Natural Language Processing (Fall 2019)
New York University
Final Research Paper
12/17/2019

Sentiment Analysis on Rotten Tomatoes Movie Reviews

Holly Cao sc6220@nyu.edu
Laurence Ininda lbi213@nyu.edu
Angel (Anqi) Fu af2641@nyu.edu

Abstract

Sentiment analysis on movie reviews provides interesting challenges to accurately classify a review whether it be a simple phrase or an entire review. Our project will focus on implementing sentiment analysis techniques from a baseline algorithm to one that tackles the challenges that our dataset presents. We start from a Bayes Naive algorithm and try to add and refine features customized for movie review analysis. We also build on top of the existing Stanford CoreNLP algorithms and add features applicable to sentiment analysis on movie reviews, such that the program could have more accurate performance than Stanford CoreNLP developed on a general dataset. We aim to improve on the existing algorithms by working to exclude contents from the review that discusses the plot instead of the sentiment. The classification will classify the phrases on a five-point scale with the following values: negative, somewhat negative, neutral, somewhat positive and positive. We will be using the average distance from the right answer as our evaluation method.

1. Introduction

Sentiment analysis is the task of classifying a document, sentence, or phrase according to sentiment, often in the form of ordinal regression. The most common classifications are negative and positive, although more intricate classifications are less intuitive and more challenging. Sentiment analysis can be performed on various datasets to predict or assign sentiment to a group of text such as movie reviews or email messages. Our dataset will be The Rotten Tomatoes Movie Reviews originally collected by Pang and Lee. With several techniques used to perform sentiment analysis, what this dataset presents is the challenges of analyzing sentiment by overcoming obstacles such as negation, sarcasm, terseness and language ambiguity among others.

2. Experiments

2. 1 Stanford CoreNLP

Stanford CoreNLP is an algorithm that takes a sentence and uses the Recursive Neural Network (RNN) model to analyze different attributes of sentences including POS tags, named entity recognition, parsing, coreference resolution system, sentiment analysis, bootstrapped pattern learning, and open information abstraction. The model is trained by the Stanford Sentiment

Treebank, which annotates different attributes of sentences in the tree format. The algorithm is introduced in the paper *Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank* by Socher, Perelygin, Wu, Chuang, Manning, Ng and Potts.

We are hoping to tailor the Core NLP algorithm towards the task of identifying sentiments for movie reviews, and potentially towards all product reviews. In the specific case a movie reviews dataset, it is common for reviews to discuss both sentiments towards the quality of the movie, and the content of the movie. Although sentences providing contextual information to the plot or characters could potentially help a human reader understand the reviewer's point of view more clearly, they are usually misleading to a sentiment analysis program. For example, if a reviewer comments on the antagonist "Max" as being "the most disappointing person you'll ever know," the sentiment of the overall movie should be unaffected. These contextual information is nonetheless evaluated by Stanford CoreNLP and the sentiments calculated into the final result. We are interested in exploring features to identify and exclude sentences conveying contextual information from the review such that they are not considered for sentiment classification from the beginning.

Also in the task of sentiment analysis, we are hoping to get a set of test data scored by the writers of the reviews themselves, to get a more genuine sentiment score, which could be perhaps more extreme than human annotators because the writings are only captures of their sentiments about the film.

In the original paper of the algorithm, the authors talked about how originally they asked the annotators to annotate the treebanks on a scale of 7: very positive, positive, slightly positive, neutral, slightly negative, negative and very negative. Turns out that annotators mark few snippets on the very positive and very negative scale, therefore their algorithm only consists of the middle 5 categories. It is interesting what we encountered in the research, that the algorithm tends to predict the extreme sentiments more conservatively than the writers' own reviews. (seen in graph below)

2.2 Naive Bayes

The dataset provided by the Kaggle task is tab-separated values with each line containing the PhraseID, SentenceID, Phrase, Sentiment. Each complete sentence is parsed using the stanford parser, and every phrase after that is a part of a sentence. The stanford parser, which is part of StanfordCoreNLP, parses a sentence into a binary tree structure, and therefore, reverse-engineering the data into a parsed tree was an important aspect of tackling this task.

