Data Mining Project

Project Overview

Using fairy tale texts from Project Gutenberg and The Internet Archive, I analyzed the frequency of color words appearing in different author's works (the brothers Grimm, Hans Christian Andersen and Charles Perrault) to get data on the historical mention and portrayal of color in fairytales and the differences between these author's usage.

Implementation

I pulled text from the Internet Archive (which is a Project Gutenberg mirror) and Project Gutenberg (directly), pickled it, turned each collection of stories into a punctuation free list of words, and used a dictionary of color words to search through my lists; the first value in the dictionary was the word, and the second the number of times that the color word had appeared in the fairy tale (or tales) being analyzed. I analyzed each author's work en masse (I used one collection of fairy tales each) and left the header/footers on as they are very unlikely to contain a significant number of color words.

The search function, color_searching, returns a dictionary for each tale (which is in list format) that it is given as an argument, in the format (color, frequency) - the dictionary is used as a counter. Another function, tale_searches, was built to iterate over the list of tales (which are word lists) that it is given as an argument, and call color_searching for each of them, returning a list of dictionaries (one derived from each tale in the list of tales). Tale_slicing also iterates over the list of tales (which, as the tales are lists, is a list of lists) so this program (the text analysis part) is fully modular and easy to expand if one wanted to analyze more works in the future. I used the list of dictionaries to make a pair of lists which were then used for for graphing.

I really focused on using mutable data structures in this project; by using lists and dictionaries I was greatly able to simplify my code by changing dictionary values. Lists of lists are easy to navigate (tales -> words in the tales) and lists of dictionaries (tales -> dictionary (color word, frequency)) are also easy to work with. I could have used tuples containing dictionaries and tuples contain lists, and I considered doing so; however, I was concerned that the immutability of tuples would really negatively impact my ability to use this code in the future as a basis for further work - lists are much easier to work with if I would like to do something such as iteratively pull more and more texts from Project Gutenburg and analyze each one, because instead of having to make a new tuple each time, I could just extend the list. Overall, I really focused on being able to easily expand upon my code in the future by having clean code, good data structures, and modularity. I felt that these were the things that I needed to work on, much more so than incredibly complex text analysis.

I used Seaborn for data graphing and dumped my each of my dictionaries (the actual output of my data analysis) into a list of tuples which I then had a helper function turn into a list containing two lists (keys and values) which was the format needed for Seaborn.

By changing one instance in pickling_import_text.py and one instance in text_filter.py (two different function calls, explained in the readme), you can analyze any text you'd like - however, the graphing will need to be tweaked in this case.

Reflection

Overall, I thought that I planned this project well. I thought about all of processes that I would need to do if I was a computer to look for these color words and made shell functions with docstrings for each of them as I went so that I would not forget anything. I focused on modularity which also really helped with this. I kept the project manageable in scope so that I could really dig into a better understanding of data structures, iteration, and modularity, and I think it paid off with some of my best code to date.

If I had done more scaffolding for myself and built more starter code initially, things would have gone faster, and that is what I would like to improve upon in the future. Additionally, due to the quasi-random ordering of dictionaries, it was very hard to write effective doc tests for almost any of my functions, so doctests weren't something I incorporated, although I did immense amounts of unit tests during development - while I didn't see a good way around the doc-test issue for this particular project, in the future it is something to be sure to include.

