Emotional Classification of Twitter Messages

Connor Riley, December 10, 2009

# Abstract

Using data gathered from the social networking site Twitter, I developed a Naïve Bayes classifier to identify a given message’s sentiment. Messages were gathered from Twitter’s public feed and hand-tagged as positive, negative or neutral. Messages were analyzed for both unigram and bigram features, as well as features specific to the syntax of Twitter messages. This classifier was able to attain significant accuracy for a representative subset of Twitter data, suggesting future directions for its application and improvement.

# Introduction

Twitter is a relatively simple system in which users can post length-limited messages which are then visible to their friends or to the world at large. The popularity of Twitter, as well as its accessible API, have made it a rich source of data for research and custom applications.

I chose to use Twitter data to develop a simple sentiment classifier, which would indicate whether a given message was positive, negative, or neutral in tone. I envisioned the following applications for such a classifier:

* Warn Twitter users who are considering whether or not to follow a user. A lot of spam-bots will follow one’s Twitter account over time, and it would be nice to have them automatically marked appropriately.
* Analyze emotional tendencies over time. For instance, I have friends who constantly post enraged messages which can reflect negatively on them; I’d like to know what my emotional classification is.
* Warn users about the emotional import of a message they are writing. Simply having an indicator of how a message is going to appear to others may discourage overly dramatic or unnecessarily hasty, angry posting.

Additionally, an accurate sentiment classifier can be useful as a tool in a more complex system, such as spam detection or other projects like the Twitter Stock Predictor.

My overarching goal in the development of the classifier was to make it as accurate as possible for as representative a subset of Twitter messages as I could gather.

# Related work

Analyzing the emotional content of texts is a popular research topic in NLP. The research of Strapparava and Mihalcea was of particular interest to me, as they used a variety of NLTK tools to both classify texts using a Naïve Bayes classifier trained on LiveJournal (as LiveJournal posts are tagged by the author with the author’s mood), and by using a specially developed extension to the WordNet lexicon called Wordnet-Affect.

Another project which I referenced was done by two students at MIT’s CSAIL laboratories to perform sentiment analysis on movie review comments. This project used NLTK tools and corpora to classify comments from the social news site Digg using both Bayesian and combinatorial models for classification. This project influenced some of my methods in building a classifier, which will be further noted.

Some text analysis applications have been developed for Twitter itself, although the relative recency of Twitter’s popularity means that little research has been done specifically on Twitter usage and output. One paper by Kim and Gilbert analyzed Twitter data on celebrity deaths from a quantitative sociological standpoint. This paper informed me of the Affective Norms for English Words (ANEW) wordlist, which the paper confirms as a useful tool for sentiment analysis of Twitter data. ANEW provides a set of normative emotional ratings for a large number of words; unfortunately access to the ANEW dataset for research purposes takes up to several weeks to confirm, so I was not able to gain access to it for this project in time to use it.

# Data and Features

Using Python’s interface to the Twitter API, I obtained a large corpus of about 20,000 Twitter messages. The majority of these messages were obtained from Twitter’s public feed, which contains messages from a wide variety of users. This included a considerable amount of messages not written in English, and also spam messages advertising product links. I decided to not include messages in either of these categories in the final data set to simplify classification to as few categories as possible. While it certainly seems possible to use similar NLP techniques to identify Twitter spam or non-English messages, I chose to maximize accuracy over a smaller dataset.

Knowing in addition that the messages in the public feed were of variable quality in terms of grammar and spelling, I additionally downloaded a smaller set of messages from trusted Twitter users whose feeds I have access to. This set was smaller due to the Python-Twitter API’s limit on how many messages can be accessed at once, but was also narrowed much less by the constraint to use only English, non-spam messages.

The content of messages were saved without including any personally identifiable metadata which is accessible through the Twitter API. I did not want to make emotional classification contingent upon the author’s identity, especially as some of the messages used in the classifier are not publically available.

I tagged a subset of 4,000 messages by hand, using three tags (positive, negative, neutral). After tagging, the proportion of messages was as follows:

|  |  |
| --- | --- |
| **Tag** | **Proportion** |
| positive | 26.6% |
| negative | 22.7% |
| neutral | 50.7% |

The final classifier uses the following features:

**Words (unigram tagging):** The standard “bag-of-words” feature for Bayesian classification models.

**Word pairs (bigram tagging):** Incorporating word pairs was important for capturing elements like emoticons, since punctuation marks were isolated by the unigram tagger.

**Special features**: Twitter messages can contain certain features like hashtags (words preceded by # to indicate subject), replies (usernames preceded by @ to indicate a reply to that user), or links to external sites.

I recognize when a message contains one of these features and separate out the relevant information: what site is being linked, who the reply is to, what the hashtag is. This information is used as a feature, while the original string is removed from the message and replaced with a placeholder. This maintains the meaningfulness of messages like “I’m at a party with @someone”, since “with @” is a feature with a positive tendency; if “@someone” were removed completely the useful bigram would be removed as well.

**Stopword elimination**: I used the list of stop words found here (<http://www.ranks.nl/resources/stopwords.html>) to limit their effect on classification. This inclusion was inspired by the CSAIL project referenced earlier.

