SWASH: A Naive Bayes Classifier for Tweet Sentiment Identification

Ruth Talbot[1], **Chloe Acheampong**[2] **and Richard Wicentowski**[1]
[1]Swarthmore College, Swarthmore, PA USA
[2]Ashesi University, Accra, Ghana
{rtalbot1, richardw}@cs.swarthmore.edu
chloeacheampong@gmail.com

Abstract

This paper describes a sentiment classification system designed for SemEval-2015, Task 10, Subtask B. The system employs a constrained, supervised text categorization approach. Firstly, since thorough preprocessing of tweet data was shown to be effective in previous SemEval sentiment classification tasks, various preprocessessing steps were introduced to enhance the quality of lexical information. Secondly, a Naive Bayes classifier is used to detect tweet sentiment. The classifier is trained only on the training data provided by the task organizers. The system makes use of external human-generated lists of positive and negative words at several steps throughout classification. The system produced an overall F-score of 59.26 on the official test set.

1 Introduction

Over the past few years, an increasing number of people have begun to express their opinion through social networks and microblogging services. Twitter, as one of the most popular of these social networks, has become a major platform for social communication, allowing its users to send and read short messages called 'tweets'. Tweets have become important in a variety of tasks, including the prediction of election results (O'Connor et al., 2010). The emergence of online expressions of opinion has attracted interest in sentiment analysis of tweets in both academia and industry. Sentiment analysis, also known as opinion mining, focuses on computational treatments of sentiments (emotions, attitudes, opinions) in natural language text. In this paper we describe our submission to Task 10, subtask B: Message Polarity Classification. The task is defined as: 'Given a message, classify whether the message is of positive, negative, or neutral sentiment. For a message conveying both a positive and negative sentiment, whichever is the stronger sentiment should be chosen' (Rosenthal et al., 2015).

This paper describes a system which utilizes a Naive Bayes classifier to determine the sentiment of tweets. This paper describes the resources used, the system details, including preprocessing steps taken, feature extraction and classifier implemented, and the test runs and end results.

2 Resources

2.1 Labeled Tweets

This system is constrained, and the only training data used is the sentiment labeled training data provided by the task organizers. The training data we used includes 8142 tweets, each labeled as positive, negative or neutral.

2.2 Sentiment Lexicon

Our system relies on an external lexicon of approximately 6800 tokens labeled as either positive or negative (Liu et al., 2005). The lexicon consists of words that humans have tagged as having either strongly negative or strongly positive sentiment. If a word in a tweet is preidentified as highly positive or negative, we add a special feature to the tweet's features to indicate that the tweet included a highly positive word or a highly negative word (Kiritchenko et al., 2014). Although multiple lexicons exist, e.g.

Preprocessing Step	F1 score change
jazzy	-5.67
stopwords	$-.56$
negation	1.74
username normalization	0.34
url normalization	0.40
overriding	1.74
lowercasing	2.21
tokenization	4.00
sentiment lexicon	5.81

Table 1: Changes in F1-score obtained by each preprocessing step (taken individually, not cumulatively) using 5-fold cross validation on the provided training set.

(Wilson et al., 2005) and (Mohammad et al., 2013), we were unable to include them due to time constraints.

3 System Details

The system consists of several preprocessing steps, feature extraction, a Naive Bayes classifier and a secondary classifier that makes use of tokens that are strongly correlating with either a positive or negative sentiment. Improvements that were attempted but were unsuccessful in improving the system are also described.

3.1 Preprocessing Steps

3.1.1 Tokenization

All tweets are tokenized using Twokenizer, a Twitter-specific tokenizer (Owoputi et al., 2013). The tokenizer can detect and handle conditions unlikely to occur in more formal writing, such as mentions, hashtags and retweet tokens.

