Sentiment analysis of users' reviews and comments

Abhijit Chakankar   Sanjukta Pal Mathur   Krishna Venuturimilli

1 Overview

The goal of our project is to apply machine learning for sentiment analysis, or opinion mining, on user-generated text on the web, such as movie or product reviews, or comments on social networks and forums. Given the content of this user-generated text, we are looking to classify the reviews/comments as being positive or negative. An opinion is defined as a positive or negative sentiment, view, attitude, emotion, or appraisal about an entity or an aspect of the entity from an opinion holder. This is a relevant problem in today's world as the amount of user-generated text on the web is increasing and sentiment analysis can be used to detect the mood of users on a forum or to detect spam if the text is too negative. By building features to categorize the content of a given text, we use supervised learning techniques to detect positive vs negative sentiment in the text.

2 Datasets

We analyzed three separate datasets, two of which comprised of movie reviews and the third involved detecting insults in users' comments.

2.1 Large Movie Review Dataset: This dataset contains movie reviews from IMDB, consisting of 25k highly polar movie reviews for training, and 25k set for testing. In each set, there is a 50-50 breakdown between positive and negative reviews: No more than 30 reviews are allowed for any given movie since reviews for the same movie could have correlated ratings. The train and test sets contain a disjoint set of movies, so no significant performance is obtained by memorizing movie-unique terms. The "Learning Word Vectors for Sentiment Analysis" paper by Maas et al. that also uses this dataset achieves a highest accuracy of 88.89%.

2.2 Polarity Movie Review Dataset: This dataset consists of 2000 processed movie reviews drawn from IMDB archive, classified into positive and negative sets, each set comprising 1000 movie reviews. We ran 10 fold cross validation to measure accuracy of classifiers.

2.3 Detecting Insults in Social Commentary: This competition was to build the best model to detect when a comment from a conversation would be considered insulting to another participant in the conversation. Samples could be drawn from conversation streams like news commenting sites, magazine comments, message boards, blogs, text messages, etc. The training data set has 3947 examples, out of which 1049 (26.6%) are insults. The test data set has 2467 examples, out of which 693 (26.2%) are insults. Since this was a relatively skewed data set, we also made use of precision and recall to evaluate our models.

3 Features

For each of the datasets, some preprocessing had to be done before applying the below-mentioned features. Examples include parsing relevant text body from the original data set, removing html tags from the text, handling quotes and other punctuation appropriately, and removing redundant repetition of letters (e.g. 'stuuuuupid' to 'stuupid', so as to defer from the original 'stupid').

3.1 N-grams: Using the bag of words approach, we tried uni, bi and trigrams. For the Kaggle dataset, we also tried using up to 4-grams upon observation of common phrases. We also tried stemming to only use stems of the words in the n-grams. In particular, we made use of the Porter stemmer from the NLTK libraries. We experimented with using a variety of values as the feature values, such as the term frequency count, just {0, 1} for whether the term was absent or present, and the tf-idf weight.

3.2 Part-of-speech (POS) tagging: We fed each piece of text from the dataset to NLTK POS tagger. We appended the POS tag to the end of the n-gram stems to distinguish between different uses within a sentence of each word. Since sentiments are often expressed with the use of adjectives and adverbs, we also tried using just the adjectives and adverbs as features.

3.3 Custom features

3.3.1 Large Movie Review Dataset: With movie reviews often comprising of a paragraph of text, users typically make their main expression of sentiment at the very start or end of the review. The middle half of the paragraph typically contains other details about the movie, which the user may end up summarizing as a positive or negative sentiment at the end. Thus, we tried incorporating the position of where the n-gram appeared with respect to the particular example text. We added a suffix to the word for which quarter of the text the word appeared in, like the Pang and Lee paper describes. Another option could have been to specifically target words that appeared in the first or last sentence as features.

