Sentiment Classification of Reactions on Twitter by Keywords

**Arnav Kumar, Bajaj Sahil, Gopalakrishnan Gayathri,
Keerthana Kumar, Sabharwal Shahbaaz Deep Singh**

Abstract

With rapid advancement in the use of micro blogging, users, producers and consumers alike, are increasingly making use of social networking platforms to express their views on a variety of topics. Hence, social networking platforms like Twitter can be used as a reliable source to assess business and product trends. To address the above, the team aims to implement a sentiment analysis and feature extraction system to help users analyze common sentiments towards different topic trends. The system will also provide the ability to users to extract key descriptive phrases. The system makes use of a Twitter tweet corpus to train the parser and tagger models and set of manually classified positive and negative tweets for the application. The trained models were used to evaluate the components of the system and the results were found to be optimistic.

1 Introduction

As micro blogging is becoming increasingly popular, individuals are resorting to networking sites such as Twitter, and Facebook to publicly share their thoughts and ideas on wide range of topics. Such networking sites are used as a means to market products and applications in addition to being utilized as a source of information dissemination. Twitter is one such social networking and micro blogging service that enables users to post and read text messages limited to 140 characters called "tweets". Micro blogging has been identified as online word-of-mouth branding [2]. The nature of social networks has evolved over the past few years into a platform for individuals to express their sentiments on various products and issues, thereby allowing large organizations to assess business and product trends, and enabling the common man to make rational decisions on whether or not to buy a particular product, based on the general opinion on particular features. Relying on blogs and reviews alone is not sufficient, as they do not provide a large dataset of succinct views in real time from various locations. By taking these factors into account, the team aims to generate a list of positive and negative statements posted on Twitter over a large timespan, on any topic specified by the user, and extract key features on a particular product or event. There are two main objectives for this project:

1. Analysis of tweets related to a certain topic: A model will be built to classify tweets on a specific topic as positive or negative, using sentiment analysis and various Natural Language Processing techniques.
2. Feature Extraction: Key features or aspects of a topic (usually an event or product) will be identified from the parse trees of the tweets.

2 Crawling and Data Preparation

A tweet can contain words or phrases prefixed with a "#' called a hash tag, to assign a topic to the tweet. Similarly, users can reference other Twitter members in their tweets with a "@" followed by the username.

2.1 Crawler

The test bed used was a set of tweets collected from Tuesday, October 22 2013 21:52:24 to Thursday, October 24 2013 05:53:13. Apple Inc.'s event on October 22, 2013 for the release of the new iPad was used as a case study to analyze the results of the experiment and draw effective conclusions. The team utilized the Twitter streaming API to retrieve all English tweets during this period. Hence, the team was able to retrieve tweets on this topic, among others, which were used to verify the comprehensiveness of the application. 401290 tweets were extracted in total.

2.2 Data Description

Using the Twitter streaming API, the team extracted tweets, each stored in a separate XML file, which was named after its ID field. The xml file contains the following fields; Source (the URL to the tweet), Date (The timestamp for the tweet) and Text (the text of the tweet).

```
<doc id="159998">
 <source>http://twitter.com/321101195/sta
tus/392728979477901312</source>
```

```
 <date>Tue Oct 22 19:07:39 +0000
2013</date>
 <text>iPad air is good.</text>
</doc>
```

2.3 Data Preprocessing

The raw Twitter data was processed to extract the tweet text and the results were stored in a single plain text file nlp.merged.clean.normal containing 4,254,726 tokens. A subset of 1290 tweets was taken to serve as the training set for the system. The training data divided into 10 sample sets and stored as SampleSet1.txt to SampleSet10.txt under Untagged Training Data folder was used to train the system. The remaining 400000 tweets saved as TestingData.txt under Untagged_Testing_Set folder was used for testing.

