Twitter Sentiment Analysis and Opinion Mining

Ravikiran Janardhana
Department of Computer Science,
University of North Carolina at Chapel
Hill

ravikirn@cs.unc.edu

ABSTRACT

I introduce a novel approach for automatically classifying the sentiment of Twitter messages. These messages are classified as positive or neutral or negative with respect to a query term. This is useful for consumers who want to research the sentiment of products before purchase, or companies that want to monitor the public sentiment of their brands. Previous research on classifying sentiment of messages on microblogging services like Twitter have tried to solved this problem but have ignored neutral tweets which leads to wrong sentiment classification and I have tried to solve this problem in this project. I present the results of machine learning algorithms for classifying the sentiment of Twitter messages using a novel feature vector. My training data consists of publicly available twitter messages obtained through automated means. I show that machine learning algorithms (Naive Bayes, Maximum Entropy, and SVM) can achieve competitive accuracy when trained using my feature vector and the publicly available dataset. This report also describes the preprocessing steps of the dataset needed in order to achieve high accuracy. The main contribution of this project is the novel feature vector of weighted unigrams used to train the machine learning classifiers.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Natural Language Processing

General Terms

Algorithms

Keywords

Twitter, sentiment classification, opinion mining, sentiment analysis

1. INTRODUCTION

Twitter is a popular microblogging service where users create status messages (called "tweets"). These tweets sometimes express opinions about different topics. I propose a method to automatically extract sentiment (positive or neutral or negative) from a tweet. This is very useful because it allows feedback to be aggregated without manual intervention. Consumers can use sentiment analysis to research products or services before making a purchase. Marketers can use this to research public opinion of their company and products, or to analyze customer satisfaction. Organizations can also use this to gather critical feedback about problems in newly released products. There has been a large amount of research in the area of sentiment classification. Traditionally most of it has focused on classifying larger pieces of text, like reviews [9]. Tweets (and microblogs in general) are different from reviews primarily because of their purpose: while reviews represent summarized thoughts of authors, tweets are more casual and limited to 140 characters of text. Generally, tweets are not as thoughtfully composed as reviews. Yet, they still offer companies an additional avenue to gather feedback. Previous research on analyzing blog posts includes [6]. Pang et al. [9] have analyzed the performance of different classifiers on movie reviews. The work of Pang et al. has served as a baseline and many authors have used the techniques provided in their paper across different domains. In order to train a classifier, supervised learning usually requires hand-labeled training data.

With the large range of topics discussed on Twitter, it would be very difficult to manually collect enough data to train a sentiment classifier for tweets. Hence, I have used publicly available twitter datasets which are in turn obtained via distant supervision proposed in [12]. However, this dataset consist only of positive and negative tweets. For neutral tweets, I have used the publicly available neutral tweet dataset provided by [13]. I run the machine learning classifiers Naïve Bayes, Maximum Entropy and Support Vector Machine trained on the positive and negative tweets dataset and the neutral tweets against a test set of tweets.

To help visualize the utility of the Twitter-based sentiment analysis tool, I have built a web application tool [14]. This can be used by individuals and companies that may want to research sentiment on any topic.

1.1 Defining the sentiment

For the purpose of research, I define sentiment to be "a personal positive or negative feeling" and when there is an absence of this, I treat it as a neutral sentiment. Table 1 shows some examples.

Table 1. Example Tweets

Sentiment	Keyword	Tweet
Positive	Football	Dammmmm I Love Football
Neutral	airplane	Comes 8 a clock, phone going on airplane mode.
Negative	Pep Guardiola	Pep Guardiola to resign as Barcelona boss

1.2 Characteristic of Tweets

Twitter messages have many unique attributes, which differentiates my work from previous research:

Length The maximum length of a Twitter message is 140 characters. This is very different from the previous sentiment classification research that focused on classifying longer bodies of work, such as movie reviews.

Language model Twitter users post messages from many different media, including their cell phones. The frequency of

misspellings and slang in tweets is much higher than in other domains.

Domain Twitter users post short messages about a variety of topics unlike other sites which are tailored to a specific topic. This differs from a large percentage of past research, which focused on specific domains such as movie reviews.

