Team Super-Win: A Logistic Regression Approach to Tweet Sentiment Classification

Michael Superdock
Swarthmore College
500 College Avenue
Swarthmore, PA 19081, USA
msuperd1@swarthmore.edu

Winnie Ngo
Swarthmore College
500 College Avenue
Swarthmore, PA 19081, USA
wngo1@swarthmore.edu

Abstract

This paper describes our system for addressing SemEval-2015 Task 10 Subtask B: the message polarity classification task. For this task, participants must design a system that can identify whether a tweet is positive, negative, or neutral in sentiment. We use a machine learning approach for this task, specifically using the *scikit-learn* implementation of a logistic regression classifier to classify our tweets. We implement a variety of methods for preprocessing and feature extraction in order to best train this classifier. To evaluate the quality of our model, we evaluate our system using the training data provded for subtask B[1] by 5-fold cross validation. Our final classifier makes predictions with over 70% acccuracy, with a SemEval score of 58.74.

1 Introduction

Twitter is among today's most popular social networks. It has over 284 monthly active users, and on average 500 million Tweets are sent every day[2]. From global news to personal anecdotes, it is one of the fastest ways that various information is transmitted. Accordingly, the wealth of information found in tweets may afford us important insight into widespread sentiments as they arise. Understanding these sentiments may prove invaluable for numerous groups and individuals, such as news networks, big businesses, political figures, and the like.

[1]http://alt.qcri.org/semeval2015/task10/
[2]https://about.twitter.com/company

In this paper we describe our approach to the sentiment analysis of tweets. We undertake the the challenge of classifying the sentiment of the entire message, as is formalized through SemEval 2015 task 10 subtask B, which states, Given a message, decide whether it is of positive, negative, or neutral sentiment. For messages conveying both positive and negative sentiment, the stronger one is to be chosen (Rosenthal et al., 2014). This is a particularly difficult task, as sentiment analysis is typically applied to larger documents. Tweets are limited to 140 characters and often contain rather informal language. Abbreviations, emoticons, and hashtags are commonplace.

Machine learning has been a widely popular approach to sentiment analysis tasks such as this one (Alexandre Denis and Bellalem, 2014) Following this trend, we use a logistic regression classifier for our sentiment classification system. This is the same machine learning approach used by TEAM X and TUGAS, who are among the most successful groups that attempted the same Task in SemEval-2014 (Amir et al., 2014; Miura et al., 2014). We came to select this specific machine learning approach due to its ability to outperform various other approaches.

Along with carefully selecting a machine learning algorithms, we focus our efforts on developing a variety of preprocessing and feature extraction techniques. Our features range from the identification of simple unigrams to the identification of emoticon polarity. Through feature engineering we are able to iteratively tune our system to make it more effective for message-level classification.

In the following sections we walk through our approach in detail. In Section 2 we describe our system, providing an in depth explanation of our feature extraction and preprocessing techniques. In Section 3 we discuss our experiments and evaluate the results of our classification system. In Secion 4 we explain some of the future directions of our work. And in Section 5 we provide some concluding remarks to highlight the utility of our sentiment classification system.

2 System Description

We now describe our approach to the SemEval-2015 Task 10 Subtask B.

2.1 Preprocessing

Before we extract a majority of the features from our tweets, we use a number of preprocessing procedures in order to revise them. This preprocessing step allows us to eliminate some of the data within our tweets that are incomplete or noisy, ultimately enabling us to identify the most salient features for determining the sentiments of a tweet (Hemalatha et al., 2012). Our preprocessing procedure consists of the following steps.

Tokenization and Tagging:

We use ark-tweet-nlp (version 0.3.2) from the CMU ARK group, for tokenization and part-of-speech tagging[3]. Through ark-tweet-nlp we develop a set of the words, along with the corresponding part-of-speech tags, for every tweet. The CMU ARK group reports that this tagger identifies parts-of-speech with approximately 90.4% accuracy footnotehttps://github.com/brendano/ark-tweet-nlp/blob/master/data/ritter/AccuracyEvaluation.txt. It also captures some of the more informal parts-of-speech common amongst tweets, such as emoticons.