3.3.2 Polarity Movie Review Dataset: With POS tagging, we tried variants of bigrams and trigrams where the adjective/adverb (including superlatives) is
a) the second word of bigrams ('not funny', 'nothing great', 'does little', 'nothing new' etc)
b) the first word of bigrams ('extremely well', 'really good', 'compelling story' etc)
c) the middle word of trigrams

3.3.3 Kaggle "Detecting Insults" Competition Dataset: We downloaded a subjectivity clues file and a badwords file containing 301 bad words that was used to identify really bad (often curse) words. We tried using words that express strong positive or negative subjectivity. This file contains 8221 entries, one per line, as follows: "type=strongsubj len=1 word1=abhors pos1=adj stemmed1=n priorpolarity=negative".

Other features used include:

1. *You*: A comment is usually an insult only if the insult is directed to the second person.
    a. Using the Google bad words file from the web, as described above.
    b. Using the subjectivity clues file from the web, as described above.
    c. Detecting bad word within a few words around variations of "you" (such as "u", "you're", "your")
    d. Generate "context words", i.e. the words between "You" and bad word.
    e. Detecting a strong negative subjectivity word within a few words around "you".
2. *badwords*: We tried raw counts as well as fraction of badwords. This feature did not work as the non-insults had a higher proportion of badwords than insults in training set.
3. *badword_pairs* : We used ordered pairs of badwords (not necessarily bigrams) in the same sentence that had a high chi-squared score as features. e.g. it found that dumb + ass is not nearly as bad as dumb + f***.
4. *Exclamations, Uppercase words*: We use fraction of these in the comment to get a small improvement.
5. *Part of Speech Tagging*: Using nltk's part of speech tagging we tried following features:
    a. Fraction of each tag type (nouns, verbs, adjectives etc)
    b. Fraction of only Adjectives, Adverbs or both

3.4 Feature selection: One mode of feature selection that was attempted across all the datasets was to use the top N-most frequently-occurring n-grams as the only features. Additionally, when using the NLTK Naive Bayes classifier, we picked out the N most informative features used for training the data to rebuild the model for application against the test set. For the Kaggle dataset, we also tried chi-squared based feature selection. We sorted the features (e.g. *badword_pairs*) on their chi-squared scores and used only the ones that exceeded a threshold (typically 10.0). This helped reduce overfitting and made some of the features useful.

4 Models

Naive Bayes and Support Vector Machines (SVM) were mainly used to classify the dual-classed sentiment of the data.

4.1 Naive Bayes Classifier: We used the Naive Bayes Classifier model from the Python NLTK libraries. We used the absence or presence of a term as the feature values after experimenting with the other measures discussed in 3.1. We also ran k-fold cross validation across the various data sets to assess whether the data was being overfitted to the training data set, typically using k = 10.

4.2 SVM: We ran the SVM model from the Python libsvm libraries. We experimented with the types of kernels and penalty values and ended up mostly using the default parameters. e.g. For the large movie review dataset, the only difference was that we used a higher penalty C=4 when training the SVM model.

4.3 Logistic Regression: We used a custom implementation of Logistic Regression in Python using Gradient Descent. The optimization step typically ran for about 50 to 100 iterations, and used the same feature set as for NB and SVM. We also used k fold CV with various values of k (3, 5, 10, 20) and the results are being reported for k=5

5 Results

Listed below are the accuracy results we got with each of the constructed models against each dataset.

5.1 Large Movie Review Dataset: All the models made use of the Porter stemmer on the tokens.

Model #	Feature	Naive Bayes	SVM
1	Unigram	0.83296	0.84852
2	Unigram with adjectives only	0.78744	0.76736
3	Unigram_quartile_position	0.82624	-
4	Unigram_POS	0.83632	0.85792
5	Model 1 + bigram	0.86538	**0.89632**
6	Model 3 + bigram_quartile_position	0.8636	-
7	Model 4 + bigram_POS	**0.86936**	0.89404