2. RELATED WORK

This project builds on the ideas proposed in [12] where the authors classify tweets using unigram features and the classifiers are trained on data obtained using distant supervision. Read [10] shows that using emoticons (distant supervision) as labels for positive and sentiment is effective for reducing dependencies in machine learning techniques and this idea is heavily used in [12]. Pang and Lee [9] researched the performance of various machine learning techniques in the specific domain of movie reviews.

However, the previous methods have not taken into account neutral tweets which lead to wrong classification and this project tries to solve this problem by including neutral tweets in the training dataset and using a novel feature vector to train the machine learning classifier and tries to classify a given tweet as positive or negative or neutral.

3. APPROACH

My approach is to use different machine learning classifiers and feature extractors. The machine learning classifiers are Naive Bayes, Maximum Entropy (MaxEnt), and Support Vector Machines (SVM). The feature extractors are unigrams and unigrams with weighted positive and negative keywords. I build a framework that treats classifiers and feature extractors as two distinct components. This framework allows us to easily try out different combinations of classifiers and feature extractors.

3.1 Query Term

I normalize the effect of query terms. My assumption is that the user wants to perform sentiment analysis about a product and not of a product. If a query term has a positive/negative sentiment by itself, this will bias the results.

3.2 Emoticons

Since the training process makes use of emoticons as noisy labels, it is crucial to discuss the role they play in classification. I will discuss in detail the training and test set in the Evaluation section. I strip the emoticons out from the training data. If I leave the emoticons in, there is a negative impact on the accuracies of the MaxEnt and SVM classifiers, but little effect on Naive Bayes. The difference lies in the mathematical models and feature weight selection of MaxEnt and SVM. Stripping out the emoticons causes the classifier to learn from the other features (e.g. unigrams) present in the tweet. The classifier uses these non-emoticon features to determine the sentiment.

3.3 Feature Reduction

The Twitter language model has many unique properties. I take advantage of the following properties to reduce the feature space:

Usernames Users often include Twitter usernames in their tweets in order to direct their messages. A de facto standard is to include the @ symbol before the username (e.g. @ravikiranj). An equivalence class token (AT_USER) replaces all words that start with the @ symbol.

Usages of links Users very often include links in their tweets. An equivalence class is used for all URLs. That is, I convert a URL like "http://tinyurl.com/cvvg9a" to the token "URL".

Stop words There are a lot of stop words or filler words such as "a', "is", "the" used in a tweet which does not indicate any sentiment and hence all of these are filtered out. The complete list of stop words can be found at [15].

Repeated letters Tweets contain very casual language. For example, if you search "hungry" with an arbitrary number of u's in the middle (e.g. huuuungry, huuuuuuungry, huuuuuuuuuungry) on Twitter, there will most likely be a nonempty result set. I use preprocessing so that any letter occurring more than two times in a row is replaced with two occurrences. In the samples above, these words would be converted into the token "huungry".

Table 2 shows the effect of these feature reductions. These reductions shrink the feature set down to 8.74% of its original size.

Table 2. Effect of Feature Reduction

Feature Reduction Steps	# of Features	Percentage of Original
None	277354	100.00%
URL / Username / Repeated Letters	102354	36.90%
Stop Words ('a', 'is', 'the')	24309	8.74%
Final	24309	8.74%

3.4 Feature vector

After preprocessing the training set data which consists of 9666 positive tweets, 9666 negative tweets and 2271 neutral tweets, I compute the feature vector as below:

Unigrams As shown in Table 2, at the end of preprocessing we end up with 24309 features which are unigrams and each of the features have equal weights.

Weighted Unigrams Instead of weighing each of the unigrams equally, I introduce bias by weighing the positive and negative keywords more than the other features present in the feature vector. I use [16-17] as my focal point (a list) in order to weight the positive and negative keywords more compared to the remaining features.

4. MACHINE LEARNING METHODS

I test different classifiers namely keyword-based, Naïve Bayes, Maximum Entropy and Support Vector Machines

4.1 Baseline (keyword-based)

In this approach, I use the positive and negative keyword list in [16-17] and for each tweet, I count the number of positive and keywords that appear. This classifier returns the polarity of the highest count. If there is a tie, neutral polarity is returned.