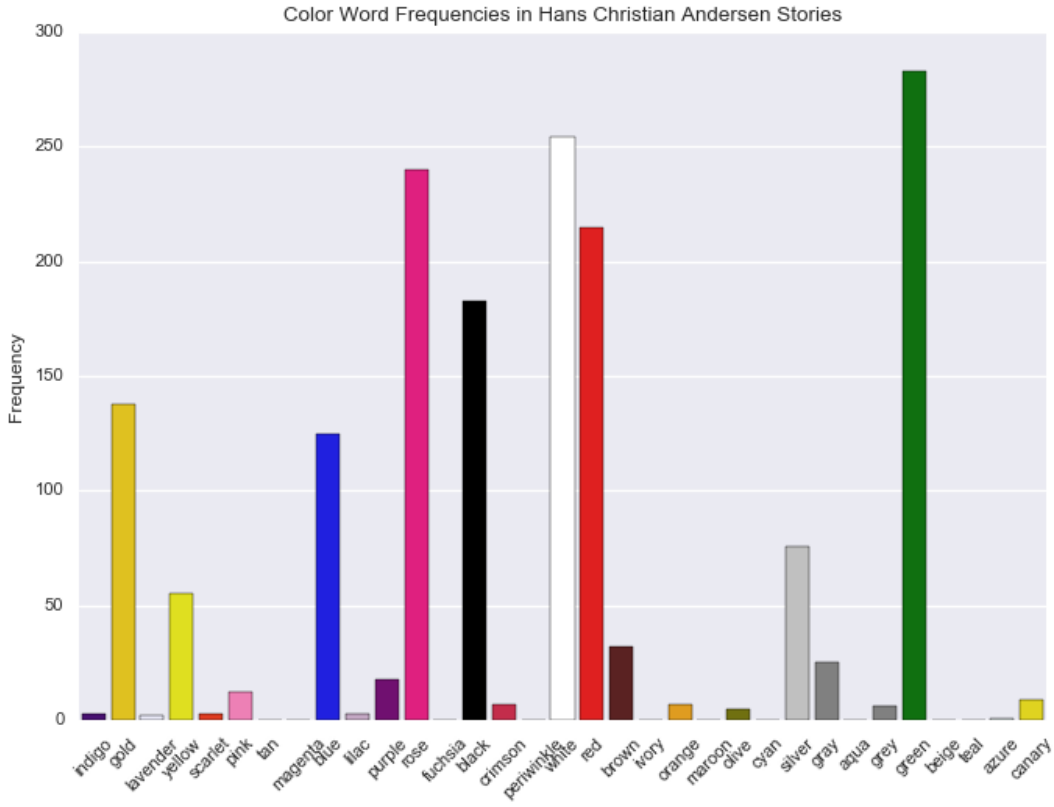
In terms of things I wish I had been able to do for this project that I did not have time for, I really wish I had time to modularize the graphing (which isn't the main focus of the project, but still would have been nice to have). I would have also liked to normalize the word frequencies over the length of the text because the texts were all of differing lengths. Lastly, I wish I had time to write a an additional .py which would call the necessary functions in pickling_import_text.py and text_filter.py with the grimm, andersen, and perrault names/text file URLs as default, but with an option to use any names and URLs the user specified (in short, I'd like to make it easier to use on other texts, which currently requires minor changes in both of these files - details within the readme).

