Swarthmore_NLP: Sentiment Detection of Tweets

Aly Valliani
Swarthmore College
500 College Avenue
Swarthmore, PA 19081, USA
avallia1@swarthmore.edu

Richard Liang
Swarthmore College
500 College Avenue
Swarthmore, PA 19081, USA
wliang1@swarthmore.edu

Abstract

This paper describes our participation in SemEval-2015 Task 10 Subtask B. We experiment across a wide range of machine learners and techniques to maximize performance. The final proposed sentiment classifier utilizes logistic regression and a diverse set of preprocessing and feature extraction techniques. Our system generates official accuracies of 64.21 and 63.15 on experimental test sets, which are competitive with those generated by systems submitted as part of SemEval-2015 Task 9 Subtask B.

1 Introduction

The explosion of social media has generated a new means for tracking public perception across a diverse set of issues in today's world. Twitter data, for example, has shown to be an accurate and immediate determiner of consumer confidence and political opinion (O'Connor et al., 2010). However, the processing of informal text poses a unique challenge to text processing systems due to the prevalence of slang, emoticons, typos, abbreviations and various other unconventional artifacts. As such, there has been an increasing interest in recent years on natural language technologies that can process such forms of text (Pang et al., 2002).

Last year multiple systems competed in the 2014 edition of the SemEval Twitter Sentiment Analysis Subtask B aimed at classifying tweet polarity as positive, negative or neutral (Rosenthal et al., 2014). Each system employed a variety of pre-processing and feature extraction techniques. In this paper we describe a supervised sentiment analyzer as part of SemEval-2015 Task 10 Subtask B that aims to incorporate and improve upon the techniques employed by its predecessors.

This paper is organized as follows. Section 2 describes resources such as labeled data and lexicons that were obtained for use within our system. Section 3 explains various system details such as preprocessing techniques, feature extractors and machine learners. Sections 4 and 5 indicate results from our evaluation and avenues for future work, respectively.

2 Resources

2.1 Training Data

Our training data consists of 8111 tweets and their given sentiments (either positive, negative, or neutral). For every tweet, the training data provides the information in the following tab-separated format:

```
TWEET_ID TWEET_ID_2 CONTENTS SENTIMENT
```

The concatenation of the tweet IDs are used for the unique identification of the individual tweets. Tweets and label information were given to us by the organizers of this task for the purpose of building an effective tweet sentiment analysis classifier. The tweets given to us were in their original, unaltered forms, meaning that spelling and grammatical errors may exist. We also use five-fold cross validation on this data set in order to test the effectiveness of our machine learning model.

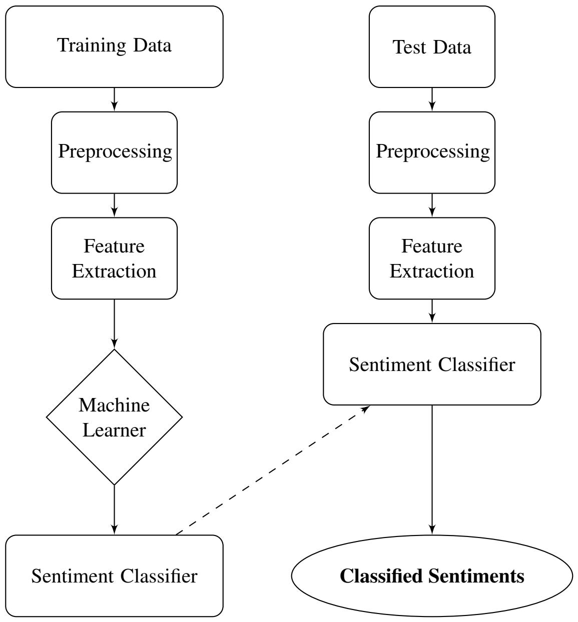

Figure 1: Our Classifier Workflow

2.2 Development Data

The development data consists of the same information as the training data, and provides another data set of 1390 tweets to test our trained machine learning model against. As with the the training data, the tweets in the development data are unaltered and may consist of various spelling and grammatical errors.

