Sentiment Analysis Classification for Rotten Tomatoes Phrases on Kaggle

Kevin Hung
kehung@ucsd.edu

ABSTRACT

In the second assignment for CSE 190: Data Mining and Predictive Analytics, we apply some techniques to improve the accuracy of classifying Rotten Tomatoes phrase sentiments.

General Terms

Algorithms, Experimentation

Keywords

Classification, Sentiment Analysis, Opinion Mining, Naïve Bayes, Binned, Regression

1. INTRODUCTION

Applying sentiment analysis on reviews based on text features is distinct from rating-scale inference problems like predicting the rating value on a review (e.g. movies, restaurants, etc) because we can gain more details and insight on the human component (e.g. opinions, emotions, feelings) than with numerical features. As Richard Hamming once famously stated, "The purpose of computing is insight, not numbers." One of the main and many applications of classifying the sentiment of Rotten Tomatoes phrases through automation and machine learning is to save the human effort of evaluating each phrase manually.

2. DATASET

The original Rotten Tomatoes sentences were gathered as described in Pang and Lee's (2005) [1] approach to sentiment classification using metric learning, using 10,662 review snippets which were usually a sentence long. Then Socher et al. from Stanford NLP refined the snippet data into a more fine-grained form of pharsed phrases and used Amazon Mechanical Turk to outsource the manual task of interfacing and annotating the sentiments of the phrases [2].

For the version of the data we obtained from the Kaggle website [3], a tab delimited file containing around 156,060 training records with only the phrase's original sentence id and the actual phrase as the features and the sentiment value as the label.

The second file for testing contains 66,292 records with only the sentence id and the phrase values provided.

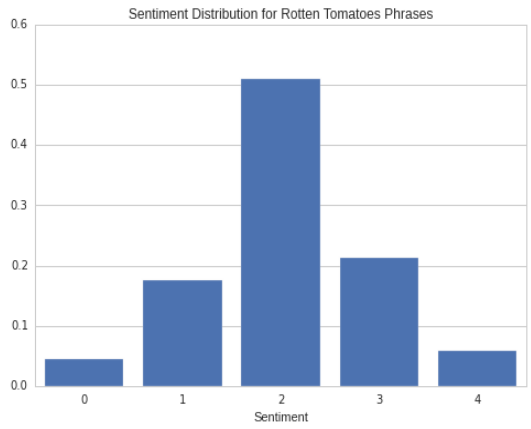
Table 1. Sentiment Label Coding

Sentiment	Label
Negative	0
Somewhat Negative	1
Neutral	2
Somewhat Positive	3
Positive	4

3. Exploratory Analysis

Analyzing the prior distribution of sentiment labels is important in developing an intuition and obtaining reasonable sense of what kind of predictions our models should make as described in the later section.

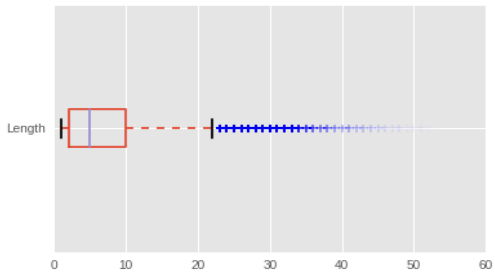
Figure 1. Sentiment Label Distribution



The sentiment labels appear to be very symmetric and slightly peakier than the normal distribution. The most frequent label is neutral which is the clear baseline that our basic model should predict.

