Mini-Project 3: TextMining

Rebecca Patterson

Overview

This project was a word comparison between The Wonderful Wizard of Oz by Frank Baum and Robin Hood by J. Walker McSpadden. Techniques used were dictionaries, sorting, and tuples. The stories for this project were obtained through Project Gutenberg. Since both stories are adventurous, and one with a touch of fantasy, I thought it would be interesting to see the similar/differing ways language was employed in the stories. To do this, I wanted to look at the top ten most used words in each text (that are five or more letters long), then at the twenty longest words that the books had in common.

Implementation

This code is built on a single dictionary. The data being used was obtained in the format of a long string, so the first function in the code, create_dictionary, converts that string into a list of strings of individual words. To do so, it breaks the string at every space, and then strips it of punctuation. It also puts all words into lower case to avoid repeating words with different capitalization. These words are then stored in a dictionary where the keys are the words and the values are the number of occurrences of that word.

The dictionary created is used in the next level of functions, top_ten and common_words. Both functions deconstruct the dictionary into a lists of types that they can utilize so that list sorting commands can be used on them. top_ten makes a list of touples because it needs to know the word count in addition to the word, and common_words uses strings because it only deals with the keys.

The overarching function, compare_books, ties all of these lower functions together. All it has to do is call the lower level functions using the input from the files saved in the script get_text, and print out the resulting lists in a manner that is readable to the user.

The most significant design decision made was for the first function. I had spent a large amount of time writing a while loop with and if else statement that would turn the string of text into a list. The last iteration of this that I wrote still did not account for all of the possible punctuation marks. After a bit of help and research, I found a single line of code that could replace the entire loop I had been writing that actually did account for all punctuation. For that reason, I chose to incorporate this line instead of the loop I sank so much time on writing.

Results

The top ten most used word in The wonderful Wizard of Oz (Table 1) and in Robin Hood are (Table 2) both reveal that the characters names and identifying traits are the most repeated words in the books. When you think about it, this makes sense for both stories. As adventure stories, both incorporate quite the cast for the main characters to interact with.

Word	doroth	scarecro	woodman	grea	littl	ther	witch	could	aske	woul

	y	w		t	e	e			d	d
Count	347	219	176	142	139	128	125	120	114	113

Table 1

Word	robin	little	sheriff	their	would	other	while	which	there	again
Count	636	232	202	187	184	172	139	131	109	109

Table 2

The twenty longest words that the two books shared were: *wide-spreading, reproachfully, entertainment, thoughtfully, respectfully, satisfaction, good-natured, nevertheless, disappointed, straightened, congratulate, disappeared, industrious, comfortable, immediately, straightway, glittering, afterwards, threatened, altogether.*

Most of these words are adjectives. I think that it is interesting that even though the story-lines do not have that much overlap besides the sense of adventure, the authors employed similar language when describing things. I'm particularly fond of "glittering" and "industrious."

Reflection

I liked that the content from the last few reading journals were relevant to this project. It was nice to be able to refer to code that I had already written and commented. Finding other examples online was good too, but I find that it is easier to follow my own logic than someone else's. I probably should have planned out my time a bit more carefully so that I could get some more help on things and dig into the analysis more. As it stands though, I feel like I did learn quite a bit from the code that I did write.