Text Mining Project and Reflection
March 6, 2017
Gretchen Rice

Project Overview

I decided to analyze fairytale books from different places around the world. I compared fairy tales from Japan, India, and America. I looked at their sentiment and top ten words (by analyzing the frequency of each word). In my code I used vader for sentiment analysis and requests to get my text from Project Gutenberg.

Implementation

I pulled my text documents from Project Gutenberg which required me to use "requests" to get the text from the txt webpages. I did two analyses on the texts. I first did a sentimental value analysis by using "SentimentIntensityAnalyzer" to find the negative, positive, and neutral sentiments in each of the fairytales.

To determine the top 10 words in each text I first separated the texts into lists of words. I had to do a lot to the lists in order to get to the top 10. I had to strip characters from words so I could create a histogram of the words from each of the texts. One all the extra characters were removed I was able to create a histogram of all of the words. I sorted this histogram from highest frequency to lowest frequency. In the histogram I removed "filler" words like "the," "and," "is," and "some" and then returned only the top 10 words from the words left in the list.

I had a lot of decisions to make when creating my topTen function. There were a lot of things I did in that function that I originally wrote as longer loops and then realized that from a design standpoint, I could write them much simpler. For example, when I filtered out "filler" words from my histogram I originally wrote a for loop. This loop ended up failing (for reasons still unknown). I then used a lambda function, which worked a lot better. When I talked to another ninja we discussed a new method (wanted = [word for word in sort if word[0] not in unwanted]) which worked very similar to the lambda function and that is the design I ended up going with.

Results

I found quite a few interesting things when looking at the resulting sentimental value and top ten words. The sentiments were as follow:

Japanese Fairy Tales: {'neg': 0.101, 'pos': 0.139, 'compound': 1.0, 'neu': 0.76}
Indian Fairy Tales: {'neg': 0.082, 'pos': 0.104, 'compound': 1.0, 'neu': 0.814}
American Fairy Tales: {'neg': 0.092, 'pos': 0.119, 'compound': 1.0, 'neu': 0.79}

I was very interested to see that the Japanese Fairy Tales had the most negative sentiment, with 0.101. Meanwhile Indian Fairy Tales only had a negative sentiment of 0.082 (difference of 0.019) and American Fairy Tales had a negative sentiment of 0.092 (difference from Japanese of

0.009 and difference from American of 0.01). It seems they all had pretty equal positive sentiment. Again, Japan was bringing up the head of the pack with 0.139 while Indian and American Fairy Tales only have 0.104 and 0.119 (difference of 0.045 and .02, respectively). For neutral sentiment it seems the Indian Fairy Tales are in the lead with 0.814 with America coming close with .79 (difference of 0.024) and the Japan with a neutral sentiment of 0.76 (difference of 0.054).

I also looked at the top ten word in each of the fairy tales. Before determining the top ten words I first removed "filler" words from each of the lists. I removed the following list of words:

```
unwanted = ['a', 'an', 'be', 'and', 'are', 'from', 'for', 'the', 'they',
   'their', "they're", 'then', 'them', 'is', 'if', 'of', 'with', 'to', 'in',
   'was', 'that', 'as', 'at', 'this', 'so', 'had', 'on']
```

I wanted the top ten words to be representative of the actual writing and stories. The above words were either words that I thought of or words that showed up in the top ten that I did not think told me anything important about the stories. The following are the top ten words I got for each fairy tale and their frequency (number of times they showed up):

Japanese Fairy Tale: [('he', 1545), ('his', 1134), ('you', 819), ('i', 677), ('her', 671), ('it', 634), ('him', 631), ('all', 494), ('she', 466), ('old', 401)]

Indian Fairy Tale: [('he', 1280), ('his', 889), ('you', 847), ('i', 806), ('it', 677), ('him', 666), ('her', 523), ('said', 462), ('she', 445), ('not', 410)]

American Fairy Tale: [('he', 545), ('you', 429), ('his', 396), ('it', 361), ('i', 293), ('but', 268), ('she', 241), ('her', 237), ('not', 179), ('said', 175)]

I thought it was very interesting that the most common word for all three of the fairy tales was "he." The next top words in all three are "his" and "you" and all of the following words are very similar, a lot of the same words, as well, as you can see. Since "he" and "his" are in the top 3 words for all three fairy tales, I would make the assumption that the majority of the main characters in the stories are male. It is also interesting to me that "old" was one of the top ten words in the Japanese Fairy Tales. I know that in Japan older adults are much reverend, I wonder if that is portrayed in the fairytales. It is interesting to me that in all three of these fairy tale books the majority of the top ten words are the same, even if they don't have the same ranking.

Reflection

I think that I had a lot of set back with my project, including losing all my code. However, this proved to be a good learning experience. I had written a nice lambda function with a ninja, which was lost. When rewriting the code i had to try to remember how to write a lambda function and how to make it work properly. I ended up finding another place I could use a lambda function as well. Then, on my second attempt writing my program I worked with another ninja and learned some other cool python tricks to making cleaner code. I also had some difficulties with lines of code that had simple mistakes because I misunderstood APIs or typed something wrong. This was very frustrating, especially when ninjas also couldn't figure it out

and I had to completely rewrite the code in a new way, which can be good because I'm learning different methods of doing the same thing. I think that I could definitely improve, perhaps pickling would have been something good for this program. My code takes a long time to run sentiment value. I think I would want to look into how to make this run faster. I wasn't really sure how to use docStrings in this since I would have to somehow know the long lists of words I was creating. Because of this, I used print and return statements to do testing. I should look more into better testing methods to implement next time. I think that one thing I wish I had learned beforehand was lambda functions, they are so nice and extremely useful!