2.3 Lexicons

We explored and utilized many lexicons for our classification system. Bing Lius Opinion Lexicon[1], which contains words and their positive/negative association is used for feature extraction. We also adapted an English slang dictionary (slang → meaning) from Wikipedia[2]. A similar emoticon dictionary (emoticon → emotion) was also adapted[3]. We also used the Sentiment-140 lexicon (Mohammad et al., 2013) and MPQA (Wilson et al., 2005) for experimental purposes.

3 System Details

The system consists of a variety of pre-processing computations, feature extractors, and a machine learner. We give a high-level overview of our components in Figure 1.

[1]http://www.cs.uic.edu/ liub/FBS/sentiment-analysis.html
[2]http://en.wiktionary.org/wiki/Appendix:English_internet_slang
[3]http://en.wikipedia.org/wiki/List_of_emoticons

3.1 Preprocessing

Preprocessing both our training and test data allows us to normalize our text into a single canonical form. We provide the elements incorporated in our processing scheme below.

3.1.1 Case Folding

We lower-case all of the alphabetical characters in our datasets. For example, "I like to watch WWE at 1:00 AM!" becomes "i like to watch wwe at 1:00 am!"

3.1.2 Arktweet Tokenization

The Arktweet tokenizer provided by CMU[4] was utilized in order to split large words into more descriptive components. For example, "tonight…the" may be split into "tonight, …, the."

3.1.3 URL Shortening

We employ URL shortening/conflating, meaning that any word in a tweet containing an obvious URL determiner (we used "http", "://", and "www." as determiners) is modified to be "_URŁ". For example "check out www.wwe.com!" would be modified to "check out _URL_"

3.1.4 Username Shortening/Conflating

Username shortening/conflating is also used - we simply modify any word in a tweet starting with '@" into "_AT_USER". Therefore, "@bob, you are the best!" becomes "_AT_USER, you are the best!".

3.1.5 Hashtag removal

Words starting with hashtags had their preceeding tag removed. For example, "Johnny had a great game! #Success! became "Johnny had a great game! Success!".

3.1.6 Emoticon Dictionary Replacement

Wikipedia contained a very useful emoticon resource which associated emoticons with certain emotions. The result was a dictionary-like lexicon consisting of emoticon → emotion label pairings. We replace every emoticon found in our dictionary with its emotion label. This also means that several

[4]http://www.ark.cs.cmu.edu/TweetNLP/

Table 1: Experiment Results (Using Sem-Eval Scoring and Five-Fold Cross Validation)

	Features	LogisticRegression	LinearSVC	SGDClassifier
(1)	Unigrams	60.72	58.16	56.34
(2)	Unigrams + BingLiu	**64.21**	60.80	60.18
(3)	Bigrams	42.85	42.48	45.64
(4)	Bigrams + BingLiu	56.80	54.9	54.71
(5)	Bigrams + Unigrams	57.41	57.13	55.91
(6)	Bigrams + Unigrams + BingLiu	62.9	61.25	60.32

emoticons which actually represent the same emotion will be associated with one common label. We provide an example below:
"this is fun :)" becomes "this is fun _HAPPY_" and "i like turtles :]" becomes "i like turtles _HAPPY_"

3.1.7 Abbreviations Dictionary Replacement

As with above, Wikipedia was consulted in order to retrieve information on common abbreviations and their associated word or phrase. We also create a similar dictionary-like lexicon consisting of abbreviation $\rightarrow$ associated word or phrase pairings. An example replacement would look this this:
"lol, that was so good!" becomes "laugh out loud, that was so good!"

3.1.8 Negation

We employ negation on our words in the following manner: when we encounter a "not" in our sentence, we simply prepend "_NOT_" to the subsequent words until a punctuation mark is reached. An example of this process is shown below:
"i will not become an elephant!" becomes "i will not _NOT_become _NOT_an _NOT_elephant!

3.1.9 Porter Stemming

We use nltk's built-in porter stemmer (Porter, 1980) on the words in our datasets. Porter stemming allows us to reduce certain words to their stems. For example, "running" becomes "run" and "sized" becomes "size."

