ISSN No. 0976-5697

Volume 2, No. 5, Sept-Oct 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Machine Learning Methods in Classification of Text by Sentiment Analysis of Social Networks

I.Hemalatha*
Dept. of Information Technology
S.R.K.R. Engineering College
China-Amiram,Bhimavaram-04,A.P,India
hema_prasad2002@yahoo.com

Dr. G.P.Saradhi Varma
Dept. of Information Technology
S.R.K.R. Engineering College
China-Amiram,Bhimavaram-04,A.P,India
gpsvarma@yahoo.com

Dr. A.Govardhan
Principal & HOD/ Prof in CSE,
JNTUH College of Engg, Jagityal, India
govardhan_cse@yahoo.co.in

Abstract: In recent years, we became witnesses of a large number of websites that enable users to contribute, modify, and grade the content. Users have an opportunity to express their personal opinion about specific topics. The examples of such web sites include blogs, forums, product review sites, and social networks. Micro-blogs are a challenging new source of information for data mining techniques. It allows users to publish brief message updates, which can be submitted in many different channels, including the Web and text messaging service. One of the most notable micro-blogging services is Twitter, Face Book Etc.,. Twitter messages are short, and generated constantly, and well suited for knowledge discovery. This paper presents an empirical study of efficiency of machine learning techniques in classifying text messages by sentiment Analysis.

Keywords: Blogs, Machine Learning, Sentiment Analysis.

I. INTRODUCTION

Sentiment analysis can be cast as a classification problem where the task is to classify messages into two categories depending on whether they convey positive or negative feelings. Twitter sentiment analysis is not an easy task because a tweet can contain a significant amount of information in very compressed form, and simultaneously carry positive and negative feelings.

Twitter is a website, owned and operated by Twitter Inc., which offers a social networking and microblogging service, enabling its users to send and read messages called *tweets*. Tweets are text-based posts of up to 140 characters displayed on the user's profile page.

Figure 1: Twitter Home page

II. SYSTEM FOR SENTIMENT ANALYSIS

Sentences that twitters are interested in are distilled automatically. Based on the topic sentence distilled, topic-specific relationship features are extracted from each tweet. We categorize features into two groups. Finally we apply the learning algorithms to measure the sentiment results (positive, negative or objective).

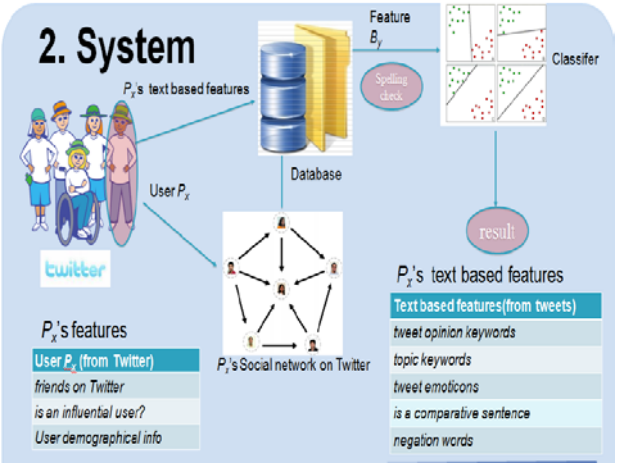

Figure 2: System for sentiment analysis

Having the data collected, features defined and preprocessing done, then for each tweet, we extract all its features: the number of positive words, the number of negative

words, what emoticons does it contain, if any, etc. After this is done, we export all the tweets along with their features to learners including a baseline method we defined.

III. MACHINE LEARNING APPROACH

A. Classification: Problem Description:

Zhou [3] described the process of text classification as follows: given a document collection {D1, D_2,..., D_n }, a category C_i from the predefined category collection {C_1, C_2,...,C_m } has to be assigned to a test document D_j. Thus we aim to estimate a mapping function f: D x C -> {0, 1} (that describes how documents ought to be classified). The mapping function between collection D and collection C is : D x C {d_jc_i | j = 1,2,.. ,n , i = 1,2, ,m} and the value of element d_jc_i is 1 means that we classify document d_j under category c_i, otherwise, the value of element d_jc_i is 0 means we do not classify document d_i under category c_i .