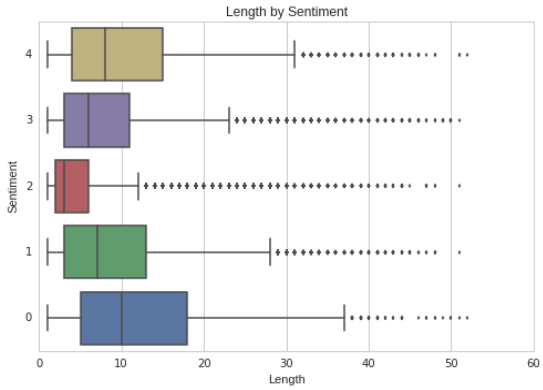
The distribution of the unigrams, bigrams and trigrams comprised 42% of phrases in the training set, and the following boxplot describes the lengths of the phrases and shows 75% of them being 10 or less and the rest mainly being 10-20 words in length:

Figure 2. Phrase Length Distribution



The fact that nearly half of the phrases are less than 5 tokens in length makes it reasonable for us to develop our features based on unigram and bigram counts that we can design models that take in a term frequency matrix as input.

Figure 3. Phrase Length Grouped by Sentiment



Finally the third figure is the most informative in that it shows 2-sentiment reviews having less variance and that there is a threshold at 5 tokens where we can design our model to use binning: using one model to tackle the case where the phrase has less than or equal to 5 tokens and another model for more than 5 tokens. Also if we are using regression models, phrase length can be a possible feature.

4 Prediction Objective

After performing a preliminary exploration of the data, the task/objective we are tackling is predicting the sentiment label given Phrases as features, using a supervised model, either classification or regression, from machine learning.

To evaluate our model, we submit a list of labels that our models output given the phrases from the training set as input online to Kaggle. Then Kaggle will calculate the categorization accuracy between 0 and 1, assuming that the competition either uses the number of correctly predicted labels divided by the total number of samples or the distance between 1 and the Hamming Loss:

$$\ell_{\text{Hamming}}(y, \hat{y}) = \frac{1}{|N||\mathcal{Y}|} \sum_{i=1}^{|N|} \delta(\hat{y}_i \neq y_i)$$

The main types of model we consider are those that deal with classification since there are 5 possible discrete categories, but we can also try a regression model since the sentiment labels are ordinal and scaled. Then we round the output of our regression to the closest integer.

A model that uses clustering could also be a possibility, and it would have to aggregate the labels of the closest neighboring training points. Unsupervised models however would not be appropriate in helping us predict the sentiment class.

4.1 Features

The combination of features we will use is a subset of the number of counts for each unigram/bigram and the number of tokens in the phrase.

The only pre-processing of the feature will be the lowercasing the text of the review. Also we leave in punctuation since some of them have sentiment labels assigned in the training set.

As described in our exploratory phrase, based on figures 2 and 3, we assume unigram and bigram counts to be reasonable features to represent as a term frequency or term frequency – inverse document frequency matrix. Finally given the difference between the variation in phrase length (number of tokens) for each sentiment category, we can use binning on a threshold like using the output of one model to handle phrases with less than or equal to 5 tokens and the output of a second one for phrases containing more than 5 tokens.

5. Models

5.1 Baseline

The most simplistic model we can use as a baseline to compare our results with more complex models to predict the majority sentiment category, and the most frequently appearing sentiment value is neutral: 2

$$f(\text{phrase}) \mapsto 2$$

And the score for the baseline model is 0.51789.

5.2 Bag-of-words Multinomial Naïve Bayes

The next model is also very simplistic in the naïve sense of just counting the unigrams and representing it as a term-

document matrix. The multinomial variation of NB can be described as:

$$c = \arg\max_{c \in C} P(c)P(d|c)$$
$$= \arg\max_{c \in C} P(c) \prod_{t \in d} P(t|c)$$

where the prior probability is

$$P(c) = \frac{N_c}{N}$$

and the conditional probability is

$$P(t|c) = \frac{N_{t,c} + 1}{N_c + |V|}$$

The accuracy obtained improves to 0.58681.

5.3 Linear Regression with Polar Words

The third model is an attempt to use linear regression with unigrams appearing in the non-neutral training phrases:

$$t \in V \setminus V_{c=2}$$

So that our model is of the form:

$$f(\text{phrase}) = \beta + \sum_{t \in V \setminus V_{c=2}} \delta(t \in \text{phrase}) \cdot \theta_t$$

The number of features is around 700, but the results of the regression model sets the accuracy below the baseline with a score of 0.50952.