The Naive Bayes (NB) system was what we adopted as the baseline to assign sentiment to the Kaggle Task. Naive Bayes is a bag of words classifier, meaning that considers whether only the existence of a word in a document and not the position of the word in the document.

Training:

For Training, the system contains a dictionary that will hold key-value pairs of each word in the training documents, the keys are the words in the document while the value for each key is a list, `logpriorlikelihood` of 5 where index i is the number of times the word occurs as an instance of sentiment in index i. The values 0 to 5 for each index represents Negative, Slightly Negative, Neutral, Slightly Positive and Positive respectively. There is also a list, `logprior`, where `logprior[i]` is the probability that a phrase occurs as an instance of sentiment `i`.

Combine naive-bayes with another system.

The algorithm then runs through the entire document, taking until training is completed and the *logprior* is calculated. These two lists, `loglikelihood` and `logprior` are then used to assign sentiments to phrases in the test set.

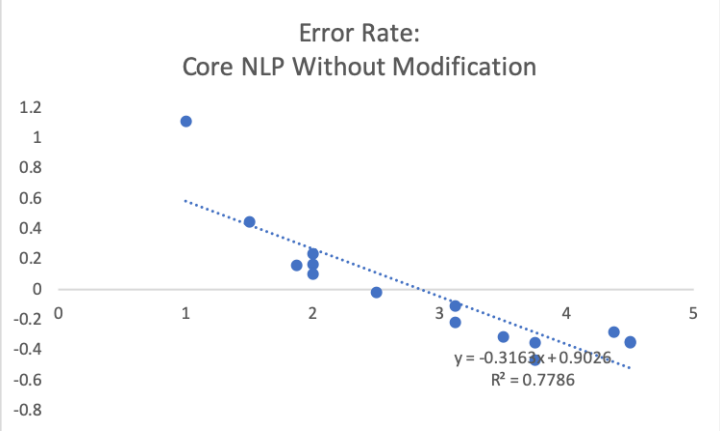
Development:

For developing the system, we partitioned the initial training set into a test set and a development set with a ratio of 7:3 on the initial training file provided.This enabled us to tune the NB system as we developed it and, in the future, as we test on other systems like those included in Talbot et (2015)

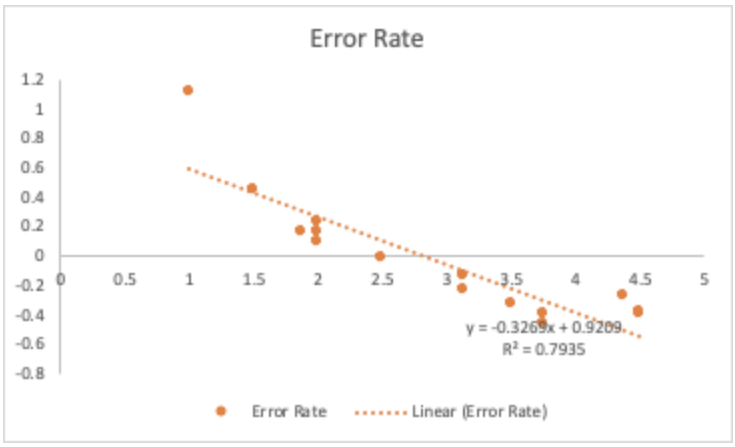
3. Evaluations

3. 1 Stanford CoreNLP

In order to evaluate our added features to CoreNLP, we used the original CoreNLP performance as the baseline. We took reviews from Rotten Tomatoes where the reviewer both wrote paragraph and gave a numeric rating out of five. When given 15 reviews to evaluate, CoreNLP without modifications has error rates shown in *figure 3.1.1* and our program has error rates shown in *figure 3.1.2*.

