Kevin Solorzano

Professor Yang Liu

IS-392

March 14, 2018

**Frequency-Based Index Report**

**Part A**

Programming Language Used: Python 3.6

Libraries or Packages used: NLTK, Regex, BeautifulSoup, OS

Major Functions:

1. checkChar() – user-defined function that checks every character of a token and returns true if the token has no special characters or integers and false if otherwise.
2. cleanMe() – another user-defined function that removes the excessive script data and meta data embedded in the HTML code so that the frequency program doesn’t save them.
3. word\_tokenize() - a NLTK-based function that takes a string text and tokenizes it.

Processing:

The program begins with a for loop that examines every HTML file in the directory path specified. This path contains the 500 downloaded HTML pages from the previous assignment (web crawler). Moving on, it creates a dictionary named ‘collection’ that will store unique words and its frequency on the page. After opening the HTML file and using the cleanMe() function, we use the page text and begin tokenizing it. At this point, we begin to remove all stop words from the page text using the NLTK-based package. Once the tokenized- page text has been filtered from all stop words, we begin looking at every token and use some conditions that it must pass before it can be saved to the dictionary. These conditions (if,else) consist of the token being 1 character long, using the checkChar() function, and checking to see if the current token position stumbled into the further excessive meta data that the cleanMe() function couldn’t scrap out. If it passes those conditions (must all be FALSE), then the program can begin placing the tokens in the collection. First, it checks if the token isn’t present in the collection, by which it will create a new entry with the token as its key and assign it an empty value. If the entry already existed, then that existing entry’s value gets incremented by 1 to refer to its frequency occurrences. Same happens for new entries after they are created. After that, the program looks into every token saved in the collection and creates new entries for each token that isn’t present in the index dictionary. Once they are created, both new and existing key entries will append their existing values with the new frequency data collected from the collection. The frequency data is formatted with the document number, followed by the frequency belonging to that token. After all tokens from the collection were added to the index, the process repeats again until the for-loop reaches the last HTML file. Then it begins to write both the unique words it collected and the frequency index to two different text files.

**Part B**

Collection Statistics

* Number of documents indexed: 500
* Vocabulary: 58, 522