The following features were not used in the final classifier:

**Message length and capitalization:** I implemented this feature to look for any correlation between length, capitalization and emotional content. Both features were measured by the quartile into which the message fell. When implemented, the classifier did indicate possible usefulness of combining length and capitalization into one feature, especially among short messages, but the accuracy of the classifier suffered, as demonstrated in the Results section. This drop in accuracy could easily indicate overfitting for the training data which was not generalizable to the test data. Additionally, if no words in a message strongly indicate emotional content, allowing the classifier to use a weak length-emotion correlation to classify the message seems likely to result in inaccuracies.

**Complex emoticons:** Emoticons range widely beyond :( and :). One component of the errors I noticed were more complex emoticons consisting of a mix of letters, numbers and punctuation. I chose initially to add delimiters to separate character strings from punctuation rather than relying on the existing spaces as delimiters; one consequence of this was breaking up complex emoticons. I chose not to implement this feature, as there is very little regular structure in complex emoticons that can be looked for programmatically; providing an emoticon dictionary could be an alternative solution.

# Models and Results

## Model

The model which provided the greatest accuracy was a simple Naïve Bayes classifier using primarily a ‘bag-of-words’ feature extractor described above, and results reported are for that classifier.

I did test the usefulness of a Maximum Entropy classifier. However, training and making changes to such a classifier takes a long time, and I was not able to achieve results that significantly outperformed the baseline value for success of approximately 50%.

In the end, the Naïve Bayes classifier provided superior results quickly, which was important in allowing me to run multiple tests to get an accurate average value for the results.

## Results

The following table demonstrates the performance of the classifier using different combinations of features:

|  |  |  |
| --- | --- | --- |
| Model | Accuracy (avg. 20 trials) | Standard Deviation |
| Bag-of-words only | 72.35% | 8.81 |
| BOW, special features, stopwords | 74.73% | 8.97 |
| BOW, special features, stopwords, length, capitalization | 72.02% | 9.03 |

The classifier consistently outperforms the baseline value for success—50.7% given the proportion of neutral messages in the data set.

In presenting the three different feature combinations, the tradeoffs in adding features on top of unigram and bigram tagging become clear: adding features for a variety of special features (hashtags, etc.) causes a small increase in accuracy, but adding features for message length and capitalization decreases accuracy, as these features are biased by the training data.

With or without special features, accuracy scores range from 60%-93%; this variability can be attributed to the breadth of words and phrases in the training data. The classifier must contend with the full complexity of the English language as well as the common idiosyncracies of unedited content. In a random sample of messages to classify, there may easily be a large number of words the classifier hasn’t encountered before, either because they are unusual, misspelled, or serendipitously not in the training set.

## Error Characteristics

The vast majority of errors in the classifier are positive messages misclassified as negative, and vice-versa. Many of these are ambiguous in that they contain negative language (a curse word in a positive message is almost sure to cause misclassification of that message). Some contain emoticons which largely determine the sentiment of the message but which are complex strings of characters and punctuation not caught by the feature extractor. Inspirational quotes and song lyrics are also prone to errors, as they contain more ambiguous, complex language than the average message.

## Threats to Validity

1. Size of corpus—even at 4,000 messages, the corpus I used only begins to scratch the surface of the dialogues that take place on Twitter every day. By not limiting the messages in the corpora to those focusing on a single topic, for instance, the sheer variety of language encountered means that a very large corpora is necessary to achieve maximum accuracy.
2. Hand-tagged data—as I was the only tagger, there was little to no error-checking or possibility of resolving differences of opinion regarding which tags were correct. Indeed, many of the errors the classifier currently encounters are cases where I would reconsider the correctness of the tag used.
3. Sample validity—by limiting myself to a subset of data sources (public feed and hand-chosen people), I may have biased the sample. For instance, as the public feed is time-based, sampling it during an event (an episode of Saturday Night Live, for example), may push the #SNL hashtag to have an unfairly positive score.

# Conclusion

I set out to create a classifier to indicate the sentiment of Twitter messages based on a corpus of messages taken from Twitter data feeds. Using a Naïve Bayes classifier and bag-of-words feature extractor optimized for Twitter messages, relatively high accuracy was obtained over a representative subset of all Twitter messages. The classifier is thus fairly successful in the goal of tagging Twitter accurately in conditions as realistic as possible.

## Future Directions

1. Investigate the use of data sets like Wordnet-affect and ANEW which accurately represent word sentiment.
2. Investigate means of hand-tagging by multiple individuals which converge on a more accurate tagging system.
3. Expand system of tagging to capture emotions within the ‘Positive’ and ‘Negative’ subsets
4. Release classifier to classify messages in personal feeds and collect tags from Twitter users.

# References

[Carlo Strapparava , Rada Mihalcea. “Learning to identify emotions in text.” Proceedings of the 2008 ACM symposium on Applied computing, March 16-20, 2008, Fortaleza, Ceara, Brazil.](http://portal.acm.org/citation.cfm?id=1364052&dl=7&coll=GUIDE&CFID=69483601&CFTOKEN=39824501) <http://www.cse.unt.edu/~rada/papers/strapparava.acm08.pdf>

Yessenov, Kuat, and Sasa Misailovic. “Sentiment Analysis of Movie Review Comments” Massachusetts Institute of Technology, Spring 2009. <http://people.csail.mit.edu/kuat/courses/6.863/report.pdf>

### Kim, Elsa and Sam Gilbert. “Detecting Sadness in 140 Characters: Sentiment Analysis and Mourning Michael Jackson on Twitter” Web Ecology Project, August 2009. <http://www.webecologyproject.org/2009/08/detecting-sadness-in-140-characters/>