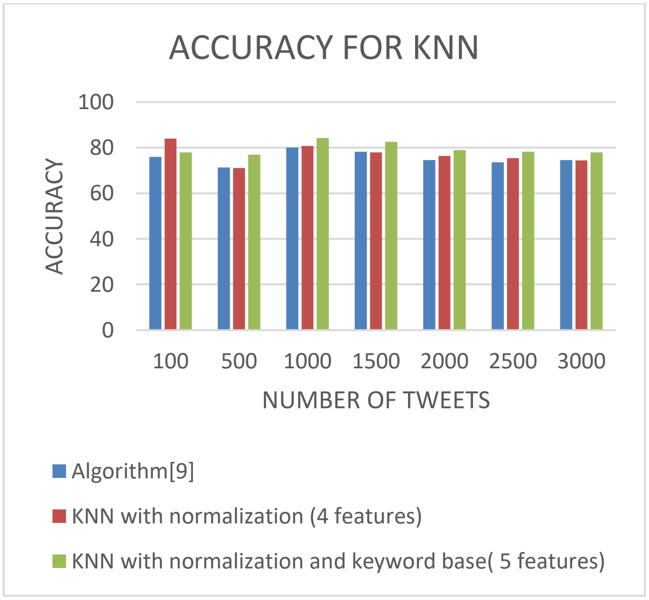

Fig. 1. Accuracy of KNN.

From Fig. 1, we can see that this graph show the accuracy of different versions of KNN for different size of the dataset. From the graph, we can see that when we use the original version of KNN with four features we get an accuracy which is increasing for all the dataset when we normalize the dataset. It is much more increasing when we add a feature name keyword-based feature.

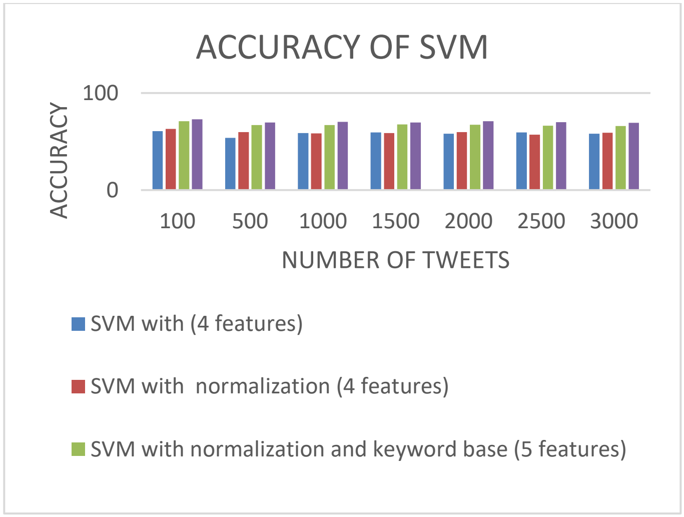

Fig. 2. Accuracy of SVM.

This graph (Fig. 2) shows the accuracy of the SVM. From the graph, we see that the accuracy of SVM with four features is around 60 percent for different size of the dataset. This is increased by normalization of the dataset. When we add another feature, keyword base features its accuracy much more increasingly and finally, we get accuracy around 70% by using grid search for every dataset.

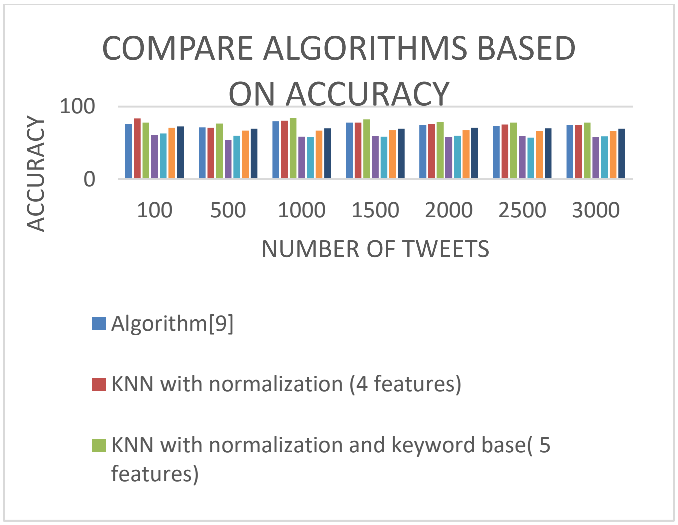

Fig. 3. Comparing algorithms based on accuracy.

From Fig. 3, we see that the accuracy of different versions of KNN is always higher than the accuracy of the different versions of SVM. From the accuracy, we see that KNN performs better than SVM for all the dataset.