3.2 Feature Extraction

Unigram features (all the words that make up the sentence) make up the bulk of our feature set. We also employ Bing Liu's opinion lexicon to generate positive and negative features based on how may of the labeled words exist in our sentence. These features utilize labels "_POS_" and "_NEG_" respectively. For example, if Bing Liu labeled "2-faced" and "abrasive" as negative, we would add 2 "_NEG_"s to our feature set.

3.3 Machine Learning

In order to ease implementation of the sci-kit[5] learner, we employed the SklearnClassifier wrapper provided by nltk. Many different classifiers were tried, including support vector machines (LinearSVC), stochastic gradient descent (SGDClassifier), logistic regression (LogisticRegression), ensemble classifiers (AdaBoost), and decision trees. In the end, we found that the classifier which performed the best was the logistic regression classifier. For the feature set parameter, we simply include a histogram of the features. We also make our term-document matrix more dense by subclassing nltk.classify.scikitlearn and overriding its train() and classify_many() methods. The resulting subclass was named SklearndDenseClassifier. Originally, this was done so we could experiment on the AdaBoosting and DecisionTree classifiers, but we later found that making our dataset denser aided in logistic regression's performance as well. Our final configuration for the machine learning model is as follows:

```
SklearnDenseClassifier
(LogisticRegression,
class_weight = 'auto',
dual = True')
```

3.4 Discarded Ideas

Throughout the course of our experimentation, we fiddled with many pre-processing ideas. Ultimately, there were some that we decided were detrimental to our final system. These included spelling

[5]http://scikit-learn.org/stable/

Table 2: Experiment Results Using Optimal Configuration

Test Set	Accuracy	F1 Negative	F1 Positive	F1 Neutral	Official Score
Training (5-fold cross validation)	71.66	55.79	72.63	75.58	64.21
Development Set	67.70	57.84	68.46	71.17	63.15

correction using Jazzy[6], removing all instances of punctuation, utilizing higher order n-grams, removing stop words, using lemmatization and using the the Sentiment-140 lexicon. In addition, we tested the MPQA sentiment lexicon, which neither helped nor hurt our accuracy but substantially increased the time for pre-processing, so it was not included in our final implementation.

4 Results

As indicated earlier in section 3.3, we utilized many different classifiers for experimentation. Support vector machines (LinearSVC), stochastic gradient descent (SGDClassifier) and logistic regression (LogisticRegression) were our top-performing classifiers. On these classifiers, we performed combinations of easily implementable feature extractions and determined logistic regression to be our best classifier as indicated in Table 1.

We evaluated our logistic regression classifier on the training data using five-fold cross validation and on the development data. We report our results using the following evaluation metrics:

- **Accuracy**, which is the percentage of correctly classified tweets.
- **F1 score** across all three classes.
- **Official SemEval Score**, which is the average F1 score across positive and negative classes.

Table 2 shows our results when using the logistic regression classifier.

5 Conclusion

In this paper we presented a twitter classification algorithm for SemEval-2015 Task 10 Subtask B that utilizes logistic regression and multiple pre-processing computations as well as feature extractors. We obtained official accuracies of 64.21 and 63.15 on our training (using five-fold cross validation) and development data, respectively.

The scores produced by our classifier are competitive with those garnered by systems submitted as part of SemEval-2014 Task 10 Subtask B. The task of discriminating between negative and neutral tweets remains to be a major a limitation of our approach. As such, future work will emphasize techniques to either effectively discriminate between negative and neutral tweets or utilize an ensemble of classifiers each specialized in the detection of specific sentiment class labels. In future works, we also hope to utilize semi-supervised learning methods aimed at increasing training instances to maximize feature extraction.

Acknowledgments

We would like to thank Professor Richard Wicentowski for all his assistance, suggestions, and insights during the course of our project.

References

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets.

Brendan O'Connor, Ramnath Balasubramanyan, and Bryan R. Routledge. 2010. From tweets to polls: Linking text sentiment to public opinion time series.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumps up? sentiment classification using machine learning techniques.

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Sara Rosenthal, Alan Ritter, Preslav Nakov, and Veselin Stoyanov. 2014. Semeval-2014 task 9: Sentiment analysis in twitter. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 73–80, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis.

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