3 Methods

3.1 Part Of Speech Tagger

The precision of general-purpose English POS taggers such as the Stanford POS tagger are not best suited for micro blogs as the content is short and noisy [4]. TwitIE, an information extraction system based on GATE algorithms and adapted specifically for micro blog content, contains an adapted Stanford tagger [3], trained on tweets tagged with Penn TreeBank (PTB) tag set as well as extra tag labels specifically for retweets, URLs, hash tags and user mentions [1]. This GATE Twitter POS tagger with its default gate-EN-twitter.model was used as the default model, upon which subsequent models were trained using the TwitIE tagger. The default model was applied to SampleSet1.txt of the training data and manually corrected. The corrected file was then used to train the next tagger model, which was then applied to SampleSet2.txt. This method was followed to boost the speed of manual correction.

```
Sample Tweet: Ipad Air. In Love!!
Tags given by default model: Ipad_NN
Air._NNP In_NNP Love!!_NN
Corrected Tags: Ipad_NNP Air._NNP In_IN
Love!!_NN
```

The final model trained with all the corrected data, finalpos.tagger under Training POS Models folder, was used to tag the test data. The final corrected POS tagging of the training data and the automatically tagged test data can be found under the POS Tagger folder in Tagged using Final Tagger and Tagged Testing Data folders respectively.

3.2 Syntactic Parser

Tweets often contain short and unstructured phrases, making it difficult for commonly used parsers such as Berkeley, Brown, Malt etc. to provide accurate parse trees based on defined conventions. The English PCFG caseless model of the Stanford parser was utilized to work out the grammatical structure of tweets, as this specific variation identifies proper nouns and ignores cases while parsing making it better for texts, tweets, and similar content. In addition, the dataset of tweets was normalized to eliminate or smooth Twitter specific jargon in order to further improve the resulting parse trees. For instance, various kinds of emoticons, ellipses, and user references in the beginning of tweets were eliminated. The english-PCFG.caseless.ser.gz model was used as the default model, upon which subsequent models were trained. This model was applied to SampleSet1.txt of the training data, and the resulting parse trees were manually corrected. This corrected set was then used to train the next parser model, which was then applied to SampleSet2.txt. The next sets were trained similarly by generating new models in the process. For instance, the first dataset contained the tweet *"IPad Air. In Love!!"* The parse tree given by the default case less model was,

```
(ROOT
 (NP
   (NP (NNP Ipad) (NNP Air.))
   (PP (IN In)
     (NP (NN Love!!)))))
```

"Ipad Air." and "In Love!!" were taken as fragments to generate the manually corrected tree as seen below.

```
(ROOT
 (FRAG
   (NP (NNP Ipad) (NNP Air.)))
 (FRAG
   (IN In) (NN Love!!)))
```

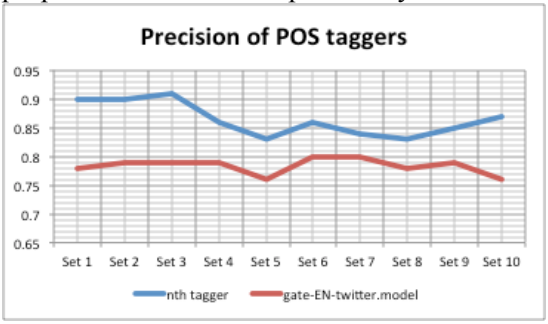
The final model SerializedModel10 in the serialized_models folder was used to generate parse trees for the test set. The final set of files can be found in the testing and training_for_parser folders.

4 Results

4.1 Part of Speech Tagger

Using the incremental retraining approach, it was found that the results of the tagger model used were largely dependent on how different the current dataset was from previous training sets. The

graph below depicts the precision of the intermediate tagger models as compared to that of the default model when applied to the corresponding datasets. Sources of errors were spelling mistakes and incorrect grammatical composition making it difficult for the model to correctly identify the appropriate word class for previously unseen words.