If we only consider the accuracy only then it may be misleading us. Sometimes a model has greater predictive power on the problem with lower accuracy may be desirable to select.

If we consider a problem where there exist a large class imbalance and there may be a model which can predict the value of the majority and high classification accuracy achieved by it but this kind of model is not useful in the problem domain.

For this reason, we need to consider some additional measures like as precession, recall to evaluate a classifier.

B. *Performance Evaluation based on Precision*

When evaluating classifiers it is also necessary to consider the precession because precession can be thought as a measure of classifier exactness. A low precession indicates a large number of false positive. Sometimes it may occur that a classifier has low accuracy but it gives high precision. In a problem where exactness is more important than the high accuracy than the precession evaluating is very important.

Precision is calculated from the number of true positives divided by the number of true positives and false positives

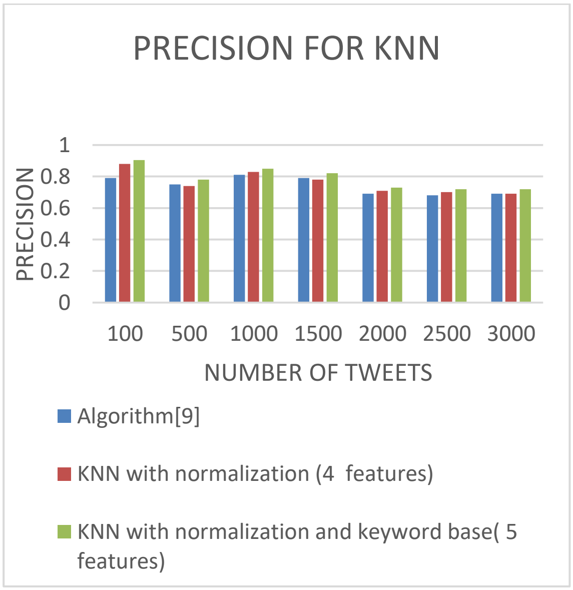

Fig. 4. Precision of KNN.

From Fig. 4, we see that the precision of the original KNN (four features) is lower than the updated version of KNN which also contains four features but the dataset is normalized. The precision is also much more increases by adding another feature keyword base feature with the previous version of KNN.

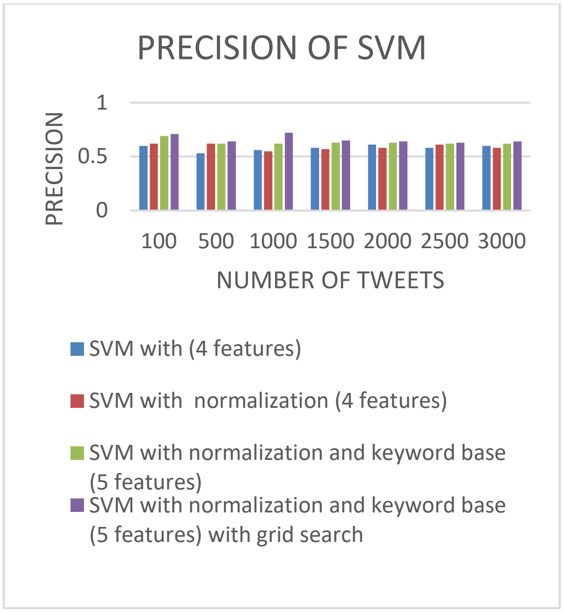

Fig. 5. Precision of SVM.

Fig. 5 contains the information about the precision of different version of SVM. When we measure the precision of SVM with four features we get precision which can be increased by normalizing it. It goes much higher when we add another feature and using grid search.

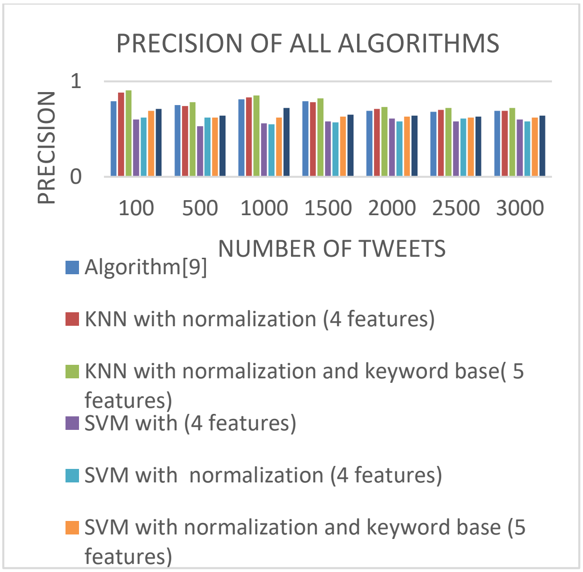

Fig. 6. Comparing algorithms based on precision.