B. Preprocessing for Text Classification:

For every document, *Stopword filtering* (where frequently occurring words of little importance are filtered out from the document) and *Stemming* (where different morphological variants of a word are reduced to a common root called a *stem*} are applied on each of the training data text files. The final set of words forms the vocabulary of the training set.

The documents are represented in the *Vector Space Model*. The core idea is to make the document become a numeral vector in multi-dimension space. Every dimension represents a word and the corresponding numerical value represents the *term frequency*, *term frequency- inverse sentence frequency* (tf-isf) or *term frequency-inverse document frequency* (tf-idf).

C. Base Line Method:

Using all available positive and negative examples in our "tagged" dictionary, including three steps:

a. Extract each word for a given sentence.
b. Calculate the positive and negative words tagged in the dictionary.
c. Classify the sentence after comparing the quantity of positive and negative examples.
d. If there are the same numbers of positive and negative ones, it outputs a random guess.

IV. MACHINE LEARNING ALGORITHMS

A. Supervised Algorithms:

a. Decision tree:

Decision tree learning uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value.

b. Naïve Bayes:

As it is typical for text categorization problems we represent documents using the standard bag-of-words approach, therefore each document D is represented as: D = {w1,w2, ...,wm} where wi is typically a single word (i.e. token) and m is the number of unique words in the training set.

The classifier function on the same premise, i.e. to maximize the posterior probability P(c|D) that document D belongs in class c. Typically the best class is the maximum a posteriori (MAP) class c_{MAP} :

$$c_{MAP} = \arg\max_{c \in C} \{P(c|D)\}$$

……..……..[1]

In practice, that means that we estimate $P(c_i|D)$ for all i's and choose the class with the highest probability. The way that P(c|D) is estimated by the Naïve Bayes classifier.

We apply the Bayes rule to equation 1:

$$c_{MAP} = \arg\max_{c \in C} \left\{ \frac{P(D|c) * P(c)}{P(D)} \right\} \propto \arg\max_{c \in C} \{P(D|c) * P(c)\}$$

…….[2]

where we've removed the denominator P(D) since it doesn't influence the outcome of the classification. P(c) if the prior probability of class c, i.e. the relative frequency of the class and can be defined as P(c) = |{D|D's class is c}|/|D|. The Naive Bayes classifier assumes that all features are conditionally independent given class c, therefore:

$$P_{NB}(D|c) = \prod_{i=1}^{m} P(w_i|c) = \prod_{i=1}^{m} \frac{\#(w_i, c)}{\#(w_i)}$$

……..[3]

where #(wi, c) is the number of times that token wi has been encountered in documents of class c in the training data set and #(wi) is the number of times that the token has occurred in all documents in the training data set. Lastly, in order to avoid zero probabilities, we apply Laplace add-one smoothing, therefore:

$$P_{NB}(D|c) = \prod_{i=1}^{m} \frac{\#(w_i, c) + 1}{\#(w_i) + m}$$

……..…[4]

Despite its simplicity, Naive Bayes classifiers have been used in a number of classification tasks and have been found to perform very adequately in most cases.

c. Support Vector Machine:

SVM, which bases on Statistical Learning Theory [3], is a universal machine learning method. It could learn in high dimension feature space using linear function space. For, it has better model extended performance and easily optimizes model structure and parameters, SVM has been widely used in regression estimation, pattern recognition, fault diagnosis, system identification, and so on. SVM derives from the optimal hyper plane under linearly separable cases. The constrained optimization problem is to find the optimal hyper-plane and can be transformed into dual problem using Lagrange optimal method. That is

$$MaxQ(\alpha) = \Sigma_{i=1}^{n} \alpha_i - \frac{1}{2} \Sigma_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i \bullet x_j)$$ ……[5]

Where α_i is the Lagrange multiplier for all training sample data.