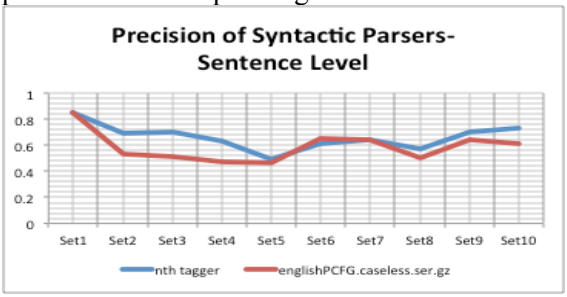
The precision, recall and F - measure of the default model and finalpos.tagger are as follows,

Model	Precision	Recall	F-Measure
gate-EN-twitter.model	0.784	0.784	0.784
finalpos.tagger	0.840	0.840	0.840

Table 1- Measures for the default and trained model for POS tagger

4.2 Syntactic Parser

Using the incremental retraining approach, it was observed that the precision of the intermediate models were largely dependent on the structure of the tweets in the dataset. The graph below depicts the precision of the intermediate parser models as compared to that of the default model when applied to the corresponding datasets.



Sources of errors were observed to be the lack of coherence and grammatical structure in tweet contents. The precision, recall and F - measure of the default model and SerializedModel10, at the sentence and phrase levels, are as follows:

Model	Precision	Recall	F-Measure
englishPCFG.caseless.ser.gz	0.590	0.612	0.601
SerializedModel10	0.596	0.665	0.629

Table 2- Measures for the default and trained models for Syntactic Parser at Sentence Level

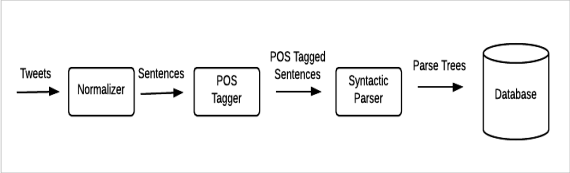
Model	Precision	Recall	F-Measure
englishPCFG.caseless.ser.gz	0.357	0.343	0.350
SerializedModel10	0.456	0.580	0.510

Table 3- Measures for the default and trained models for Syntactic Parser at Phrase Level

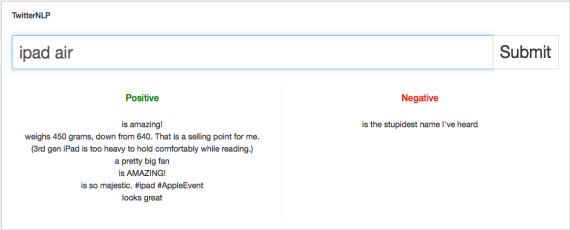
5 Application

The application lists the positive and negative descriptions of a topic on Twitter. The most precise descriptive phrase for the subject needs to be extracted before the sentiment can be determined which implies that the entire corpus can't be annotated with the sentiment for each sentence. For different queries different parts of the sentence might be considered. Hence, the sentiment analysis is the last step before the display of results to the user.

5.1 Data Pre-processing



5.2 Querying



5.3 Descriptive Phrase Extraction Algorithm

A recursive algorithm is used to find the approximate descriptive phrase for the query term. It is initially called on the node with the last matching search term. We first look at this node's siblings. If there are none, than we traverse higher in the tree and find siblings of the node's parent. There are additional checks for the types of the nodes examined. Simple heuristics are used to discard implausible nodes in the search. In case a suitable approximation is not found, the algorithm default's to returning the whole sentence. This assumption is pragmatic given the length constraint on tweets. According to some studies, the average number of sentences is 1.40 [5]. It is reasonable to assume the sentence will be related to the query term. The pseudo-code is as follows,

```
function getDescriptiveSubtree (Tree complete-
Tree, Tree matchedNode)
       if matchedNode is null or has no parent
nodes, return the whole tree.
       if the matchedNode has no siblings, re-
turn the descriptive subtree of its parent by
calling the function recursively with its par-
ent like so, getDescriptiveSub-
tree(completeTree, matchedNode.parent)
       for each sibling, if the sibling con-
tains select phrases that render it less im-
portant, then it skipped.
       if none of the siblings got selected,
the algorithm is run on the parent.
```