As the precision of different versions of KNN is higher than the different versions of SVM (from Fig. 6), we can conclude that based on precision, KNN performs better than SVM.

C. Performance Evaluation Based on Recall

The recall is also important in classifier performance evaluation. It can be considered as a measure of a classifier's completeness and a low recall indicates many false negatives.

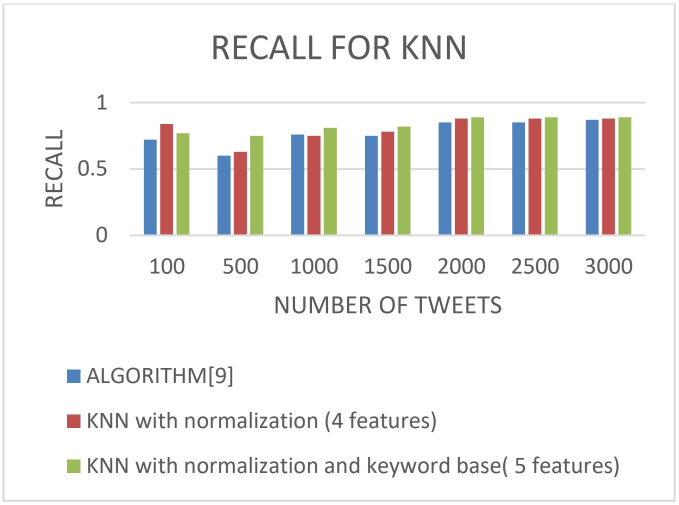

Fig. 7. Recall for KNN.

The recall is calculated from the number of true positives divided by the number of true positive and the number of false negatives.

The different versions of KNN provide good recall for different size of the dataset as shown in Fig. 7. As we modify the version of KNN, recall is increased than the previous version and it goes up to around 90% which is satisfactory.

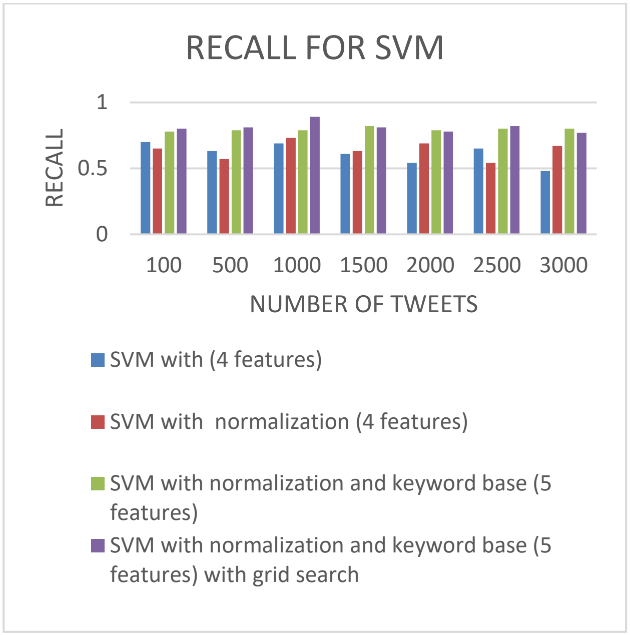

Fig. 8. Recall for SVM.

Performance based on recall is also good for the different versions of SVM as shown in Fig. 8. As version upgraded recall is also increased. SVM with four features without normalization and with normalization version recall is around 60% to 70% and it is going to 80 up when we added another feature and using grid search.

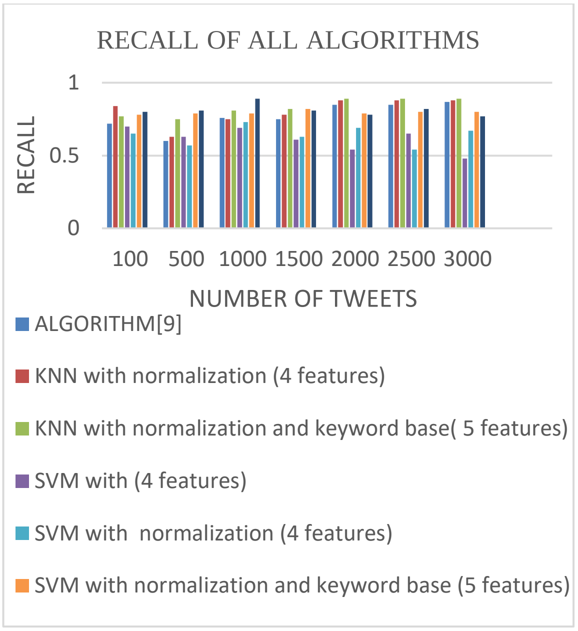

Fig. 9. Comparing algorithms based on recall.