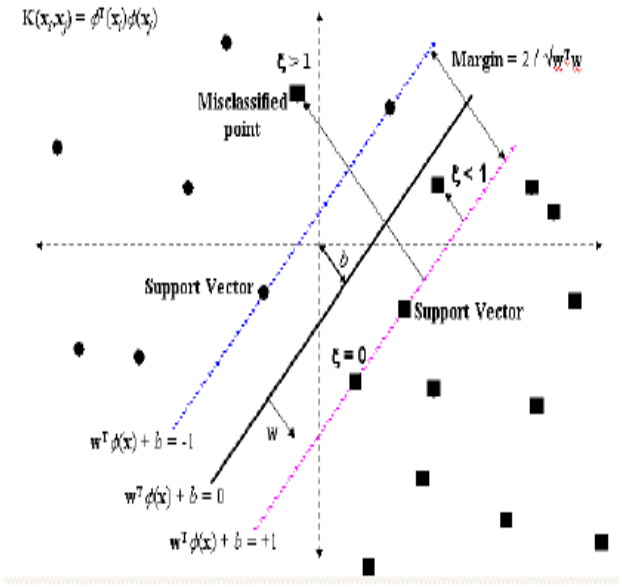

Figure 3: Non linear Support Vector Machines.

To solve the above problem, the optimal classification function can be obtained as follows:

$$f(x)=\mathrm{sgn}\{(\omega \bullet x)+b\}=\mathrm{sgn}\Sigma_{i,j=1}^{n}\alpha_i^* y_i y_j (x_i \bullet x_j)+b^* \quad \text{.....[6]}$$

Introducing the concept of kernel function K(x_i , y_i) inner product (x_i , y_i) in (1) is replaced by the inner-inner product kernel K(x_i , y_i) and the original data space will be transformed to a new feature space. The new optimization objective function can be rewritten as:

$$Q(\alpha)=\Sigma_{i=1}^{n}\alpha_i-\frac{1}{2}\Sigma_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \quad \text{.....[7]}$$

Accordingly the discriminate function is replaced by

$$f(x)=\mathrm{sgn}\Sigma_{i,j=1}^{n}\alpha_i^* y_j K(x_i,x_j)+b^* \quad \text{.....[8]}$$

In the new space, the primary linearly inseparable problem becomes linearly separable problem, and then can be solved according to linearly separable cases. The learning machine which constructs the discriminate function as (8) is called Support Vector Machine.

V. EXPERIMENTS AND RESULTS

We obtained a set of top-1000 twitters, crawled all the tweets they published so far (total 1,021,039) and then repeat for all the followers and friends of each individual twitter. To build classifier for sentiment analysis, we need to collect training data so that we can apply appropriate learning algorithms. Labeling tweets manually as positive or negative may be a feasible way. We have labeled about 1000 items of tweets. Annotated tweets can be used to train a sentiment classifier.

Then for preprocessing, we utilize the POS tagging and spelling check function of the openNLP toolkit. following charts explain the performances of the different classifiers we tested and their comparisons with the baseline method.

