LING 406 Final Report

Sentiment Classifier

Hongshuo Zhang (hz13)

May 11, 2020

Introduction

In the golden age of the internet, much of human activities are migrated online. Nowadays, the internet contains many services that are essential to people. Usually, people nowadays rely on other people's commentary when choosing a service online. That's where the area of text sentiment analysis comes into play. Text sentiment analysis gives people an idea of, for example, people's sentiment on a product sold on Amazon, by processing the review that people post without explicitly mentioning what their sentiment is for a particular product. It is important to work on it today because it is a very important tool for companies, especially social media companies, to quickly gauge public opinions.

Problem Definition

In the context of Computational Linguistics, sentiment analysis can be seen as a huge blackbox, with a string as an input and with a sentiment as an output. Specifically inside of these blackboxes, there needs to be a system in place to learn the polarity of sentences (by polarity, I mean whether a sentence expresses a sad, joyful, or shocking sentiment, for example) and make predictions on the polarity of sentences based on what the system has already learned. Exact Implementation of such systems varies, as there are many ways to approach this problem. The input training/testing dataset can be preprocessed in different fashions and the preprocessed dataset can be learned with many machine learning algorithms. For this assignment, I am going to showcase the approaches I tried to build a sentiment analysis system.

Previous Work

The study of sentiment analysis started a long time ago, around the beginning of the 20th century, long before mass-produced computers were available to the populace. However, few publications of this particular field was published before the advent of computers and the internet since there weren't a lot of computerized reviews to study from. A Finnish paper discovered that 99% of academic publications on sentiment analysis were published after the year of 2004.[1]

The earliest academic publication I could find on sentiment analysis was written by Peter D.Turney from the National Research Council of Canada back in 2002.[2] In his paper, he talked about an unsupervised machine learning algorithm that classifies each review, which is first try to assign each word from the review with a part of speech tag, use an algorithm to determine

---

[1]The Evolution of Sentiment Analysis - A Review of Research Topics, Venues, and Top Cited Papers
Mika V. Mäntylä, Daniel Graziotin, Miikka Kuutila
https://arxiv.org/pdf/1612.01556.pdf
Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews
Peter D.Turney
[2] https://arxiv.org/pdf/cs/0212032.pdf

each word's *semantic orientation,* finally assign each review with a sentiment, based on the *semantic orientation* of all words in that review.

Approach

I used the nltk movie reviews dataset to test and train my sentiment analysis system. I first developed an easy to develop baseline system that yields a relatively high accuracy of roughly around 80%. Below is an account of how I developed the baseline system:

- First, I downloaded the movie reviews dataset, randomly shuffled the dataset so that test result won't favor either sentiment, then I split 80% of the dataset to train and the rest to test.
- As I trained the dataset, I counted the number of occurrences of a word belonging to each sentiment.
- After I trained the dataset, I computed the probabilities of each word from a sentence if they were all negative or positive and compared which sentiment was more likely for this sentence.
- There might be out of vocabulary words in the testing set that we might not have come across. In this case, I applied Laplace smoothing with a smoothing factor of 1.

However, this accuracy is not high enough for me. So I compared three machine algorithms to see which machine learning model yields a higher accuracy. Below is an account of how I developed the machine learning system:

- Text-Preprocessing: I first converted each review from the movie reviews dataset into a string, then I tokenized and converted each word from the string into part of speech that they belong to.
- Since each of the sklearn classifiers expect a sparse matrix as a dataset, therefore, I need to make sure that the dimension of all reviews are equal. However, each review has a different length ranging from 200 words to 1100 words. Therefore, after some tweaking, I only chose the first 95 words from each review to train and test. In other words, the feature vector I chose is 95 elements long, each element is the part of speech that each word corresponds to. (I also tried only feeding the exact words from each movie review and a tuple of the exact words and their part of speech. However, the performance was not as good.)
- Before sending the dataset for training and testing. I first need to encode words and the sentiment. Each unique type of word gets a unique number ranging from 0.00 to 0.45 (there are only 45 parts of speech available from the nltk pos tagger.). As for sentiment labels, I encoded positive sentiment to be 1 and negative sentiment to 0.
- Finally, I processed the dataset using the following classifier -- Naive Bayes, Decision Tree and K-Nearest Neighbor.