From Fig. 9 which indicates the comparison among different versions of KNN and SVM, we see that the last two versions of SVM and KNN almost close to each other and the percentage are very good.

D. ROC Graph for Performance Evaluation

For selecting classifiers based on their performance receiver operating characteristics (ROC) graph is a good technique [19]. By plotting the true positive rate (TPR) against the false positive Rate (FPR) the curve is created. The true positive rate indicated the sensitivity or probability of detection and false positive rate indicate the fall-out or probability of false alarm.

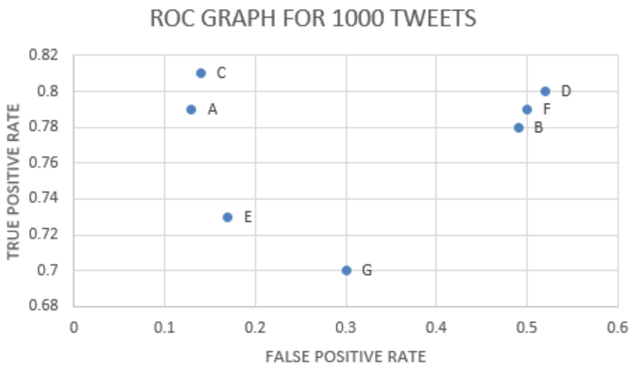

Fig. 10. ROC graph for 1000 tweets.

A: Original (KNN) (4 features).

B: KNN with normalization (4 features).

C: KNN with normalization and keyword base (5 features).

D: SVM with (4 features).

E: SVM with normalization (4 features).

F: SVM with normalization and keyword base (5 features).

G: SVM with normalization and keyword base (5 features) with grid search.

Fig. 10 shows the performance of the algorithms for 1000 tweets. We know that according to the ROC graph the performance of an algorithm is better than the other algorithms if its true positive rate is high and false positive rate is low. Based on this, if we plot the true positive rate in Y-axis and false positive rate in X-axis then we get a point in a two-dimensional plane for an algorithm. The more the algorithm is left and upper side the performance of the algorithms is better than the other algorithms. Fig. 10 shows A, C, and E performance better compared with other algorithms. According to the ROC graph, the algorithm is the best for which the data point is northwest corner compare to other algorithms that algorithm is best. But in our figure, no data point can fulfill the requirement. But we can easily say that B, D, and F perform worse than other algorithms. As G has less True positive rate than B, F, and D it has also less false positive rate compared to them. Comparing between A and C algorithms A has less true positive rate and false positive rate than C but for the C True positive rate is much more increases than the false positive rate increase.

We can see from all the graphs draw based on different parameters all time KNN always performs better than SVM. There are some reasons behind this kind of performance. Those reasons are given below.

SVM performs better when the number of dimensions is very high. But in our experiment, we only use five features and every data point represents in five dimensions. For a lot of points in few dimensions, SVM cannot perform better.

As the tweets are collect randomly and it is not guaranteed that which dataset we use as training dataset the number of positive and negative tweets are equal. It may also produce highly imbalanced dataset when we use k-fold-cross validation algorithm. Imbalance dataset means the difference between the number of positive and negative tweets is very high in the training dataset.

Learning factor c and gamma vary based on the dataset. Finding the best pair of c and gamma for a particular dataset is very tough. We try to find the best c and gamma by using grid search algorithm from some particular values of c and gamma. Grid search algorithm returns the best pair of c and gamma but there may exist better c and gamma.

SVM always assumes a hyperplane exist between the classes. But sometimes it may be very difficult to determine the hyperplane for the position of the data point in the dataset.

V. Conclusion and Future Work

Sentiment analysis based on micro-blogging is still in the developing stage and far from complete. As an example a positive sentiment is “It is a nice Day!” and a negative sentiment is “it is a horrible day!” In this paper, we will try to find out the positive and negative sentiment on Twitter data.

Currently, we have worked with a simple model and in our work, we design our classifier with only a few features like n-gram feature, pattern feature, punctuation feature, keyword-based feature and word feature. We also use machine learning algorithm SVM (support Vector Machine). We also use KNN classifier and calculate the accuracy of all algorithms. In this paper, we are focusing on dividing the tweets into positive and negative sentiment. In our work, we see that sentiment classifier algorithm (SCA) performs better than SVM.