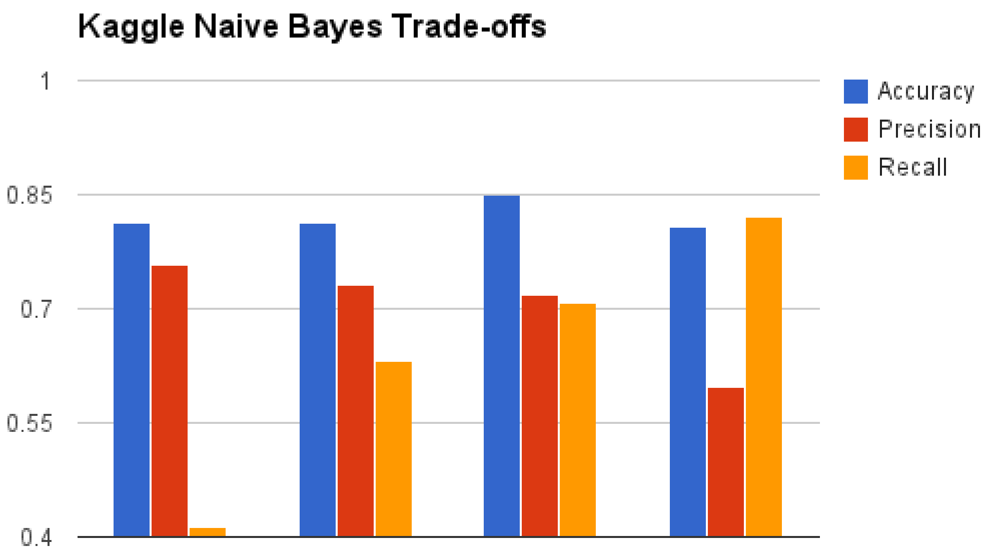
5.2 Polarity Movie Review Dataset: Part-of-speech tags were used to select the particular n-grams.

Model #	Feature	Naive Bayes	SVM
1	Unigrams (absence/presence)	0.83	0.86
2	Unigrams with frequency count	0.8	-
3	Unigrams (only adj/adv)	**0.85**	0.86
4	Bigrams (absence/presence)	0.8	**0.89**
5	Bigrams (first word adj/adv)	0.83	0.83
6	Bigrams (second word adj/adv)	0.77	0.77
7	Trigrams (absence/presence)	0.79	0.77
8	Trigrams (middle word adj/adv)	0.75	0.77

5.3 Kaggle “Detecting Insults” Competition Dataset: Used the bad + subjective word files with n-grams.

Model #	Feature	Naive Bayes	Logistic Regression	SVM
1	Unigrams only	0.798	0.802	-
2	You_<bad_word> only	0.827	0.829	-
3	Model 1 + Model 2	0.847	0.843	0.839
4	Model 3 + bigrams	0.839	0.837	-
5	Model 4 + up to 4-grams	0.838	0.836	-
6	Model 3 + bad word pairs	0.848	0.843	-
7	Model 6 + exclamations	0.849	**0.845**	-
8	Model 7 + no stemming	0.843	0.837	-
9	Model 7 + neg subjective words	**0.851**	0.846	**0.8415**

Furthermore, we did an analysis of precision and recall as well for this dataset since it was known to be skewed with respect to having more non-insults than insults in both training and test sets.



Accuracy, Precision, Recall trade-offs using different models

6 Analysis

6.1 Large Movie Review Dataset: The combination of unigrams and bigrams with the part-of-speech tag appended as a suffix to each word performed best on Naive Bayes. While using the adjectives by themselves did not perform well, appending the part-of-speech tag to each token helped distinguish between the different uses of words, especially in the construction of bigrams to express sentiment. Examples of the most informative features used with this setup included:

* negative: “`worst_JJS|film_NN`”, “`thi_DT|crap_NN`”, “`worst_JJS|movi_NN`”, “`wast_VB|your_PRP`”, “`just_RB|aw_JJ`"
* positive: “`highli_RB|recommend_VBD`”, “`wonder_JJ|movi_NN`”, “`perfectli_RB|cast_VBN`”, “`well_RB|worth_IN`”

On the other hand, the standard unigrams and bigrams worked best on SVM, without any use of the part-of-speech tags. The quartile position of each n-gram within the text did not provide any further improvements.

