Sentiment Analysis on Twitter Data set

Anvesh Athmakuri
anveshathmakuri@my.unt.edu

Ashik Shaik
ashikshaik@my.unt.edu

Abhishek Malyala
abhishekreddymalyala@my.unt.edu

Bharath Ramagoni
bharathramagoni@my.unt.edu

***Abstract*—Sentiment analysis is a process of identifying a person's emotion/attitude towards a topic. In this work, we develop models to automatically classify a tweet into positive or negative. We train different classifiers using a twitter data set and compare their performance. We use the tf-idf feature extraction in this project. An exploratory analysis of the data using Weka for other algorithms is also considered into account for comparison of performance. The classifiers compared are support vector machines, Naive Bayes, Decision Table, J48 over the same data set. This report describes the task, data and results.**

Keywords— Weka, SVM, Naïve Bayes, Decision Table, J48 tree classification, algorithms, term-frequency, preprocessing.

I. INTRODUCTION

Sentiment is simple to understand. It's just feeling or emotion, an attitude or opinion. On social media, the sentiment of a post can be seen in the tone or emotion conveyed of the brand mentioned. In social listening tools, sentiment analysis features will measure and report on the tone or sentiment of your social mentions. Sentiment analysis can help you to gauge opinion, which, in turn, can guide strategy and help decision making. Twitter, data can separate the positive tweets about your company, brand, product, or service from the negative and neutral tweets so you can see how well you are doing in the Twitter verse. In addition, it's important to note that not all communications can be classified as positive, negative, or neutral. Human language, feelings, and the way we communicate are just too complex for that. As a result, experts predict sentiment analytics soon will move beyond a simple positive/negative scale and expand into classifying a broader range of human emotions. And as sentiment analytics grows in its ability to accurately recognize a wider range of feelings and shades of meaning, organizations will become more comfortable with the idea of sentiment analytics and begin using it in new and even more exciting ways.

Sentiment analysis is an important research on understanding and learning public/user's opinion which helps in various verticals, such as:

- Determine marketing strategy,
- Improve campaign success,
- Improve product messaging,
- Improve customer service,
- Generate leads.

Social Media is a rich source of text that portrays a person's attitude/emotion/opinion. Text pre-processing is a major challenge in performing sentiment analysis of social media data. Huge amount of labeled data is also required to automate sentiment analysis. Previous work Jadav and Vaghela (2016) classifies social media text to positive and negative using Support Vector Machines and Naive Bayes. We follow a similar approach to pre-process tweets and train models to classify them into positive or negative tweets.

II. DATA

Sentiment analysis can be incredibly useful, and can help companies better answer pertinent questions and gain valuable business insights. Sentiment analysis technologies will continue to improve as they become more widely adopted. Different types of sentiment analysis use different strategies and techniques to identify the sentiments contained in a text. There are two main types of sentiment analysis: subjectivity/objectivity identification and feature/aspect-based sentiment analysis.

Subjectivity/Objectivity Identification: Subjectivity/objectivity identification entails classifying a sentence or a fragment of text into one of two categories: subjective or objectivity. However, it should be noted that there are challenges when it comes to conducting this type of analysis. The main challenge is that the meaning of the word or even a phrase is often contingent on its context.

Feature/Aspect-Based Identification: Feature/aspect identification allows for the determination of different opinions or sentiments (features) in relation to different aspects of an entity. Unlike subjectivity/objectivity identification, feature/aspect based identification allows for a much more nuanced overview of opinions and feelings.

Our Dataset:

The Twitter Sentiment Analysis Data set consisting of data from Sanders (2011) and UMICH SI650 - Sentiment Classification competition contains 1,578,627 classified tweets; each row is marked as 1 for positive sentiment and 0 for negative sentiment. We had huge amount of data available. In our experiments, we found that it is time taking to train any single classifier over the entire data. We hence took a subset of this data and reported results.

