Text Mining Project

Camille Xue

February 2017

1 Project Overview

For this text mining project, I analyzed two of Shakespeare's plays, *Hamlet* and *Romeo and Juliet*, primarily focusing on word frequency and sentiment analysis. I used Project Guteburg as my primary data source for accessing the content of these two plays. I primarily wanted to see if what similarities and differences existed between these two plays in these categories.

2 Implementation

This project consists of two main files. One of these files retrieves the text data from Project Gutenburg, and pickles this text into a separate text file. The main python file references these text files containing the text of each play instead of having to retrieve the play text repeatedly. For the main python file itself, the analysis is done with a histogram function that counts word frequencies, and the SentimentAnalyzer from the NLTK package. Additionally, in order to effectively analyze the text from the plays, I had several functions to modify the text and isolate the important information that I wanted to analyze.

Since I analyzed two different things, I had to modify the text differently for each function. For calculating word frequency, it made the most sense to strip the text down to eliminate extraneous characters like whitespace and punctuation, then split these words and store them individually into an list instead of as one string. For the histogram, a dictionary was used to store the count for each word individually.

On the other hand, for doing sentiment analysis, it made more sense to keep the text in string form instead of putting each word into a list. Instead, most of the work had to do with finding the dialogue for each character so they could be analyzed on individual levels and compared to each other. To accomplish this, I took advantage of play formatting in order to find starting and end markers for when a character started and stopped speaking, and used a recursive function in order to compile that character's dialogue for the entire play.

3 Results

3.1 Word Frequency

After filtering out common words, I could find some of the most common meaningful words for each character. In Romeo and Juliet, one of the most common words for many of the characters is "love," which makes sense because the play centers around their tragic romance. Hamlet on the other hand, does not focus on romance, thus it does not show up despite Ophelia having feelings for Hamlet.

3.2 Sentiment Analysis

Generally, most of the text was neutral, but it was still really interesting to look at the negative vs positive sentiments for each character. For both plays, the characters that served for comic relief such as the Nurse in Romeo and Juliet or Guildenstern in Hamlet, had higher positive scores and lower negative scores. Additionally, I found it particularly interesting that Paris, whom Juliet was supposed to marry, had high negative ratings and low positive ratings, making him one of the most negative characters that I analyzed.

Figure 1: Sentiment Analysis and Word Frequencies of Hamlet and Romeo and Juliet

Hamlet			
	NEG	POS	Frequent Words
Hamlet	0.11	0.148	man
Ophelia	0.098	0.19	good, sings, sweet
Horatio	0.099	0.103	speak
Polonius	0.089	0.139	ill, time, said
Guildenstern	0.042	0.177	we, good
Rosencrantz	0.081	0.137	we
Laertes	0.134	0.137	no, ill, nature
Romeo & Juliet			
	NEG	POS	Frequent Words
Romeo	0.147	0.192	love, more, death
Juliet	0.137	0.176	romeo, love, night
Nurse	0.098	0.16	lady, romeo, god
Paris	0.161	0.091	love, tears, give
Friar	0.143	0.191	love, romeo, death
Tybalt	0.134	0.106	villain

4 Reflection

For this project, I found that starting with simple building blocks and working my way up to more complicated things proved to be very effective for me. For example, I originally started with just Romeo and Juliet, and looked at word frequency for the entire play. Then I tried isolating Romeo's dialogue, then all the characters, and so on. After that, I decided to add Hamlet, which posed other problems because of differences in formatting. Adding a second play also forced me to make the entire file more modular so that the same functions would work for each play.

As for other important aspects of the process, I did feel that I learned a lot. At first I didn't do a lot of unit testing, but as it got more complicated, I saw the need for it and started implementing these tests as I went along. I was constantly testing small portions of my code to ensure that each function worked, before trying to run them all together.

Going forward, I want to improve on making more complicated programs, since I felt that for this project I didn't attempt something that was very challenging. I also want to manage my time better for future projects, since the former was a consequence of not having enough time.