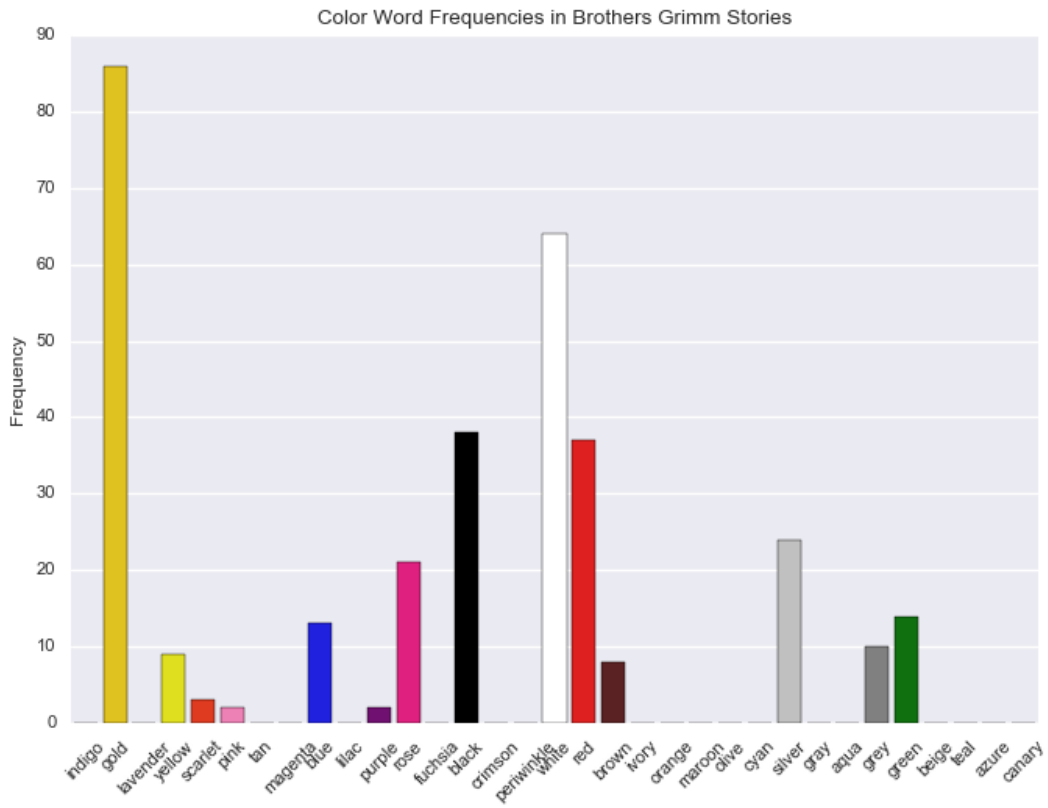
RESULTS

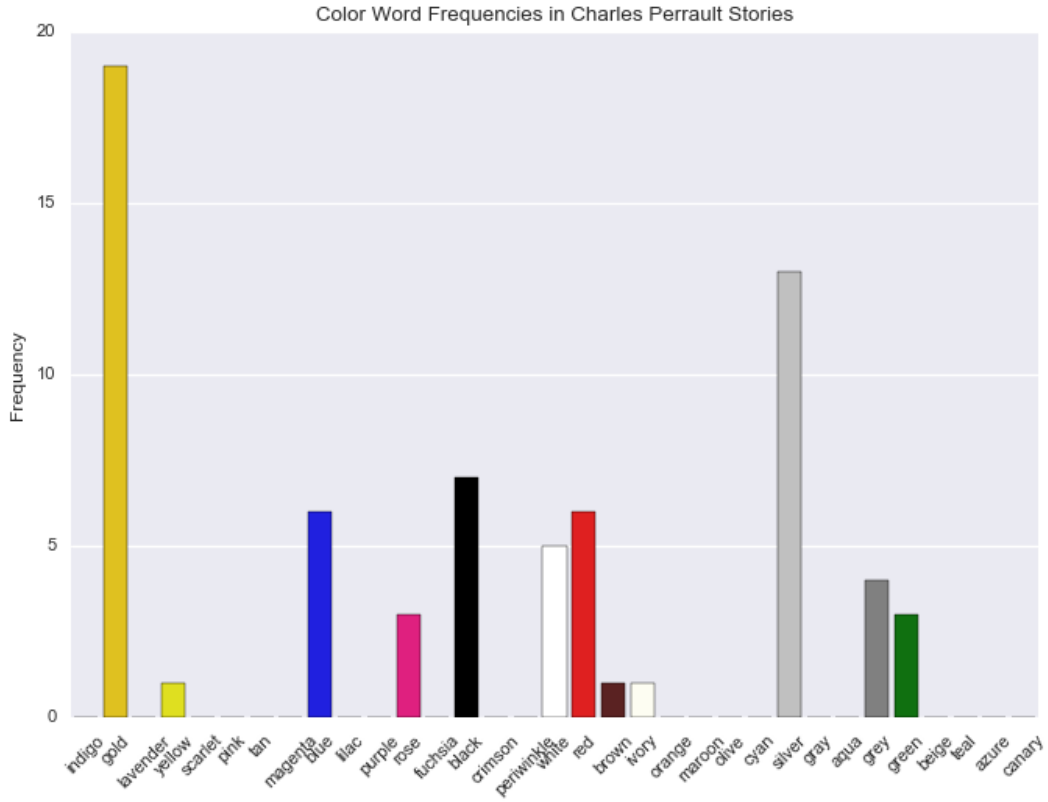
When I thought about what I would enjoy doing for this project, I wanted to do something with fairy tales because I really like folktales in general, and fairytales really inform our cultural context. I thought that colors would be really interesting to analyze, and I was right. Overall, Hans Christian Anderson used the most color words in his stories, followed by the Brothers Grimm, with Charles Perrault using the fewest color words in his stories. Of course, this is not a particularly good comparison due to the differing length of the story collections - Andersen's story collection was 2053574 characters long, the Brothers Grimm's was 549756 characters long, and Perrault's was 180233 characters long; unsurprisingly, the total number of color words in each collection was correlated with the length of the collection. If I had more time, I would have really liked to normalize the frequencies so that it would be easier to compare between story collection, but I spent a lot of time bug-fixing so I didn't have time to get to that.
However, I can look at the trends across the story collections, and the results were fairly interesting.
Gold and silver are very common across the board as color words, partially because silver and gold can also be used as words for money or high-value items("the pot of gold at the end of the rainbow" is a good example); additionally, silver and gold were revered during the times when these stories were written for similar reasons and carry wealth/royalty symbolism.
White was also common, I believe because of the associations that it has with purity and religion, and the relative rarity of white objects during the time period of composition.
Red was common as well, probably due to it's associations with life, blood, love, and royalty.