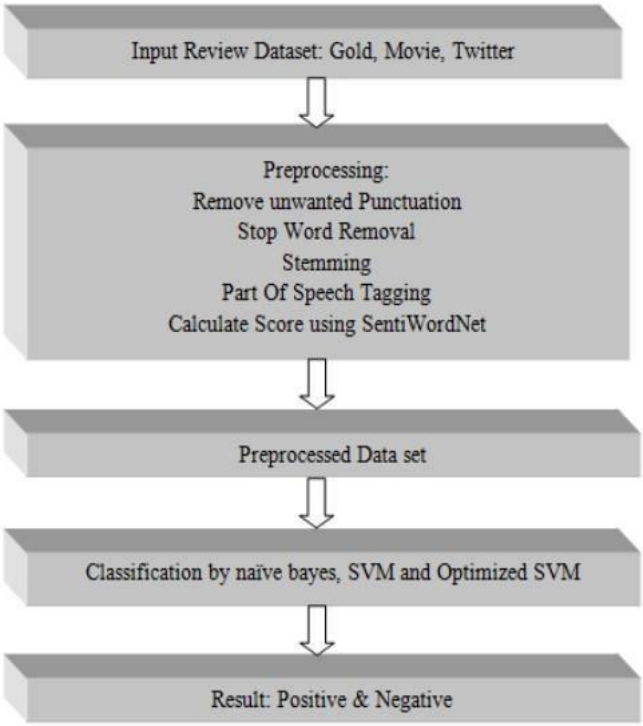


Figure 1: Method

III. Previous Work

The method followed in Jadav and Vaghela (2016) is illustrated in figure 1. We follow similar steps to pre-process tweets before training our models.

In our experiments, we do the following pre-processing:

1. Remove unwanted punctuation: All punctuation which are not necessary, it has been re- moved

2. Stop Word Removal: Some words used more and more time such words are called stop word. This pronoun, prepositions, conjunctions have no specific meaning.

3. Stemming: It converts word into its grammatical root form. Stemming technique converts word like “teach”,” teacher”,” teaching”,” teached”,” teaches” to root word teach. Among many available algorithms, we have used Porter stemming algorithm.

IV. Implementation

In this section, we describe the method used for feature extraction and how we trained different models.

A. Feature Extraction

We used tf–idf, short for term frequency–inverse document frequency, a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and text mining. The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

In a large text corpus, some words will be very present (e.g. “the”, “a”, “is” in English) hence carrying very little meaningful information about the actual contents of the document. If we were to feed the direct count data directly to a classifier those very frequent terms would shadow the frequencies of rarer yet more interesting terms. In order to re-weight the count features into float- ing point values suitable for usage by a classifier it is very common to use the tf–idf transform. Tf means term-frequency while tf–idf means term-frequency times inverse document-frequency:

$$tf - idf\,(t, d) = tf\,(t, d)\; idf\,(t).$$

Using the TfidfTransformer ‘s default settings,

$$Tfidf\ Transformer\ (norm = 'l2',\ use idf == True,\ smooth idf == True,\ sublinear\ tf == False)$$

the term frequency, the number of times a term occurs in each document, is multiplied with idf component, which is computed as

$$idf\,(t) = log(1 + nd\,(1 + df\,(d, t))) + 1$$

where n*d is* the total number of documents, and df(d,t) is the number of documents that contain term t. The resulting tf-idf vectors are then normalized by the Euclidean norm:

$$vnorm = v(||v||2) = (v(v12) + v22 + \ldots + vn) 2.$$

This was originally a term weighting scheme developed for information retrieval (as a ranking function for search engines results) that has also found good use in document classification and clustering.

B. Learning

In this project, we perform the following 3 experiments:

1. Experiment 1:

• We select a random 10000 tweets and split it into 6000 training tweets and 4000 tweets for testing

• We then train an SVM over the 6000 tweets. We optimize the SVM by tuning kernel, C and gamma parameter over 10-fold cross validation on training data.