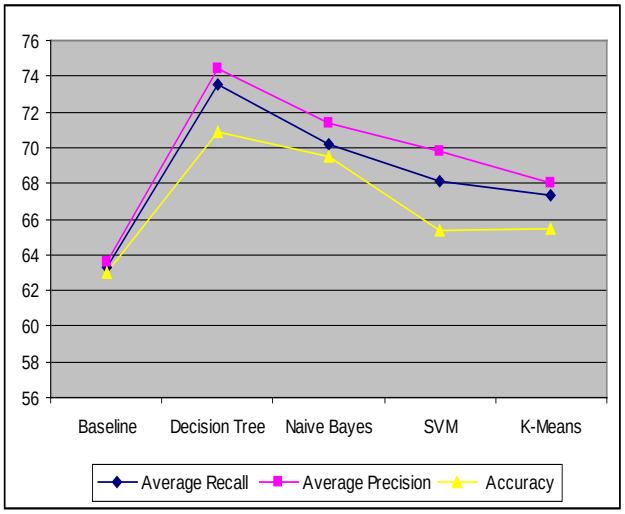

Figure 4: Performance of positive/negative classification

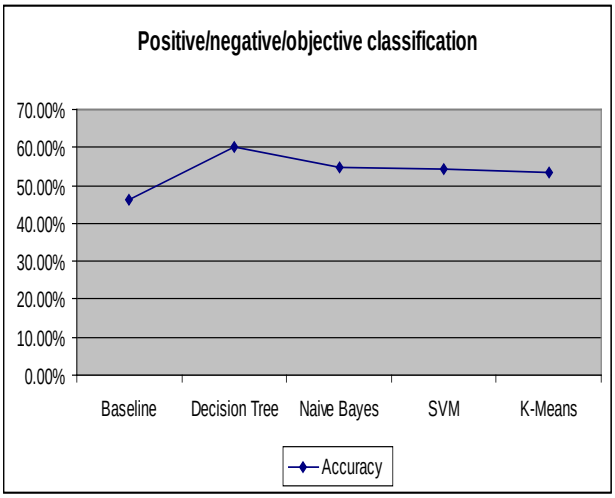

Figure 5: Performance of positive/negative/objective classification

VI. RESULTS

Baseline: the baseline method for sentiment classification problem (positive or negative) can get 63.28%, 63.55% and 62.95% for recall, precision and accuracy respectively, which is pretty good. But if we consider objective sentences, it performs badly, which can just get 46.0% in accuracy.

Table 1: Influences of different features

Feature Component	Accuracy (SVM)	Accuracy (Naïve Bayes)	Accuracy (Decision Tree)
All of the component	54.3%	54.8%	60.3%
Text Based F(x)	54.5%	55.1%	60.2%
Pos + Neg words	51.7%	51.2%	56.3%
Pos + Neg words +Negation	53.1%	53.2%	59.4%
Pos + Neg words +Negation +Emoticon	53.9%	54.2%	60.3%

For the results posted in the Table 1, both text-based and user-based features are considered. Here, we study how these factors affect the prediction of sentiment analysis individually. That is, how would the model perform if we use subset of them for prediction? In this section, we train the individual models with the corresponding feature vectors and measure the accuracy for each one. For POS, NEG and OBJ classification problem, we use SVM, Decision Tree and Naive Bayes.

Generally speaking, all of features considered are very important to the accuracy of classification. The performance of Decision Tree keeps increasing as we consider more and more features. But is the always the concern of over-fitting which is reflected in the performance of Naïve Bayes.

VII. CONCLUSION

Twitter sentiment analysis is not an easy task because a tweet can contain a significant amount of information in very compressed form, and simultaneously carry positive and negative feelings. Based on this observation, we propose a novel framework for this task: first, sentences that twitters are interested in are distilled automatically. Based on the topic sentence distilled, topic-specific relationship features are extracted from each tweets. Finally we apply the learning algorithms to measure the sentiment results. In order to obtain substantial improvements from the results we have achieved, we think advanced NPL techniques should be employed in future efforts.

VIII.REFERENCES

[1] Kennedy, A., Inkpen, D.: Sentiment classification of movie reviews using contextual valence shifters. Computational Intelligence 22(2), 110–125 (2006).

[2] Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. 2005. Emotions from text: machine learning for textbased emotion prediction. In Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, pages 579–586.

[3] Mullen, T., Collier, N.: Sentiment analysis using support vector machines with diverse information sources. In: Lin, D., Wu, D. (eds.) Proceedings of EMNLP 2004. pp. 412–418. Association for Computational Linguistics, Barcelona, Spain (July 2004).

[4] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10, pages 79_86. Association for Computational Linguistics, 2002.

[5] P. Melville, W. Gryc, and R.D. Lawrence. Sentiment analysis of blogs by combining lexical knowledge with text classification. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1275_1284. ACM, 2009.