James Ruben
Neel Patel
Eddy Grafstein
Nithin Tumma

Project Checkpoint 1

Progress

While the different sides of our project have not yet quite come together seamlessly, we have made a significant amount of progress in getting a lot of the basic functionality to start to emerge.

So far we have figured out how to scrape the Wikipedia corpus by going through a random selection of 10,000 Wikipedia articles and cataloging the set of unique words as well as how many times each of these words appears in each article. Using this massive collection, we are now able to compute the TF-IDF (term frequency-inverse document frequency) of each inputted word to figure out how important the word is to each text. We have also learned how to get the number of queries we want from Google. Based on a received URL, we can then retrieve the HTML and parse out the relevant details. By performing this process iteratively (for now, twice), we can generate the family of keyword subtrees (100 keywords total), which is essentially a tree comprised of the 10 most important keywords to a query and the 10 most important keywords branching off of each of the original keys.

The first main step of our process (the pre-processing step) is to go through the 10,000 wiki articles and store the set of all unique words, along with their computed IDFs, in a huge dictionary. After this step we are able to, for a query, find the wiki page, find the key words by computing the TF of all of the words in the article, and then use the IDF computed in the preprocessing step to get the combined TF-IDF score. Once we

have a list of all of these scores we simply sort them and take the top 10 entries to form the set of keywords for the query. Then we do a Google search on each of the keywords and take the HTML from the top return hit from Google and perform the process again, filling out our 100 keywords. For each of these 100 keywords, we are able to return the top 10 Google URLs and generate a document containing all of these URLs, which will form the basis of information with which to put together our succinct summaries. The hard part for next week will be figuring out how to sort these URLs.

Problems

So far we've had a little bit of trouble downloading things from Wikipedia. Since the website doesn't support scraping very well, we have had to take a lot of unforeseen measures to access the information programmatically. However, we were able to figure out how to spoof a Mozilla user agent to make it look as if the requests were coming from a browser.

Also, the sorting algorithm for the URLs is proving to be more difficult to build than we previously expected, although we have made some significant progress with it since the beginning of the week. In addition, although we have been able to create our global dictionary of words, we are still working on a more efficient way to go through and keep track of each word that has been visited.

Teamwork

James and Neel have been concentrating most of their attention on the pre-processing phase of the project, while Nithin and Eddy have mostly been occupied with the first step of processing the material. This setup has been working pretty well since Neel and Nithin both have more experience with Python as a language so pairing our

more comfortable teammates with our less comfortable has made sure that we can tackle the different problems efficiently. We may be able to improve by assigning ourselves more specific tasks to accomplish individually instead of wasting some time and effort either working on similar problems or relying on the work of other teammates to move forward.

Plan

Our plan for next week consists of the following steps:

· Implement function that finds Jaccard Coefficients of a keyword in a document -- (completed by Tuesday, James)

· Determine the correct operation to sort the set of URLs in terms of their breadth (standard deviation, media Jaccard Coefficients, etc) -- (completed by Wednesday, Nithin)

· Implement the above function to sort the URLs in terms of decreasing breadth (by Thursday, Eddy)

· For things of low breadth and high depth, figure out the focus keyword (by Thursday, Neel)

· Implement the function that will sort these URLs in the correct order. (by Friday, Neel)

Note: So far we have mostly been completing our tasks as teams (James+Neel, Nithin+Eddy)