• We also train a Multinomial Naive Bayes classifier with the 6000 training tweets.

• We finally report the precision, recall and F1-score over the 4000 test tweets.

2. Experiment 2:

• From above experiment we learned that Naive Bayes classifier is faster.

• We now split considered 50000 random tweets from the data and divided them into 30,000 train and 20,000 test tweets.

• We report the results of a Multinomial Naive Bayes classifier trained over the 30,000 tweets, and testing them on 20,000 test tweets.

We utilized Scikit learn Pedregosa et al. (2011) a python library to perform these experiments

3. Experiment 3:

• In this experiment, we perform two classifications with Decision Table and J48 tree on 15,000 randomly selected tweets on Weka Explorer.

• We select a random 15,000 tweets and use percentage split of 80% with 12,000 training tweets and 3000 tweets for testing.

We have converted the existing .csv dataset file to .arff format using python script and applied “StringToWordVector” filter.

V. Results and Discussion

Table 1 presents results with Experiment 1. The training of an SVM along with its optimization took over 1 minute when 6000 tweets are used for training. This was the case with both processed and un-processed tweets. Naive Bayes classifier was much faster and took only 0.004 seconds to train.

From Experiment 1, we realized training a Naive Bayes classifier is much faster. From the results:

• With Naive Bayes classifier, the overall performance with processed tweets is slightly better compared to unprocessed tweets

• However, training an SVM with unprocessed tweets yielded better results.

Table 2 Presents results with Experiment 2. The training of a Naive Bayes classifier took 0.018 seconds with 30,000 tweets and 0.0014 seconds to test on 20,000 tweets.

From Experiment 2 results:

• With Naive Bayes classifier, the overall performance with processed tweets is slightly lower compared to unprocessed tweets

• Increasing the training data evidently improved the performance of Naive Bayes classifier

Table 3 Presents results with Experiment 3 on Weka Explorer. The training of a Decision Table and J48 classifiers took 0.46 and 0.39 seconds respectively with 12,000 tweets and to test on 3000 tweets.

From Experiment 3 results:

• With Decision Table classifier, the overall performance with un-processed tweets is slightly better compared to J48.

• Increasing the training data alters the F-Measures exponentially.

A. Figures and Tables

1) Table 1: Results experiment 1: 10-fold cross validation over 6000 random training tweets and tested over 4000 random tweets from the entire corpus.

Model	class	Processed Tweets			Un-processed Tweets		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score
Optimized SVM	Positive	0.71	0.70	0.70	0.74	0.70	0.72
	Negative	0.80	0.81	0.80	0.80	0.84	0.82
	Weighted Avg.	0.76	0.76	0.76	0.78	0.78	0.78
Naive Bayes	Positive	0.82	0.46	0.59	0.85	0.42	0.57
	Negative	0.72	0.93	0.81	0.71	0.95	0.81
	Weighted Avg.	0.76	0.74	0.72	0.77	0.74	0.71

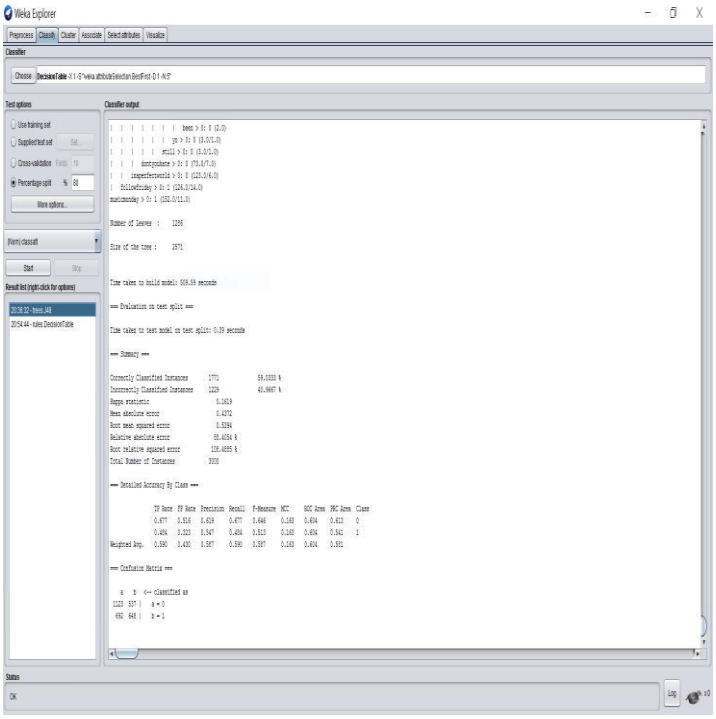
2) Table 2: Results Experiment 2: Naive Bayes trained over 30,000 random tweets and tested on 20,000 tweets.