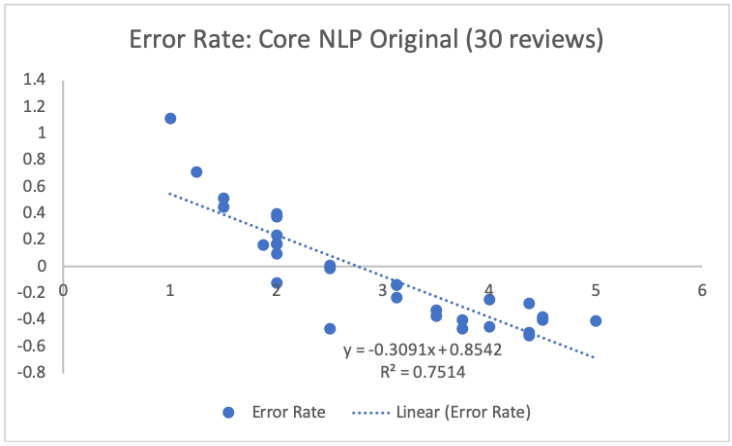


(figure 3.1.1 This is the error rate of CoreNLP with 15 reviews as input. The x-axis is the real score given by the reviewer on Rotten Tomatoes; the y-axis is the difference between the real score and the predicted score.)

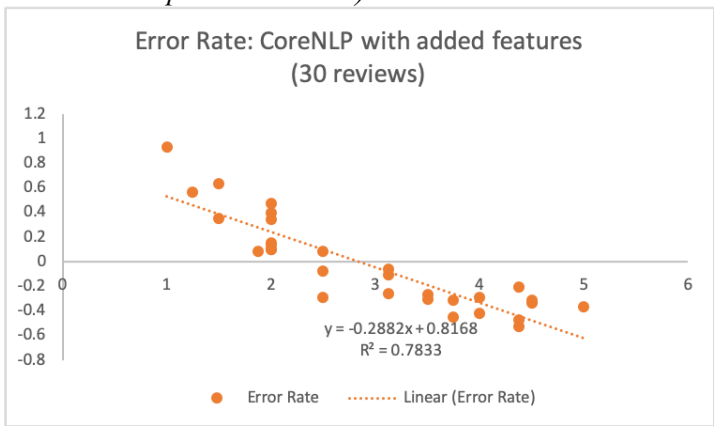


(*figure 3.1.2 This is the error rate of CoreNLP + summary and isRelevant feature, with 15 reviews as input. The x-axis is the real score given by the reviewer on Rotten Tomatoes; the y-axis is the difference between the real score and the predicted score.*)

With twice the input size, 30 reviews, the error rates of CoreNLP original and our program is shown below.



(figure 3.1.3 This is the error rate of CoreNLP with 30 reviews as input. The x-axis is the real score given by the reviewer on Rotten Tomatoes; the y-axis is the difference between the real score and the predicted score.)



(figure 3.1.4This is the error rate of CoreNLP with 30 reviews as input. The x-axis is the real score given by the reviewer on Rotten Tomatoes; the y-axis is the difference between the real score and the predicted score.)

When sentences we identified as summary of the review were given more weight in calculating the classification (isSummary() function), R^2increased from 0.7514 to 0.7833. isSummary() seems to have increased the error rates of CoreNLP classifications instead of improving them. Through the previous comparisons of different sizes, the error rates also seem to be uncorrelated with size of input.

In addition, as mentioned earlier in this paper, the annotators were reluctant to classify a review on either end of the spectrum: *extremely positive* or *extremely negative*. Our results and their error rates confirmed that the Sentiment Analysis system is more conservative than human reviewers. The program classified results are more closely clustered around *neutral* than the real score.

3.2 Naive Bayes:

We adopted a simple F-score scoring system for the NB system on the development set. Using a contingency table where the system output was compared to the actual output for each of the sentiments, the average precision was taken for all the scores to get a micro-average precision and recall score.

