LING 406 - Term Project Report

Emily Roller

eroller2@illinois.edu

1 Introduction

When one reads a review online while searching for the best new phone to buy, or deciding whether or not to go see a new movie in theaters, what must they do to make a decision? Obviously, they must analyze the texts they read or hear, decide whether or not the overall opinion of these texts is positive or negative or some other emotion, and then make a decision based on that analysis. Generally speaking, this is the process of sentiment analysis. Sentiment analysis is one of the most relevant topics of natural language processing and has wide-ranging uses and applications, from reading online reviews to determine the best product to buy to helping businesses better understand their customers' opinions to even generating predictions regarding stock market trends or political election results. Furthermore, sentiment analysis is not a "closed" topic - thought there are many models today that have very high performance, the changing landscape of language use with respect to slang, social media, emoji use, and more has kept both sentiment analysis model developers on their toes.

2 Problem Definition

We define sentiment analysis more specifically as the process of identifying positive or negative sentiments or opinions in text. Sentiment analysis is a question of both linguistic understanding and machine learning; that is, good sentiment analysis requires natural language processing to first understand the sentiments of the training data as well as machine learning techniques to predict the sentiments of the testing data.

With this in mind, we will look at one of many applications of sentiment analysis: movie review classification. This project will process movie reviews and classify them as having positive or negative sentiment, and then determine the accuracy of its classifications. Since the review corpus contains thousands of reviews, we will not output the Positive or Negative sentiment predicted for each review, but instead the mean performance metrics for each machine learning model.

3 Previous Work

Approaching sentiment analysis first requires some background knowledge of the question at hand, and how the data must be processed and analyzed in order to best extract the its linguistic features. I used two resources to get a better understanding of the task: one of the provided papers, "Sentiment Analysis of Movie Review Comments", and an online article, "How To Perform Sentiment Analysis in Python 3 Using the Natural Language Toolkit (NLTK)", both cited at the end of the report. The paper gave me a more thorough understanding of how to define the problem of sentiment analysis, as well as some ideas for features and classifiers I could implement. The online article also was very helpful for understanding the more nuanced aspects of writing the code, such as how exactly to set up the data in order to use the classifiers. Previous work that was also very helpful for this project was my code for Assignment 3; I was able to use some of the k-fold cross validation portion in this project, for example.

4 Approach

My baseline approach was fairly simple; it used a Naive Bayes machine learning model and bag-of-words features. The text preprocessing was also somewhat minimal; I only did lower-casing and tokenization based on whitespace. I decided that I would implement more preprocessing methods and text features in the improved classifier so that I could have a simple but solid baseline to work off of. The improved model used decision tree and Bernoulli models in addition to the Naive Bayes ML model. In terms of text preprocessing, I did the same methods of whitespace tokenization and lower-casing, plus lemmatizing using the NLTK Lemmatizer and removing empty and punctuation-only tokens. I also experimented with removing stop words for the improved classifier, but I found that this was actually not beneficial at all; I suspect this is because the NLTK English stop words list is actually fairly lengthy and removing all such words could change the meaning or sentiment of the review a non-trivial amount. I manually experimented with using some additional feature sets such as POS tags, but the only one that I found improved performance was using the EmoLex database to give words associated emotions. I removed words with emotions contrary to the review's main sentiment (i.e. removing words tagged with "fear" as the main emotion from positive reviews), and I found that this improved the performance only when I removed negative words from positive reviews. To train and test the data, I used k-fold cross validation (with $k = 10$ in the baseline and $k = 10$ in the improved)

5 Results

The accuracy and precision results of the baseline and improved classifiers, with word-emotion association features are as follows:

Baseline

Naive Bayes Accuracy: 0.8272727272727274

Naive Bayes Precision: 0.8582738095238096

Improved

Naive Bayes accuracy: 0.8552188552188551

Naive Bayes precision: 0.8728174603174602

Decision Tree accuracy: 0.7575757575757576

Decision Tree precision: 0.7366402116402115

Bernoulli accuracy: 0.8367003367003368

Bernoulli precision: 0.7150573192239859

I wasn't able to implement F1 score or recall; for some reason, neither would work with the dataset and I wasn't able to find out why.

Based off of the results, it seems that Naive Bayes was consistently the best learning model for this problem. Even when I implemented POS-tag features I found that Naive Bayes still had the best performance. This seems in line with what I know from lecture and previous statistics classes; Naive Bayes is a good model for binary classification, which this particular sentiment analysis problem was. The Bernoulli model (which is actually a Naive Bayes Bernoulli model) performed second best, and Decision Tree performed the worst. However, none of the accuracy or precision scores were below 50 percent, which was to be expected since even a random guessing model should be expected to achieve at least 50 percent accuracy.

6 Discussion and Conclusions

Both working on this project and learning about sentiment analysis in lecture made it incredibly apparent how challenging natural language processing is. Even just a binary classification problem is absolutely no small task; countless linguistic challenges such as scope of negation and sarcasm make it much more complicated to extract meaning from text. If I had more time or more NLTK experience I think a good next step to improve my classifier would be to start addressing some of these issues. I think detecting sarcasm would be an especially appropriate challenge since movie reviews, especially negative ones, definitely seem to be the kind of text that could contain more sarcasm. In terms of my project implementation

specifically, I think the thing I would want to change the most would be how I represented text features - instead of using dictionaries and numpy arrays, I think the most convenient way of representing features would have been a pandas dataframe, like in Assingment 3. This way, it would have been easier to implement an incremental or leave-one-out approach.

References

[1] Yessenov, Kuat, and Sasa Misailovic. "Sentiment Analysis of Movie Review Comments" Massachusetts Institute of Technology, Spring 2009.

[2] Daityari, Shaumik. "How To Perform Sentiment Analysis in Python 3 Using the Natural Language Toolkit (NLTK)." DigitalOcean, 28 Jan. 2021, www.digitalocean.com/community/tutorials/how-to-perform-sentiment-analysis-in-python-3-using-the-natural-language-toolkit-nltk.