4.2 Naïve Bayes

Naive Bayes is a simple model which works well on text categorization [5]. I use a multinomial Naive Bayes model. Class $c*$ is assigned to tweet d, where

$$c* = argmac_c P_{NB}(c|d)$$

$$P_{NB}(c|d) := \frac{(P(c)\sum_{i=1}^{m} P(f|c)^{n_i(d)})}{P(d)}$$

In this formula, *f* represents a feature and $n_i(d)$ represents the count of feature f_i found in tweet d. There are a total of m features. Parameters *P(c)* and *P(f|c)* are obtained through maximum likelihood estimates, and add-1 smoothing is utilized for unseen features. I used the Python based Natural Language Toolkit [18] library to train and classify using the Naïve Bayes method.

4.3 Maximum Entropy

The idea behind Maximum Entropy models is that one should prefer the most uniform models that satisfy a given constraint [7]. MaxEnt models are feature-based models. In a two class scenario, it is the same as using logistic regression to find a distribution over the classes. MaxEnt makes no independence assumptions for its features, unlike Naive Bayes. The model is represented by the following:

$$P_{ME}(c|d,\lambda) = \frac{\exp[\Sigma_i \lambda_i f_i(c,d)]}{\Sigma_{c'} \exp[\Sigma_i \lambda_i f_i(c,d)]}$$

In this formula, *c* is the class, *d* is the tweet, and λ is a weight vector. The weight vectors decide the significance of a feature in classification. A higher weight means that the feature is a strong indicator for the class. The weight vector is found by numerical optimization of the λ_i's so as to maximize the conditional probability.

I use the Python NLTK library to train and classify using the Maximum Entropy method. For training the weights I use conjugate gradient ascent.

Theoretically, MaxEnt performs better than Naive Bayes because it handles feature overlap better. However, in practice, Naive Bayes can still perform well on a variety of problems [7].

4.4 Support Vector Machines

Support Vector Machines is another popular classification technique [2]. I have used libsvm [19] library with a linear kernel. My input data are two sets of vectors of size m. Each entry in the vector corresponds to the presence a feature. In the unigram feature extractor, each feature is a single word found in a tweet. If the feature is present, the value is 1, but if the feature is absent, then the value is 0. I use feature presence, as opposed to a count, so that I do not have to scale the input data, which speeds up overall processing [1].

5. EVALUATION

5.1 Experimental setup

There are publicly available data sets of twitter messages with sentiment indicated by [12] and [13]. I have used a combination of these two datasets to train the machine learning classifiers. For the test dataset, I randomly choose 4000 tweets which were not used to train the classifier. The details of the training and test data are explained in Table 3 as below:

Table 3. Example Tweets

Dataset	Positive	Negative	Neutral	Total
Training	9666	9666	2271	21603
Test	Randomly chosen Tweets			4000

The Twitter API has a parameter that specifies which language to retrieve tweets in. I always set this parameter to English (en). Thus, my classification will only work on tweets in English because the training data is English-only.

I have also built a web interface which searches the Twitter API for a given keyword for the past one day or seven days and fetches those results which is then subjected to preprocessing as specified in Section 2.3. These filtered tweets are fed into the trained classifiers and the resulting output is then shown as a graph in the web interface. The web version of the twitter sentiment analyzer can be found at http://66.212.17.122:9999/.

6. RESULTS

I explore the usage of unigrams and weighted unigram features and Table 4 summarizes the results.

Table 4. Classifier Accuracy

Features	Naïve Bayes	Max Entropy	SVM
Unigrams	66.82	60.35	63.90
Weighted Unigrams	78.52	70.42	80.10

Unigrams The unigram feature vector is the simplest way to retrieve features from a tweet. The machine learning algorithms perform average with this feature vector. One of the reasons for the average performance might be the smaller training dataset of 20000+ tweets. If one could get hold of millions of tweets and train these classifiers, the accuracy would improve substantially. Twitter API places a limit of 150 unauthorized requests per hour and hence one can download only 3600 tweets per day via the labeled tweet IDs from the publicly availably tweet ID dataset.

Weighted Unigrams In this approach, I took advantage of the fact that it makes sense to weight the positive and negative keywords more than other words while trying to classify the sentiment of a tweet and this trick produced competitive accuracy as shown in Table 4. As expected, SVM performed the best with 80.10% accuracy and surprisingly Naïve Bayes outperformed Max Entropy by a substantial margin i.e. 78.52% to 70.42%. This is in accordance with the results shown by Pang and Lee [9].