3.1.2 Normalization

During preprocessing, all tweets are normalized. This included several steps:

- Lowercasing all words (e.g. ‘Hello’ to ‘hello’ or ‘heRe’ to ‘here’)
- Converting all URLs (identified as strings containing ‘.com’, ‘http’, ‘www’ and ‘.co’) to the string ‘URL’
- Converting all mentions (identified as strings beginning with ‘@’) to ‘username’

3.1.3 Negation

The system implements a basic version of negation to improve the accuracy of the classifier. When processing the data, any words in between a negative adverb or verb, a ‘negation key’ (e.g. never, not, can’t) and the next end of sentence indicator, in this case, any punctuation symbol, are negated. Negation was implemented to avoid misclassification of tweets due to a word of one sentiment following a negation key and therefore being of the opposite sentiment. For instance, a sentence could state: “That movie was not the best thing I’ve ever seen.” Clearly, this sentence is negative, but without negation, the presence of the word ‘best,’ a typically positive word, might lead this tweet to be classified as positive, not negative. If however, a tag is added (in this case ‘NOT_’) to any words following a negation key, those words will be more likely to be classified appropriately, as ‘NOT_best’ will more often be seen in negative contexts (Kiritchenko et al., 2014).

3.1.4 Other Preprocessing Considered

Several other preprocessing steps were considered. In particular, a spell corrector, Jazzy[1], was used as it had previously been shown to be effective (Miura et al., 2014). This step was taken to reduce dimensionality and provide better matches with the sentiment lexicon, e.g. converting ‘luve’ into ‘love’, so instead of seeing ‘love’ once in a positive context and ‘luve’ once in a positive context, we would see ‘love’ twice in a positive context, giving it more weight as a positive feature and finding a match in the sentiment lexicon. However, Jazzy actually reduced accuracy and F-score of our system. One potential explanation for this finding is that tweets may contain significant amounts of abbreviations, slang and misspellings that are too far removed from the original spelling for a spell checker to identify and adjust to its correct spelling.

Additionally, removing stopwords was attempted. A list of the 25 most common words in the English language was acquired using the Brown Corpus. This list provided the system with common words unlikely to be strongly associated with any sense. These words were then removed before feature selection. In our final implementation of the

[1]http://jazzy.sourceforge.net/

Sentiment	Precision	Recall	F-score
Negative	61.72	50.45	55.52
Positive	73.98	66.17	69.86
Neutral	64.09	76.21	69.63
F1 score:	62.69	Accuracy:	67.42

Table 2: F-scores for individual sentiments and overall score, produced using 5-fold cross validation on SemEval-2015 training data.

classifier, removing stopwords has a small negative effect on performance.

3.2 Feature Extraction

The features used in our classifier are unigrams, negated unigrams, and two special tags indicating the presence or absence of words in the tweet being found in the sentiment lexicon. During preprocessing, negated unigrams are created by prepending 'NOT_' to a unigram if it follows a negation key, described above. If the unigram follows a negation key, only the negated unigram, not its original form, is included as a feature. In addition, a 'positive' or 'negative' feature (represented by 'POSW' or 'NEGW') is added for each positive or negative word a tweet contained, as identified by inclusion in the sentiment lexicon.

3.3 Classifiers

3.3.1 Naive Bayes Classifier

We used a Naive Bayes classifier to classify the tweets. Naive Bayes relies on the assumption of conditional independence among the features, something that is clearly not true here. While Naive Bayes classifiers manage to perform well despite this assumption, a classifier not reliant on this assumption might outperform a Naive Bayes classifier (Gamallo and Garcia, 2014).

The Naive Bayes classifier employed Laplace smoothing. More advanced smoothing techniques were attempted, but actually reduced both the accuracy and F1 score of the system. The additive smoothing constant was empirically chosen to be 0.4. The Naive Bayes classifier was trained solely on the training data from SemEval-2015.

3.3.2 Other Classifiers Attempted

In addition to Naive Bayes, several other classifiers were tried, and an attempt was made to employ a combination of multiple classifiers to predict sentiment. These classifiers included a typical decision list (which defaults to most frequent sense classification), and a number of classifiers included in scikit-learn (Pedregosa et al., 2011): LinearSVC, GaussianNB, NearestCentroid, MultinomialNB, and BernoulliNB. Each classifier used the same preprocessing and feature selection employed by the Naive Bayes classifier. However, after implementing all of these classifiers and attempting to use a combination of their sense decisions to make a more accurate prediction, none of the classifiers, nor any combination of their decisions, outperformed the Naive Bayes classifier, and therefore none were used in our submission.