Word Removal:

We further preprocess our tweets by removing some of the tokens that are less useful for determining the tweets overall sentiment. The two types of tokens that we remove completely from our data set are URLs and Twitter Handles. Neither holds much, if any, semantic meaning. We remove them because they do not provide useful information that might inform our sentiment classification.

Hashtag Symbol Removal:

We remove the symbol # that is found before hashtags. Hashtags often illustrate a salient point that the tweets author is trying to get across. We remove the hashtag because its removal allows us to make stronger inferences based on the token(s) that the hashtag precedes.

Case Folding:

We use case-folding so that no differentiation is made when the same token is differently capitalized. We do this because the identity of the word itself tends to be a more salient feature then the identity of various versions of the same word with slightly different capitalization. All capitalization-based features are identified before this preprocessing step. We expand on these capitalization features, along with other features, in the following section.

2.2 Features

We identify features from our tweet dataset both before and after prepocessing is complete.

2.2.1 Features Before Preprocessing

Repeated Characters:

We identify tokens that contain a contiguous set of three or more identical characters, and we modify these tokens to use them as features. Here we describe the modification process briefly. First, we remove all extra characters in our token, along with any attached punctuation. Then we use case folding, just as we would in preprocessing, to convert our token to strictly lowercase letters. Lastly, we attach the suffix _REPETITION to indicate that the token originally consisted of repeated characters, but that we have modified it for our purposes. This process allows us to convert tokens such as Niiiiiiiiiiccccceee!!, and other variations of it, to the feature nice_REPETITION.

Capitalized Words:

We identify tokens longer than 3 characters that contain strictly uppercase letters. We impose a minimum token length of 4 characters so that abbreviations, such as NYC, are not included in this feature

[3]http://mvnrepository.com/artifact/edu.cmu.cs/ark-tweet-nlp/0.3.2

set. We generate a count of these capitalized words in each tweet.

2.2.2 Features After Preprocessing

Token n-grams:

We employ word unigrams as features. We find that higher order word n-grams and discontiguous word n-grams adversely affect classification accuracy, so we do not include them in our feature set.

POS n-grams:

We use both POS unigrams and bigrams as features. We note specifically that the identification of POS unigrams allows us to generate relative counts of certain relevant token types, such as emoticons, adjectives, and punctuation. Beyond POS bigrams, we find that higher order POS n-grams and discontiguous POS n-grams adversely affect classification accuracy. We do not include higher order POS n-grams in our feature set.

Negated Words:

We modify tokens under the semantic scope of a negation using an algorithm proposed by Christopher Potts in his Sentiment Symposium Tutorial[4]. We consider something to be within a negative scope if it is found after a negating word (i.e. not, no, never, and words ending in nt) and before the next punctuation mark. To these tokens within a negative scope, we add the suffix _NOT.

Emoticon Polarity:

We classify emoticons as being either positive or negative in sentiment. We give no classifications to ambiguous emoticons. We generate a count of positive emoticons and a count of negative emoticons in each tweet.

2.3 Machine Learning

2.3.1 Initial Work

We first developed a number of systems using a variety of machine learning classification methods. We implemented three different classifiers using naive Bayes, Support Vector Machines (SVM) and logistic regression. We iteratively compared the results of each of these three machine learning approaches in order to determine the optimal solution for the message-level SemEval task.

We found that our system performed reasonably well using Naive Bayes, in tandem with a variety of weighted features. Using similar features, SVMs achieved even higher accuracies. However, we ultimately used a logistic regression implementation to develop our system. This machine learning classification method consistently outperformed the previously mentioned methods by at least 2%. We describe the underlying details of our logistic regression algorithm implementation below.

2.3.2 Logistic Regression

Logistic regression is a statistical method for classification. For our system, we are using multinomial logistic regression (as opposed to a binomial logistic regression) because we are aiming to select the best of three classes. It does this by minimizing the cost function that we describe below.