Figures 1-4 show the twitter sentiment analysis of topics 'Ron Paul', 'Newt Gingrich', 'Obama' and 'Mitt Romney' between 20-Apr-2012 to 25-Apr-2012 using the Naïve Bayes Classifier.

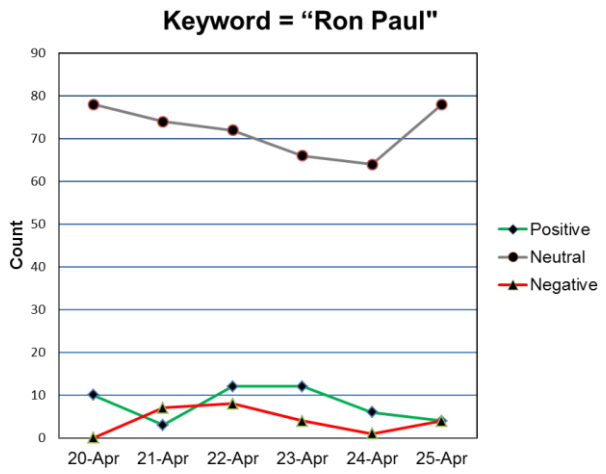

Figure 1. Twitter Sentiment Analysis of ‘Ron Paul'

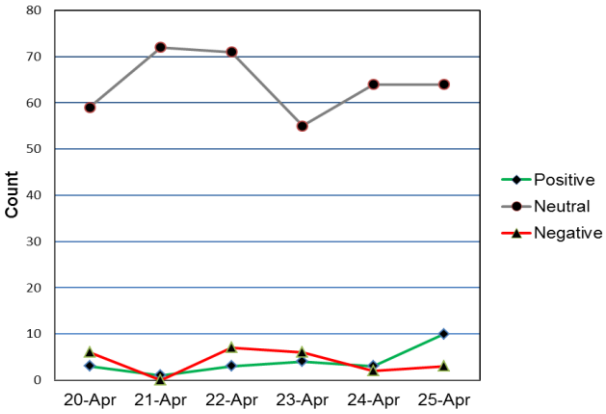

Figure 2. Twitter Sentiment Analysis of ‘Newt Gingrich'

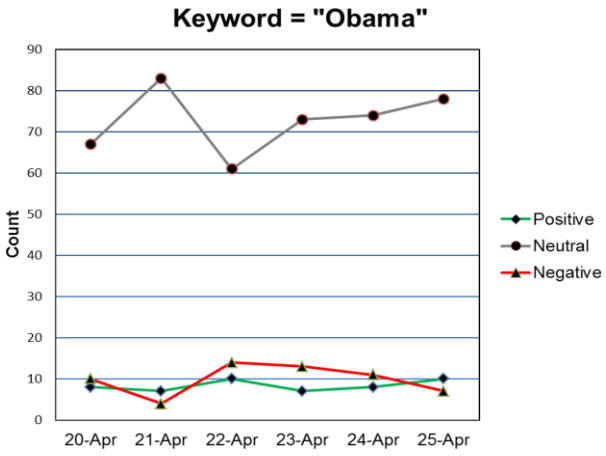

Figure 3. Twitter Sentiment Analysis of ‘Obama'

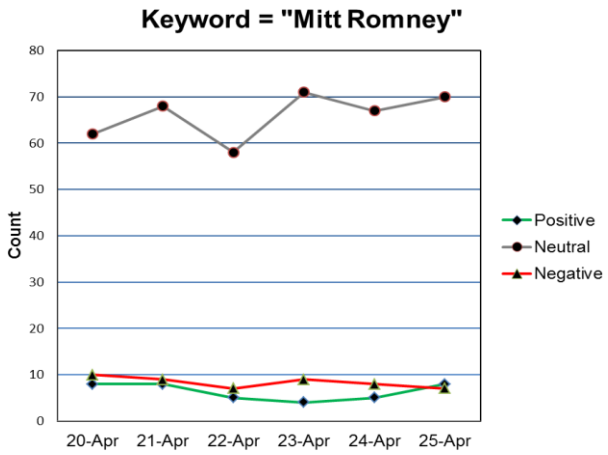

Figure 4. Twitter Sentiment Analysis of ‘Mitt Romney'

It is to be noted that ‘Obama’ had a lot of negative press at the early part of the week during 20-Apr to 25-Apr. However, his speech at UNC on 25-Apr lifted his positive sentiment and overall public opinion as noted by reputed media stream and this is seen in Figure 3, thus validating the results of the twitter sentiment analysis.