3.3.3 Post-Processing

Several features were identified that, when present, were strongly indicative of a positive or negative sense (e.g. ':)', ':(', 'awful', 'love'). If one of those features was present in a tweet, a rule-based system overrode the decision of the Naive Bayes classifier, labeling the tweet as either positive or negative. This step was conducted after negation so that no unnegated words would be used to classify a tweet incorrectly. Suprisingly, this 'overriding' step improved our F1 score by several points, indicating that there are several features that when present are strongly indicative of a tweet's sense.

These strongly positive or negative overriding features were determined by inspection of training data and using our own knowledge to come up with symbols and words which were highly polar in sentiment. The positive word list contained 4 emoticons[2] and 7 overly positive words: 'love', 'great', 'happy', 'wonderful', 'good', 'perfect', and 'beautiful'. The negative list contained 6 emoticons[3], 4 curse words and 5 negative words: 'fuck', 'shit', 'ass', 'crap', 'hate', 'awful', 'stupid', 'horrible', and 'ugh'. Future work could include automatically inducing such a list from training data.

[2]Positive :) :D :-) ;)

[3]Negative :(:-(:/ :'(:.(>:(

Laplace λ	**F-score**	**sentiment weight**	**F-score**
.3	62.40	1	59.25
.4	**62.69**	**5**	**62.69**
.5	62.39	6	62.39

Table 3: Two parameters empirically determined using crossvalidation. In Laplace smoothing, λ is the additive constant for unknown words. The 'positive' and 'negative' features introduced by the sentiment lexicon were given five times the weight of the token unigrams.

Dataset	**Rank**	**F1**
Twitter 2015	21	59.26
Live Journal 2014	24	69.43
SMS 2013	34	56.49
Twitter 2013	27	63.07
Twitter 2014	31	62.93
Twitter 2014 Sarcasm	15	48.42

Table 4: Performance on the official 2015 test data as well as on the progress data sets.

4 Test Runs

In addition to attempting additional classifiers, several parameter values were experimented with using 5-fold cross validation to determine which produced the best F-scores.

4.1 Parameter Selection

As mentioned earlier, the constant used for additive Laplace smoothing was determined empirically. Values between 0 and 1 were tested, and it was determined that the ideal value was 0.4. Table 3 shows the change in score for 3 different values close to 0.4.

The second parameter tuned empirically was the weight given to the 'positive' and 'negative' features added if a tweet contained a positive or negative word listed in the sentiment lexicon. After experimenting with various ways of oversampling this feature, we determined that giving these words five times the weight of other unigrams was the optimal number under crossvalidation. (see Table 3).

5 Conclusion

This paper describes the implementation of a sentiment classification system that uses extensive preprocessing and a Naive Bayes sentiment classifier. Using only a Naive Bayes classifier the system achieved a 59.26 F1 score, placing 21st out of 40 overall in Task 10, subtask B. Interestingly, our system overperformed in the sarcasm progress data set, requiring some further investigation.

While our attempt at weighting the decision of multiple classifiers was unsuccessful, we believe this was due to using the same features for each classifier, and that these features may have been overfitted to those found effective in a Naive Bayes classifier.

Additionally, our human-generated list of positive and negative words and symbols, whose presence automatically overrode the classifier's decision, should be further explored. It is highly likely that more words and symbols exist whose presence is highly indicative of a negative or positive tweet sentiment. Automatic creation of these lists would likely improve performance and be more experimentally justified.

References

Pablo Gamallo and Marcos Garcia. 2014. Citius: A naive-bayes strategy for sentiment analysis on english tweets. In *Proceedings of SemEval-2014*, pages 171–175.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014. NRC-Canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of SemEval-2014*, pages 437–442.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: analyzing and comparing opinions on the Web. In *Proceedings of WWW'05*, pages 342–351.

Yasuhide Miura, Shigeyuki Sakaki, Keigo Hattori, and Tomoko Ohkuma. 2014. TeamX: A sentiment analyzer with enhanced lexicon mapping and weighting scheme for unbalanced data. In *Proceedings of SemEval-2014*, pages 628–632.

Saif Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of SemEval-2013*, pages 321–327.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan Routledge, and Noah Smith. 2010. From tweets to polls: Linking test sentiment to public opinion time series. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*, pages 122–129.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL-2013*, pages 380–390.

Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. Semeval-2015 task 10: Sentiment analysis in Twitter. In *Proceedings of SemEval-2015*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings HLT '05*, pages 347–354.