Given a set of m feature-label pairs (x_i, y_i) with $i = 1, ..., m$, $x_i \epsilon \Re^n$ and $y_i \epsilon \{-1, +1\}$, modeling the classifier involves solving the following optimization equation, where $C > 0$ is a penalty parameter.

$$\min_{w,c} \parallel w \parallel_1 \sum_{i=1}^{m} log(1 + e^{-y_i w^\theta x_i})$$

To implement this as our classifier, we used *scikit-learn*[5] which is a machine learning library for Python. A wrapper is used over the implemtentation available in the *liblinear*[6] package.

For a multiclass classification problem like ours, *scikit-learn* uses the one-vs-rest strategy. This method minimizes the cost function shown above by building one classifier per class. It then trains each of these classifiesrs to distinguish the samples in one class from the samples in all remaining classes.

3 Experimental Setup and Results

3.1 Experimental Set-up

In order to evaluate the quality of our system we run our classification system on set of the training data for SemEval-2015 Task 10 Subtask B. This set of data consists of 8111 tweets, of which 3019 have

[4]http://sentiment.christopherpotts.net/

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

[6]httpL//csie.ntu.edu.tw/-cjlin/liblinear

	Precision	Recall	F-measure
Negative	39.21	56.64	46.34
Neutral	80.41	70.19	74.96
Positive	68.73	73.71	71.13

Table 1: Measures of precision, recall, and F-measure across all 3 tweet classifications.

positive sentiments, 1186 have negative sentiments and 3906 have a neutral sentiments. We classify this set of data using 5-fold cross validation. For each of the three possible classifications, we measure precision, recall, and F-measure. Results for our experimental classifications are shown in Table 1.

3.2 Evaluation Metrics

From the results of our classification on this data set, we derive three different quantitative measures by which we can evaluate our results: Accuracy, Average F-measure, and SemEval score.

- Accuracy: The percentage of correct classifications
- F-measure Average: The average of the positive, negative, and neutral F-measures, giving equal weighting to each class
- SemEval Score: The average of the positive and negative F-measures, giving equal weighting to each class.

3.3 System Evaluation

Given the aforementioned evaluation metrics, our system performed with an accuracy of 70.04%, a traditional F-measure average of 64.14%, and a SemEval score of 58.74. We use the SemEval score as the primary metric by which we validate our system. We make changes to our system in an effort to optimize this score specifically.

4 Conclusion

We have described our approach to the message-level tweet classification task outlined in SemEval-2015 Task 10 Subtask B. We utilize a supervised machine learning approach with a diversified set of features specific to the informal vocabulary of tweets. With respect to our machine learning algorithm, we find that logistic regression performs the best out of the algorithms we tested, and so is used in our final system. With respect to our features, we find that having a wide variety of features, especially those that capture tweet-specific information, are necessary for the development of a successful message-level classifier.

With our classification model, we found that we could classify tweets with a decent accuracy of 70.04% and a SemEval score of 58.74. We prove that we can identify the polarity of tweet sentiment decently well, although our system is not competitive with the best classifiers. Some future directions that we are considering for our system include (1) incorporating new preprocessing procedures, such as spell checking, (2) introducing clustering or some form of semi-supervised learning, and (3) adjusting our weighting for various features.

Acknowledgments

We would like to thank both the SemEval organizers for providing the tweet data for this project and Richard Wicentowski for his assistance throughout this process.

References

Nadia Bellalem Alexandre Denis, Samuel Cruz-Lara and Lotfi Bellalem. 2014. Synalp-empathic: A valence shifting hybrid system for sentiment analysis. pages 605–609, August.

Silvio Amir, Miguel B. Almeida, Bruno Martins, Joo Filgueiras, and Mario J. Silva. 2014. Tugas: Exploiting unlabelled data for twitter sentiment analysis. pages 673–677, August.

I. Hemalatha, G.P. Saradhi Varma, and A. Govardhan. 2012. Preprocessing the informal text for efficient sentiment analysis. *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, 1:58–61.

Yasuhide Miura, Shigeyuki Sakaki, Keigo Hattori, and Tomoko Ohkuma. 2014. Teamx: A sentiment analyzer with enhanced lexicon mapping and weighting scheme for unbalanced data. pages 628–632, August.

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.