The system was then trained on the entire training set available and the output submitted on Kaggle for an accuracy score. This is how the system performed:

Development set precision	41.45%
Development set recall	21.62%
Development set F-Score	28.41%
Test-Set Average on Kaggle	0.51238

Conclusion and Future work

The Core NLP algorithm currently analyzes sentence-by-sentence. Some possible things that we have not implemented (successfully) in the Sentiment Pipeline for CoreNLP that could make the predictions better is to unskew the biases of Core NLP towards Rotten Tomatoes. Another thing that would be good to explore options on is to take advantage of the POS tags as well as some other features if possible to make logical rules to better filter relevant as well as important sentences. In the future there will hopefully a way to consider co-reference and other things in paragraphs, and also connect the sentiments in different paragraphs for the analysis. It would be another art of analysing why writer separate sentences in those paragraphs and what it says about the sentiment. Another possible thing that we can explore is using the Core NLP algorithm to analyze Amazon reviews for films and products, since as this research had

demonstrated, Rotten Tomatoes reviews have too complicated sentence structures to be captured by grammar rules and keywords. In the analysis of Amazon reviews, we can still have the ratings written by the authors for a more genuine rating data. A third possible thing to explore is to change the training set to only pertaining film reviews (for example the dataset on kaggle) for some industry-specific sentiment insights.

The Naive Bayes system implemented was simple enough to have an accuracy of over 50%, which is a good baseline score to develop upon. Adding a few more features such as sentence negation would certainly improve the score. Since the data provided by the Kaggle task was parsed using the Stanford Core NLP, more features could be attached to the tree structure that the data resembles. Since sentiment is seen building up, i.e. as one traverses up the tree, the most pressing question would be to see how sentiment changes up a tree as experimented by Socher et al. (2013).. Further combining the Naive Bayes system with other systems, leaving NB to handle the easier cases and passing on the harder-to-classify phrases to more advanced classifiers such as Support Vector Machines, such as the system adopted by Nguyen at al (2013), could provide a more accurate classification.

Our attempt in assigning different weights to sentences in the review, such as identifying sentences using "overall" and giving them more weight, did not lead to the same effects as expected. For the *isRelevant* feature, we realized our identification method was not a common grammatical structure most reviewers used. Excluding "movie+verb" structures while including "movie+adj" and "This movie is" was not conducive to improving program performance. These structures are too simplistic and not common in the dataset. A possible explanation for this result is that Rotten Tomatoes has a more critical and professional user base. Many film critics publish their movie reviews on the site and their comments on a film could be more sophisticated than the general public, using complicated language and grammatical structures.

The Rotten Tomatoes Dataset does provide advanced structure to experiment with in the future and the Stanford Core NLP has proved to be an exciting system work with, with certain insightful implementations and flavors that could be tweaked to perform sentiment analysis.

Appendix

All the code for the programs can be found on the github link here**.**

References

Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank, Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Chris Manning, Andrew Ng and Chris Potts. *Conference on Empirical Methods in Natural Language Processing* (EMNLP 2013).

Andrew L. Maas, Raymond E. Daly, et al. Learning Word Vectors for Sentiment Analysis. Proceedings of the 49th

Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. 142-150, 2011.

Bo Pang and Lillian Lee. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics.* 115-124, 2005.

Ainur Yessenalina and Claire Cardie. Compositional Matrix-Space Models for Sentiment Analysis. *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. 172-182, 2011.

Peter D. Turney. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. 417-424, 2002.

Ruth Talbot, Chloe Acheampong, Richard Wicentowski, SWASH: A Naive Bayes Classifier for Tweet Sentiment Identification, 2015

Nguyen, Dai & Nguyen, Dat Quoc & Pham, Son. (2013). A Two-Stage Classifier for Sentiment Analysis. 897-901.