7. FUTURE WORK

Machine learning techniques perform well for classifying sentiment in tweets. I believe the accuracy of the system could be still improved. Below is a list of ideas I think could help the classification:-

Semantics The algorithms classify the overall sentiment of a tweet. The polarity of a tweet may depend on the perspective you are interpreting the tweet from. For example, in the tweet “Federer beats Nadal :)”, the sentiment is positive for Federer and negative for Nadal. In this case, semantics may help. Using a semantic role labeler may indicate which noun is mainly associated with the verb and the classification would take place accordingly. This may allow “Nadal beats Federer :)” to be classified differently from “Federer beats Nadal :)”.

Bigger Dataset The training dataset in the order of millions will cover a better range of twitter words and hence better unigram feature vector resulting in an overall improved model. This would vastly improve upon the existing classifier results.

Internationalization Currently, I focus only on English tweets but Twitter has a huge international audience. It should be possible to use my approach to classify sentiment in other languages with a language specific positive/negative keyword list.

8. CONCLUSION

Using a novel feature vector of weighted unigrams, I have shown that machine learning algorithms such as Naïve Bayes, Maximum Entropy and Support Vector Machines achieve competitive accuracy in classifying tweet sentiment.

9. ACKNOWLEDGMENTS

I would like to acknowledge “Laurent Luce” for his insightful introductory post [20] on classifying tweet sentiment which helped me immensely in implementing this project.

10. REFERENCES

[1] D. O. Computer, C. wei Hsu, C. chung Chang, and C. jen Lin. A practical guide to support vector classification chih-wei hsu, chih-chung chang, and chih-jen lin. Technical report, 2003.

[2] N. Cristianini and J. Shawe-Taylor. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, March 2000.

[3] B. J. Jansen, M. Zhang, K. Sobel, and A. Chowdury. Micro-blogging as online word of mouth branding. In CHI EA '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems, pages 3859–3864, New York, NY, USA, 2009. ACM.

[4] T. Joachims. Making large-scale support vector machine learning practical. In B. Scholkopf, C. J. C. Burges, and A. J. Smola, editors, Advances in kernel methods: support vector learning, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

[5] C. D. Manning and H. Schutze. Foundations of statistical natural language processing. MIT Press, 1999.

[6] G. Mishne. Experiments with mood classification in blog posts. In 1st Workshop on Stylistic Analysis Of Text For Information Access, 2005.

[7] K. Nigam, J. Lafferty, and A. Mccallum. Using maximum entropy for text classification. In IJCAI-99 Workshop on Machine Learning for Information Filtering, pages 61–67, 1999.

[8] B. Pang and L. Lee. Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval, 2(1-2):1–135, 2008.

[9] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 79–86, 2002.

[10] J. Read. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In Proceedings of ACL-05, 43nd Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2005.

[11] T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In Proceedings of Human Language Technologies Conference/Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005), Vancouver, CA, 2005.

[12] Alec Go, Richa Bhayani, Lei Huang. Twitter Sentiment Classification using Distant Supervision. Technical report, Stanford Digital Library Technologies Project, 2009.

[13] Publicly available twitter dataset - http://www.sananalytics.com/lab/twitter-sentiment/sanders-twitter-0.2.zip

[14] Twitter Sentiment Analyzer - http://66.212.17.122:9999/

[15] Stop words list - https://github.com/ravikiranj/coursework/blob/master/data-mining/data/feature_list/stopwords.txt

[16] Positive keyword list - https://github.com/ravikiranj/coursework/blob/master/data-mining/data/pos_mod.txt

[17] Negative keyword list - https://github.com/ravikiranj/coursework/blob/master/data-mining/data/pos_mod.txt

[18] Python Natural Language Toolkit - http://www.nltk.org/

[19] Libsvm - http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[20] Laurent Luce's introductory post on twitter sentiment classification - http://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/