Model	Class	Processed Tweets			Unprocessed Tweets		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score
Naive Bayes	Positive	0.74	0.79	0.77	0.75	0.80	0.78
	Negative	0.74	0.68	0.71	0.75	0.69	0.72
	Weighted Avg.	0.74	0.74	0.74	0.75	0.75	0.75

3) Table 3: Results Experiment 3: Decision Table and J48 trained over 12000 random tweets and tested on 3000 tweets keeping all the parameters constant on Weka Explorer.

Model	Class	Precision	Recall	F1-Score
Decision Table	Positive	0.616	0.804	0.697
	Negative	0.609	0.379	0.467
	Weighted Avg.	0.613	0.614	0.595
J48	Positive	0.619	0.677	0.646
	Negative	0.547	0.484	0.513
	Weighted Avg.	0.587	0.590	0.587

Weka Screens for Decision Table and J48 Tree:



VI. Conclusion

From the various experiments done in this project, we observed that sentiment analysis of tweets requires high amount of training data. It is computationally time taking to train any single classifier even with that available data. An optimized SVM performed with an F1 score of 0.78 when trained with 6000 unprocessed tweets.

We compared Naive Bayes and optimized SVM classifiers with respect to their performance and training time. We realized that Naive Bayes is faster to train but portrayed less performance compared to an optimized SVM.

We have selected random 15000 tweets from the dataset and ran in Weka Explorer with percentage split as 80% and classified using Decision Table and J48 classification algorithm. We have noticed that the F1 scores of Decision Table are better than J48 tree when trained with 3000 unprocessed tweets.

Finally, with our experiments we can conclude that with unprocessed tweets, SVM performance is better than the other mentioned classification algorithms and with processed tweets, Naïve Bayes classification is better at performance.

Appendix

The following will be submitted as part of reference for this report:

1. sentiment analysis.py: This python script takes as input the data file and number of tweets to output results with various classifiers

2. Data.csv: A comma separated value file that has tweets and sentiment denoted as positive and negative using 1 and 0 respectively

3. weka_train. arff: A comma separated value file that has 15000 randomly selected tweets from Data.csv. Experiments on Weka Explorer are conducted with this dataset.

4.Readme.txt: This file describes the requirements to execute the learning script

5. Experiment 1 and 2 result output files will also be submitted.

References

[1] Jadav, B. M. and V. B. Vaghela (2016). Sentiment analysis using support vector machine based on feature selection and semantic analysis. *International Journal of Computer Applica- tions 146*(13).

[2] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Pret- tenhofer, R. Weiss, V. Dubourg, et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research 12*(Oct), 2825–2830.

[3] Sanders, N. J. (2011). Sanders-twitter sentiment corpus. *Sanders Analytics LLC*.

[4] http://thinknook.com/twitter-sentiment-analysis-training-corpus-dataset-2012-09-22/

[5] http://scikit-learn.org/stable/modules/svm.html#classification

[6] http://scikit-learn.org/stable/modules/naive_bayes.html

[7] https://nlp.stanford.edu/sentiment/treebank.html

[8] http://scikit-learn.org/stable/modules/feature_extraction.html