It was hard to draw conclusions from a particular author's use of color words individually; in the future it would be really interesting to do sentiment analysis on the sentences where color words appear to be able to draw more deep conclusions about the meanings and associations of each

color in the author's work; this was sadly beyond the scope of what I was able to do in this project, but I would love to do more work on this project in the future! Attached are three graphs of the word frequencies in the different author's stories.




Color Word Frequencies in Brothers Grimm Stories
Frequency
0
10
20
30
40
50
60
70
80
90
indigo
gold
lavender
yellow
scarlet
pink
tan
magenta
blue
lilac
purple
rose
fuchsia
black
crimson
periwinkle
white
red
brown
ivory
orange
maroon
olive
cyan
silver
gray
aqua
grey
green
beige
teal
azure
canary


Color Word Frequencies in Charles Perrault Stories
Frequency
0
5
10
15
20
indigo
gold
lavender
yellow
scarlet
pink
tan
magenta
blue
lilac
purple
rose
fuchsia
black
crimson
periwinkle
white
red
brown
ivory
orange
maroon
olive
cyan
silver
gray
aqua
grey
green
beige
teal
azure
canary

RESOURCES AND CODE SNIPPETS

In the course of this project I used a few resources and code snippets to help. These are listed below.

Starter Code/Softdez Website - I used the code and explanations about how to use pickle from the softdez website's page on this miniproject.

Punctuation Removal - this snippet of code was from: https://mail.python.org/pipermail/tutor/2001-October/009454.html, and I used it for punctuation removal with the tale_slicing function by calling it's helper function, is_punct_char, which I got from the above link. I knew I could write such a function, but this function was much more efficient and concise than what I would have done and was only a tiny part of the project.

Dictionary item dump output conversion to a pair of lists - this snippet of code was from: http://stackoverflow.com/questions/7558908/unpacking-a-list-tuple-of-pairs-into-two-lists-tuples. I used it to format the list.items(dictionary) output which was a list of two-item tuples, into a pair of ordered lists which was what I needed for graphing.

Colors - I got some ideas for what colors to search for from Crayola Crayon color names and Wikipedia. When I did the graphs, I wanted the color of the bar to match the color that had been searched, so I used Wikipedia and http://www.color-hex.com to get hex colors for each color name.

ThinkPython, Python documentation (python.org), and http://www.tutorialspoint.com/python/- I think this is self-explanatory - I read these all the time when I am not sure of things.