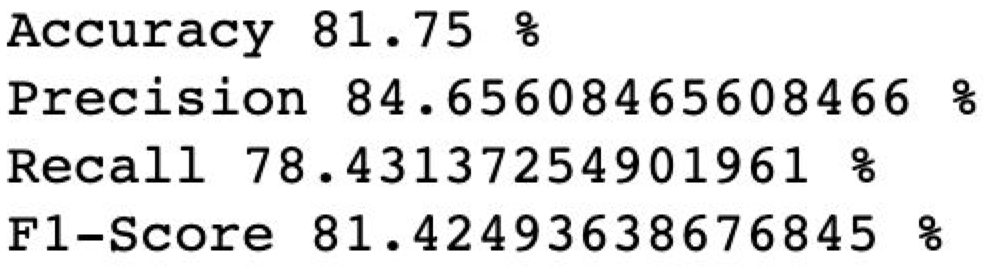
Empirically speaking, the performance of Decision Tree blows my mind. This machine learning algorithm yields an accuracy of 95% percent. 30% higher than other machine learning algorithms!

Results

In this report, we try to analyze the performance of the system with accuracy, precision, recall and F1 score.

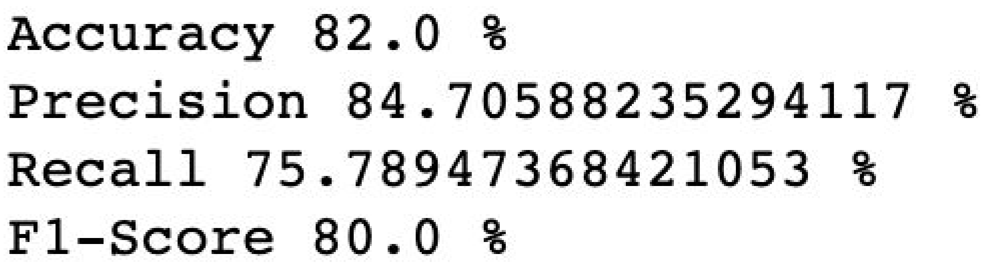
Here is the list of all features I tried for my baseline model:

- Takes all tokens from a sentence as unigram. (feature: all words in a sentence)



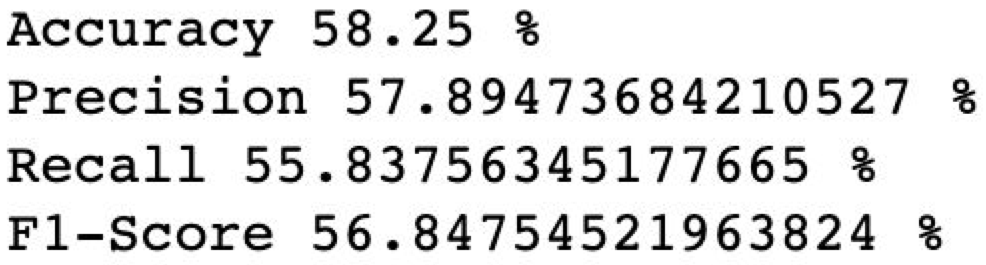
```
Accuracy 81.75 %
Precision 84.65608465608466 %
Recall 78.43137254901961 %
F1-Score 81.42493638676845 %
```

- Takes all non-trivial tokens from a sentence as unigram. (stripped off punctuation marks)



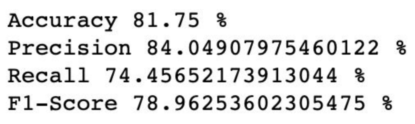
```
Accuracy 82.0 %
Precision 84.70588235294117 %
Recall 75.78947368421053 %
F1-Score 80.0 %
```

- Takes all trivial-tokens from a sentence as unigram.



```
Accuracy 58.25 %
Precision 57.89473684210527 %
Recall 55.83756345177665 %
F1-Score 56.84754521963824 %
```

