Text mining

Harrison Young

Overview

During English class in high school I remember my teacher telling us that there is actually very little difference between Shakespeare's comedies and his tragedies. To see if this is true I downloaded a number of Shakespearian plays off of Gutenberg.org and performed word frequency and sentiment analysis on them. If my English teacher was right than the Shakespearian comedies and tragedies should have similar positive and negative sentiment values.

Implementation

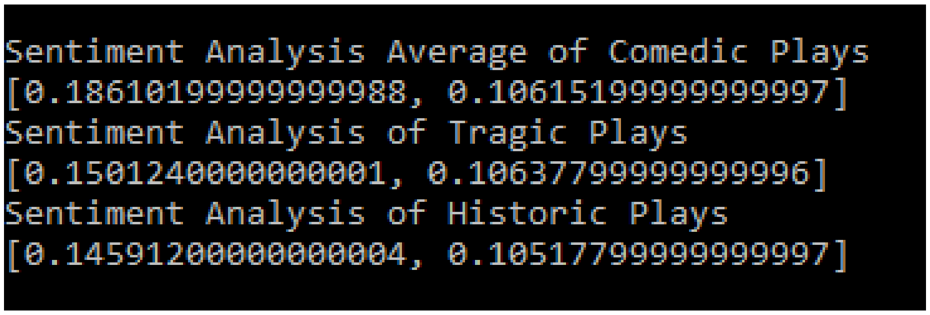
The text mining program is divided into two sections. The first downloads plays off of Gutenberg.org and saves them to a file. The second section processes the text by removing extraneous text from the files and grouping the plays based on category. The last section is the analysis.

Sentiment analysis is the primary form of analysis used in this project however the text needed to be modified before analyses could be performed. The first modification is to remove the common words from the text file. The reasoning for this is words that are commonly found in all of the texts will not be useful for differentiating them. The program finds the 25 most common words in each type of play and removes the words that are commonly found in all three categories. During frequency analysis the text was stored in both list and dictionaries. Lists were used because of the ease at which words can be separated. Dictionary were used for storing the word frequency because the key value structure of a dictionary makes if very easy to store how many times a word is found.

After all of the common words were removed from the text, a random sample of words from each type of play were taken and analyzed using sentiment analyses. I decided to preform sentiment analysis on small samples rather than entire plays because I read that the NLTK sentiment analyzer is not very accurate at analyzing large bodies of text. The program analyses a random sample of 10 words at a time. In order to ensure the analyses is accurate the positive and negative values are averaged over 500 random samples.

Most of my functions are designed to only take one list at a time. The means that I have to call each function three times, once for comedies, tragedies, and histories. As a result of this design decision, new categories have to be hard coded into the text mining function. While this limits the functionality of the program outside of analyzing Shakespeare, I feel that it was a good design starting out because it simplifies the rest of the functions and makes the code in textmining a lot easier to follow. That being said if I were rewrite the code, I would probably have chosen a more flexible design.

Results



```
Sentiment Analysis Average of Comedic Plays
[0.18610199999999988, 0.10615199999999997]
Sentiment Analysis of Tragic Plays
[0.1501240000000001, 0.10637799999999996]
Sentiment Analysis of Historic Plays
[0.14591200000000004, 0.10517799999999997]
```

The following is the result of sentiment analysis of the 9 total plays, three of each category. The first element of the list is the positive intensity of the text and the second is the negative intensity. The first thing I noticed is all three types of plays have the same level of negative sentiment. This supports my teacher's claim that Shakespeare's comedies are just as dark as dark as his tragedies. While the comedic plays have just as much negative sentiment as the other two types, they also have almost 25% more positive sentiment. The last thing I noticed is that Shakespeare historic plays practically identical to his tragedies in terms of positive and negative sentiment.

This analysis reveals a lot about human perception and the importance of timing. The primary difference between comedies and traditions is that comedies always end in marriage where as tragedies end in death. While the final scenes of the play may not strongly effect the average sentiment intensity of the entire play it certainly effects the audience. This incite reveals a weakness in sentiment analysis in that how people reactive is more complex than just the number of positive and negative adjectives that are written.

Reflection

I thought this project went well. It drastically increased my application for functions. At the start of the project I wrote everything in textmining an only made functions when the code got so complex that I could not read it. By then end I wrote functions to perform a useful task before I had completely figured out how to apply it in the main code. One thing I need to improve on is my use of unit testing. Most of the project I used print statements and my own knowledge to identify bugs. While I thought this project was fairly difficult, I felt that I was adequately prepared for it.