**Shreya Rangarajan**

## Project Overview *[Maximum 100 words]*

Using the Gutenberg database for text, I obtained texts written by Jane Austen and Nathaniel Hawthorne. I started out with the goal of only determining whether there were similar most frequent words in multiple texts. For example, comparing Emma by Jane Austen and Scarlet Letter by Nathaniel Hawthorne resulted in 290 words that were frequently used in both texts. I went to a step further and used the I used the lexical diversity analysis process to determine which books corresponded to which authors. If they had similar lexical diversities, they were written by the same author.

## Implementation *[~2-3 paragraphs]*

I have 5 main functions in my program that relate to each other. The main algorithm functions are the ones that compute and plot the lexical diversity of a text. The functions at the top are used to compare the top 500 most frequent words between texts. Having a function for storing the most frequent words in a text allowed me to use that when comparing most frequent words against different texts in another function that took in a list of texts. This made the code simpler and shorter.

One design decision that I made was that I decided to make a separate function to store all the comparisons and their commonality of most frequent words along with their titles instead of doing that in the compare_authors function. I made this decision due to better flow of the program and more clarity for someone trying to figure out what the functions do. In my opinion, the choice I made was better than my initial thought because it didn't clutter up the compare_authors function. The output of the frequency_table function allows the reader of the code to understand which two books are being compared along with how many most frequent words are similar for the books. It provides more clarity instead of outputting a number as I do in the compare_authors function.

## Results *[~2-3 paragraphs + figures/examples]*

Figure 1 shows the lexical diversity of 4 different books. Two are by Jane Austen and two are by Nathaniel Hawthorne. I found it interesting that the lexical diversities of the Jane Austen books were very similar and the lexical diversities of the Nathaniel Hawthorne books are very similar. This program might allow someone, if given a list of random books with no author, to determine if the books were written by the same author.

Since the lexical diversity takes the number of unique words and divides that by the entire length of the text, it is very plausible that authors will be able to be distinguished based on different vocabulary used in their writing.

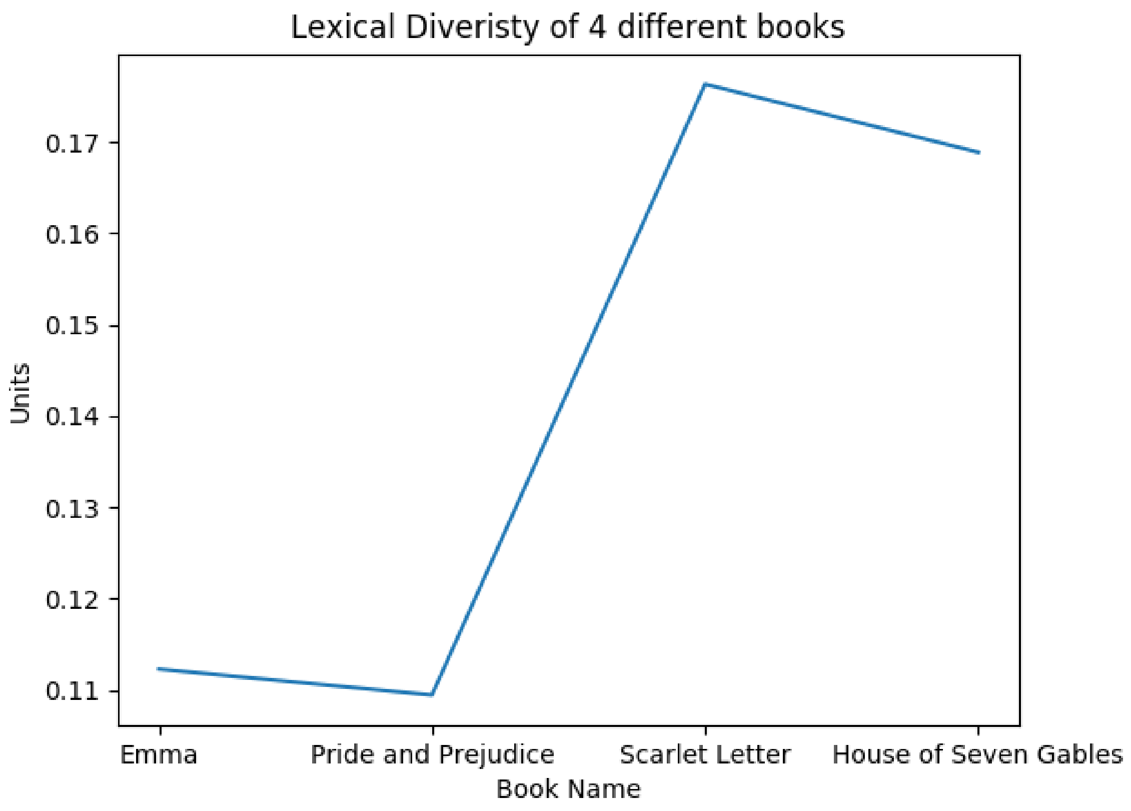

**Figure 1.** Lexical Diversity of 4 different texts

**Reflection** *[~1 paragraph]*

From a process point of view, what went well? What could you improve? Other possible reflection topics: Was your project appropriately scoped? Did you have a good plan for unit testing? How will you use what you learned going forward? What do you wish you knew before you started that would have helped you succeed?

I think my use of doctests went well, except for one doctest that simply would not work (disclaimer in the comments) even after going through it with a NINJA. I believe that the project was well scoped and I think I even went a little beyond what I thought I could by including the lexical diversity functions. One thing I could do better next time is implementing the doctests earlier in the process of writing the program. For some functions, I would write the function, test it and then write the doctest. Writing them before I write the function next time might be better. Using dictionaries and tuples have made it easier for me to understand the two data structures a lot more and I hope to use them more in later projects.