In future, we would study further many related problems. For this, we will try to improve our models by adding some extra features. In this paper, we work with only the English tweets and we are not considering the emoticons tweet. So our next plan is to work with other language tweets and add the emoticons tweets. Apart from this, we will also try to detect another sentiment label of human being and at the same time, we will work with a big amount of tweets. From this, we can say that our future work list may contain the following actions:

- *Adding some extra features*: Adding some features will helps us to detect sentiment more correctly and provide a better result than the present result.
- *Working with others language*: In our work, we use Java language and Java JAR files. In future, we can use different language.
- *Working with emoticons tweets*: We only work with text tweets. In future, we need to work with text tweets as well as emoticons tweets.
- *Focusing on detecting another sentiment label of human being*: We only work with positive and negative sentiment label. We will extend our work to consider other sentiment labels.
- *Working with a lot of tweets*: In future, we would like to work with a dataset which contains a large number of tweets.
- *Accuracy calculation and performance evaluation*: In current work, we use confusion matrix for calculating accuracy and performance evaluation. In future Apply others machine learning algorithms to calculate accuracy and performance evaluation.
- *Working with real world problems*: Given an efficient sentiment label, we will try to see how it can be applied to solving real-world problems. As for example, predicting presidential election, estimating product reputation, etc.

In this paper, we are mainly focusing on general sentiment analysis like the positive and negative sentiment. There is the potential of work in the field of sentiment analysis and we will try to use our knowledge in this field. On the other hand, we would like to compare sentiment analysis with other domains.

References

[1] AnalyticsVidhaya: https://www.analyticsvidhya.com/blog/ 2015/10/ understaing-support-vector-machine-example-code/ (accessed on 12/11/16)

[2] Sebastianraschka:http://sebastianraschka.com/Articles/ 2014_pca_step_by_step.html(accessed on 10/11/16)

[3] StackOverflow:http://stackoverflow.com/questions/19335165/cross-validation-and-grid- Search (accessed on 10/11/16)

[4] Cs.uregenia: http://www2.cs.uregina.ca/~dbd/cs831/notes/ confusion_matrix.html (accessed on 10/11/16)

[5] Apoorv Agarwal, Boyi Xie,Ilia Vovsha, Owen Rambow, Rebecca Passonneau: Sentiment Analysis of Twitter Data. In: Proceedings of the workshop on Language in Social Media (LSM), pages 30-38, 2011.

[6] Luciano Barbosa, Junlan Feng: Robust Sentiment Detection on Twitter from Biased and Noisy Data. In 23rd International Conference on Computational Linguistics, 2010.

[7] Efthymios Koulompis, Theresa Wilson, Johanna Moore: Twitter Sentiment Analysis: The Good the Bad and the OMG!.In: Proceeding of the Fifth International AAAI Conference on Weblogs and Social Media, 2011.

[8] Hassan Saif, Yulan He and Harith Alani: Semantic Sentiment Analysis of Twitter. In: Proceedings of the 11th International Semantic Web Conference, 2012.

[9] Nikolaos Nodarakis, Athanasios Tsakalidis, Spyros Siouts, Giannnis Tzimas: Large Scale Sentiment Analysis on Twitter with Spark. In: Proceedings of the First International Workshop on Multiengine Data Analytics, 2016.

[10] Archive.org:https://archive.org/details/twitter/cikm_2010 (accessed on 07/07/2016)

[11] Sentiment140: http://help.sentiment140.com/api (accessed on 10/07/2016)

[12] Github:https://github.com/aababu/sentiment-analysis/blob/master/stopword.txt (accessed on (29/11/16)

[13] Adam Bermingham, Alan F. Smeaton: Classifying sentiment in microblogs: is brevity an advantage? In: Proceedings of the 19th International Conference on Information and Knowledge Management, ACM, 2010, pp. 1833-1836.

[14] Michael Gamon. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In: Proceedings of the 20th International Conference on Computational Linguistics, 2004.

[15] Github:https://github.com/aababu/sentiment-analysis/blob/master/wordstrength.txt(accessed on 29/11/16)

[16] Sourceforge:https://sourceforge.net/projects/java-ml/files/java-ml/ (accessed on 10/08/16)

[17] Java2s: http://www.java2s.com/Code/Jar/l/Downloadlibsvm317sourcesjar.htm (accessed on 11/08/16)

[18] Java2s: http://www.java2s.com/Code/Jar/j/Downloadjama103jar.htm (accessed on 12/08/16)

[19] Tom Fawcett: ROC Graphs: Notes and Practical Considerations for Data Mining Researchers. In: Journal of Pattern Recognition Letters – Special issue in Roc analysis in pattern recognition archive, 2006.