5.4 Binned Multinomial Naïve Bayes

Given that we explored the length/number of tokens in phrases grouped by sentiment categories, we found that the mass of neutral phrases had less than or equal to 5 tokens, so we decide to use that as a threshold for our binned multinomial NB model. We trained two multinomial NB models based having more or no more than 5 tokens and predicted values correspondingly too.

As a result our model score increased to 0.60457.

5.5 Nearest Neighbor based on Cosine Similarity of TF-IDF

The model that used clustering of similar phrases based on TF-IDF features did not have adequate or reasonable computation time, but the decision function developed is listed below:

$$f(\text{phrase}) = \text{label}\Big(\arg\min_{\text{x} \in \text{Train}} \{ sim(x, \text{phrase}) \} \Big)$$

where

$$sim(x, \text{phrase}) = cos^{-1} \frac{< \text{tf-idf}(x), \text{tf-idf}(\text{phrase}) >}{||\text{tf-idf}(x)||||\text{tf-idf}(\text{phrase})||}$$

6. Related Works and Literature

The Rotten Tomatoes phrases data obtained for this study originated from Pang and Lee's work in which they describe using item and label similarity for metric labeling/positive sentence percentage and compare it with multi-class SVM (one-versus-all) and regression, and they found that incorporating PSP helps improve average accuracies. Also mentioned in the dataset section was Sochel et al.'s work in creating a sentiment tree bank and labeling the nodes using Neural Networks obtaining high accuracy of nearly 80% much past the baseline, which qualifies it as state-of-the-art. In the following section we will see that their results and conclusions are far accurate than our findings.

Other related datasets the 50,000 IMDb positive and negative reviews like in described in Maas et al.'s work in developing probabilistic model related to LDA that can learn word vector representations and is able to capture sentiment and semantics similarities[4].

The techniques and theories described in the cited works were too advanced to incorporate into the models used in this study but features and models that were mentioned in common included are Naïve Bayes, SVM tf-idf matrix representations and similarity measures. The following section on results and conclusions cover the use of Naïve Bayes and tf-idf matrix representations and an attempt at calculating similar phrases.

7 Results and Conclusions

The models we developed in this study do not perform as well as the state-of-the-art or even close to the top scores of the Kaggle competition. Some of the Kagglers implemented their own RNN.

The significant results and insight we gained in this study is that Naïve Bayes again outperforms linear regression in simplicity (i.e. no need to calculate the weight vectors, just count the number of times each unigram appears) and accuracy. And another significant result is that the binning threshold discovered in the exploratory section can help increase accuracy by 2%.

Table 3. Model Performance

Model	Score
Binned Multinomial Naïve Bayes	0.60457
Multinomial Naïve Bayes	0.58681
Baseline	0.51789
Linear Regression	0.50952

The feature representation that worked well is the term-document matrix, unlike the best fitting line found by linear regression. An explanation as to why linear regression performed worse than the baseline is that there is a high bias/misassumption that adding weights linearly based feature words represents the sentiment accurately. Because of the misassumption and high inaccuracy, the interpretation of the parameters for linear regression can not reliably represent the sentiment of the phrase.

The models used in this study were not complex, and scaling was not an issue given the size of the training and testing sets. If there were more time and resources to conduct the study, then overfitting could be estimated using cross-validation.

8. Acknowledgements

A deep token of appreciation for all members of the Data Science community at UCSD and the Computer Science and Engineering Department for giving the opportunity to offer a Data Mining course at an undergraduate level.

9. REFERENCES

[1] Pang, Bo, and Lillian Lee. "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales." *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2005.

[2] Socher, Richard, et al. "Recursive deep models for semantic compositionality over a sentiment treebank." *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*. Vol. 1631. 2013.

[3] https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data

[4] Maas, Andrew L., et al. "Learning word vectors for sentiment analysis."*Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011.