5.4 Sentiment Analysis of the Phrases

After all the descriptive phrases for the query have been found, this list is pruned to remove phrases with only a single word which is passed through the sentiment analyzer to get the approximate sentiment for each of the phrases. A Naive Bayes' Classifier instance is trained on a custom set of negative and positive sentences where every word of the incoming sentence is considered a feature. The set used comprised of some pre-existing training data for the classifier plus the manually classified sentences form our training set. We chose to use the words, rather than all bigrams + all words as training the classifier took significantly more time and the serialized model is much larger in size resulting in a slower classification at the cost of little increase in accuracy. Furthermore, the deployment on Heroku's free tier had processor limitations and the running time had to be contained accordingly. The comparative analysis of the two-classifier instances: Train set: 7996 instances, Test set: 2666 instances

Performance Metrics	Using all words as features	Using all words and bigrams as features
Accuracy	0.773	0.819
Positive Precision	0.787	0.825
Positive Recall	0.748	0.810
Negative Precision	0.760	0.813
Negative Recall	0.797	0.828

Table 4- Measures for the Sentiment Analyzer

6 Discussions and Conclusion

In conclusion, the main objective of the application to serve as a tool that provides a list of positive and negative aspects of a specific product/event over duration of time has been achieved. Tweets over a specific time period were retrieved using the Twitter API, and were divided to train and test the POS tagger and parser. Once the trained tagger and parser were ready, they were used to preprocess the dataset before it was inserted into the database. In the application, for each search query, all sentences with an occurrence of the query are pulled from the database. For each sentence, the descriptive phrase for the query term is found using the algorithm defined above. This list of descriptive phrases is then passed through the sentiment analyzer, which classifies them as either "positive" or "negative." These two sub lists are then displayed to the end user. Obtaining the descriptive phrases is not an exact science and often the phrases obtained with the algorithm were not the most descriptive phrase possible from the sentence, but the results were encouraging. The descriptive phrases' sentiment was evaluated with an average accuracy of 71%. Although the team was able to meet the defined objectives specified, there were some limitations. Some limitations and suggestions for further improvement are presented below,

1. Sarcasm Detection - People's tweets often express a positive sentiment at face level but aim to convey the opposite. Taking sarcasm into account would help make the application more solid and authentic.
2. Twitter-specific Jargon - The parsing of tweets can be improved by using a more diversified corpus, or by utilizing a parser that specifically deals with tweets and other micro blogs.
3. Language Detection - Restricting the dataset to only English tweets limits the scope of the application by a certain margin. Analyzing sentiments of tweets across languages would further improve the usefulness and value of the application.

References

[1] L. Derczynski, A. Ritter, S. Clarke, and K. Bontcheva. 2013. "Twitter Part-of-Speech Tagging for All: Overcoming Sparse and Noisy Data". In Proceedings of the International Conference on Recent Advances in Natural Language Processing, ACL.

[2] B. Jansen, M. Zhang, K. Sobel, A. Chowdury. The Commerical Impact of Social Mediating Technologies: Micro-blogging as Online Word-of-Mouth Branding, 2009.

[3] A. Pak, P. Paroubek. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. The International Conference on Language Resources and Evaluation. 2010.

[4] K. Bontcheva, L. Derczynski, A. Funk, M. Greenwood, D. Maynard, N. Aswani. TwitIE: An Open-Source Information Extraction Pipeline for Microblog Text. 2013.

[5] RT this: OUP Dictionary Team monitors Twitterer's tweets. 2009. http://blog.oup.com/2009/06/oxford-twitter/