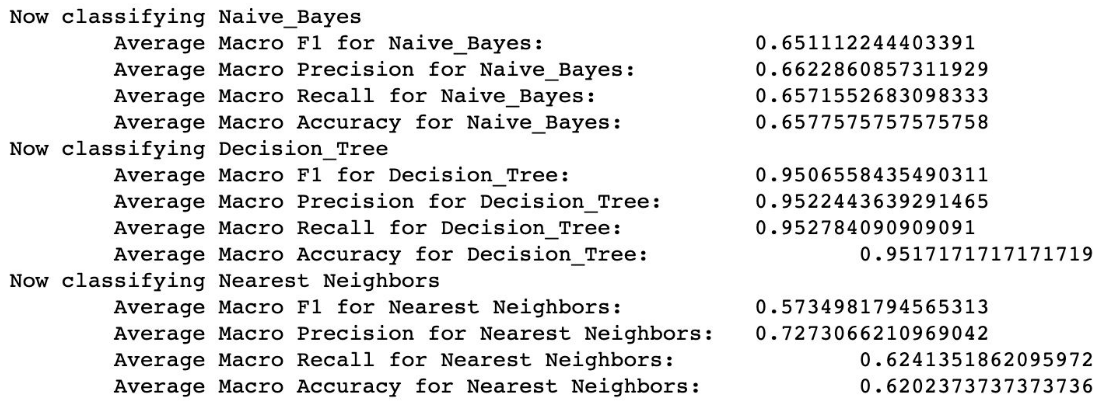
- Takes only nouns, verbs, advs and adjs from a sentence as unigram.



```
Accuracy 81.75 %
Precision 84.04907975460122 %
Recall 74.45652173913044 %
F1-Score 78.96253602305475 %
```

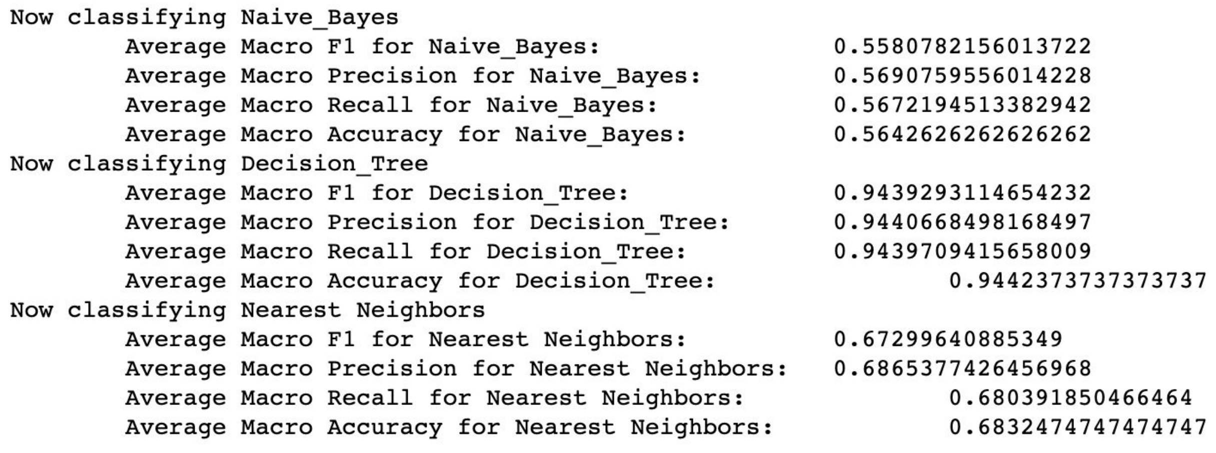
At this point, I fail to see which feature will significantly boost my accuracy, these features yield similar accuracy, precision, recall and F1-score (except the obvious outlier) and I am not sure which approach is going to be the most optimal.
(refer to baseline.ipynb)