6.2 Polarity Movie Review Dataset: Unigrams (absence/presence of adj/adv) performed best on NaiveBayes and Bigrams (absence/presence) performed best on SVM. In unigrams (using the top N most frequently used words among all documents), adjectives such as outstanding(+), astounding(+), atrocious(-), ludicrous(-), thematic(+), insulting(-) etc., came up as the most informative features. Trigrams degraded the performance and using frequency counts did not show any improvement. To push the accuracy further, experimenting with the feature selection techniques here is not enough. Advanced NLP techniques are necessary to extract the subject of the sentences in the reviews and their meanings. For instance, the author might describe what was expected out of the movie in positive terms, but in the last sentence indicate that he was disappointed, or vice versa.

6.3 Kaggle “Detecting Insults” Competition Dataset: Many features were dedicated to detecting the second person subject “you”, since a comment was usually labeled as an insult only if it can be established that the insult is directed to the second person. Trying to learn the contexts in which "you … <bad word>" is not an insult did not work out well due to the sparsity of the Kaggle data set. With a bigger corpus that strategy may succeed.

The use of the bad words and subjective words files proved useful to seek out the bad words in detecting an insult. Detecting a bad word within some words around variations of "you" (such as "You are an idiot" being an insult) worked more often than not, but fell short on several occasions, such as "You are not an idiot" or "You think he is idiot" not being labeled as insults.

Using the additional features such as part-of-speech tags did not provide any gains, and some combinations actually saw a minor loss in performance. Additionally, we think there are several mis-labeled examples that end up confusing the training model. Here are a few of examples we think that are incorrectly labeled as "not insults": “f*** all u p**** asz crackers” is an obvious insult, while “You are as naive as a teen-age girl” is more subtle.

7 Conclusion

We were able to achieve comparable results as the corresponding papers for both the movie review datasets using our described feature sets. For the Kaggle dataset, while we didn’t have a baseline performance to compare to, in comparing against the 73% skewed data set itself, we were able to build a model that improved upon that. The bag of words approach using the n-grams worked well for sentiment analysis, where the inclusion of both unigrams and bigrams made the most improvements. Incorporating the part-of-speech tag of the n-grams also boosted our models. All the algorithms we tried (NB, LR and SVM) worked reasonably well, with SVM performing best on the movie review datasets.

Overall, gaining further improvements through these models for the purposes of sentiment analysis falls more into the domain of Natural Language Processing. We should do some sentence structure analysis and associate positivity/negativity to the subject of the sentiment -- e.g. the second person “You” in the Kaggle data set, or the actual movie in the case of the movie review data sets. If we build features using more information about the semantic structure of the sentence, these supervised models will probably perform better. It is also to be noted that sentences involving sarcasm, negations and others such as ‘this movie is worth watching several times’ cannot be classified by our above-described feature set.

Lastly, we dabbled a little into trying to detect a specific type of sentiment, namely insults, via the Kaggle data set. Going beyond the more general positive vs negative sentiment analysis of the movie review data sets, we applied more custom features specific to detecting bad words and detecting the second person “You”. This work can be extended to further detect specific types of sentiment, as well as experiment with different domains (e.g. politics, sports) other than movie reviews to determine whether there are some common types of features that can be used for sentiment analysis or if there are sentiment-specific (e.g. bad words for insults) or domain-specific features (e.g. political party jargon) that ultimately provide further improvements.

8 References

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. Proceedings of EMNLP, pp. 79--86.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

Bo Pang and Lillian Lee. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. Proceedings of the ACL, pages 271–278.

8.1 Data Sources

1 Large Movie Review Dataset: http://ai.stanford.edu/~amaas/data/sentiment/
2 Polarity Movie Review Dataset: http://www.cs.cornell.edu/people/pabo/movie-review-data/
3 Kaggle “Detecting Insults” Competition Dataset:
http://www.kaggle.com/c/detecting-insults-in-social-commentary