Here is the list of all features I tried for my machine learning models:
The performance of sentiment analysis is not directly related to the absolute size of the feature set. For this task, I switched back to the Decision Tree Model and I tried playing with various feature sets. My first feature set contains every word in the document, then I took the first 95 words of each review for training and testing. Here is the end result.



```
Now classifying Naive_Bayes
        Average Macro F1 for Naive_Bayes:             0.651112244403391
        Average Macro Precision for Naive_Bayes:      0.6622860857311929
        Average Macro Recall for Naive_Bayes:         0.6571552683098333
        Average Macro Accuracy for Naive_Bayes:       0.6577575757575758
Now classifying Decision_Tree
        Average Macro F1 for Decision_Tree:           0.9506558435490311
        Average Macro Precision for Decision_Tree:    0.9522443639291465
        Average Macro Recall for Decision_Tree:       0.952784090909091
        Average Macro Accuracy for Decision_Tree:             0.9517171717171719
Now classifying Nearest Neighbors
        Average Macro F1 for Nearest Neighbors:       0.5734981794565313
        Average Macro Precision for Nearest Neighbors:   0.7273066210969042
        Average Macro Recall for Nearest Neighbors:           0.6241351862095972
        Average Macro Accuracy for Nearest Neighbors:         0.6202373737373736
```

The result seems to be pretty good, but I wonder what will happen if I try to take each word and its part of speech into account? Here is the end result.



```
Now classifying Naive_Bayes
        Average Macro F1 for Naive_Bayes:             0.5580782156013722
        Average Macro Precision for Naive_Bayes:      0.5690759556014228
        Average Macro Recall for Naive_Bayes:         0.5672194513382942
        Average Macro Accuracy for Naive_Bayes:       0.5642626262626262
Now classifying Decision_Tree
        Average Macro F1 for Decision_Tree:           0.9439293114654232
        Average Macro Precision for Decision_Tree:    0.9440668498168497
        Average Macro Recall for Decision_Tree:       0.9439709415658009
        Average Macro Accuracy for Decision_Tree:             0.9442373737373737
Now classifying Nearest Neighbors
        Average Macro F1 for Nearest Neighbors:       0.67299640885349
        Average Macro Precision for Nearest Neighbors:   0.6865377426456968
        Average Macro Recall for Nearest Neighbors:           0.680391850466464
        Average Macro Accuracy for Nearest Neighbors:         0.6832474747474747
```

It seems a little bit lower for the Decision Tree model, however, the accuracy for Naive Bayes and Nearest Neighbor actually plummeted. This leads me thinking, do we need to take care of each word at all? Can we replace each word with the part of speech of each word in the feature set? Here is the end result.

```
Now classifying Naive_Bayes
        Average Macro F1 for Naive_Bayes:                0.5945666407104399
        Average Macro Precision for Naive_Bayes:         0.5958112281776623
        Average Macro Recall for Naive_Bayes:            0.5960295423283979
        Average Macro Accuracy for Naive_Bayes:          0.5972828282828283
Now classifying Decision_Tree
        Average Macro F1 for Decision_Tree:              0.9527096415489273
        Average Macro Precision for Decision_Tree:       0.9530015155066428
        Average Macro Recall for Decision_Tree:          0.952961333878887
        Average Macro Accuracy for Decision_Tree:                0.9527474747474749
Now classifying Nearest Neighbors
        Average Macro F1 for Nearest Neighbors:          0.6600389206706108
        Average Macro Precision for Nearest Neighbors:   0.6702061004098779
        Average Macro Recall for Nearest Neighbors:              0.6643329612096799
        Average Macro Accuracy for Nearest Neighbors:            0.6707424242424241
```

To conclude, it seems that this is the best arrangement we have for this task. i.e. Take the part of speech of each word for training and testing. However, the accuracy for this arrangement isn't higher than other feature sets by a very wide margin. I guess this arrangement works better than others that I have tried since it only has 45 different encodings, which dramatically reduces the number of different tokens for the classifier to learn.

Discussion and Conclusion

The field of sentiment analysis is very new and it's rapidly evolving. What I learnt from this project was that the study of sentiment analysis is interdisciplinary and there are often multiple approaches from multiple fields of studies to actually implement a good sentiment analysis system. There are multiple ways of improving the solution to this problem, namely improving the machine learning algorithm, or improving how text is represented inside the system.

I regret that I do not have much time to explore many other good approaches to implement the sentiment analysis system. My approach only cares about the occurrence of each individual word (or occurrence of each individual kind of part of speech tag), not the actual sentiment of each word, the relationship between different words in a string, nor the sentence structure. If I still have time, I will explore these aspects to build a better, more thought-out sentiment analysis system.