Erik Gabrielsen

Web Crawler Questions

1. Identify the key properties of a web crawler (see course slides 2/2/2016). Describe in detail how each of these properties is implemented in your code. [20 points]

Robustness – Web Crawler must be immune to spider traps, duplicates, very larger, pages, and dynamic pages, etc. In order to make my web crawler robust, I kept a visited page list that contains all pages that the crawler has already visited, so that the crawler does not visit the same page over and over, eliminating spider traps. I handle duplicates by checking to see if the links out are the already in the link\_list and if so it does not add a duplicate link to the array. The crawler also takes in a max number of pages to read in and defaults to 100 to deal with very large websites and dynamic pages.

Politeness – Web Crawler should only crawl pages in which it is allowed to crawl. My Web crawler reads the robot.txt file first to see which files or directories it is not allowed to visit. Then I save this to a ‘don’t\_go’ array containing files or directories of pages to not visit. I then check each page to see if I have access there before getting the page.

Distributed - The web crawler is able to execute across multiple platforms by ruby versioning (RVM is what I use). As long as you have the same version of ruby running it is able to handle my crawler. Also Mechanize – part of the web crawler framework, allows you to specify what search engine you prefer i.e. Chrome, Firefox, Safari, etc.

Scalable – Need to sub select what the crawler indexes. In my crawler implementation I ignore all non-english characters outside the range a-z and A-Z. This gets rid of random numbers and such that the user probably does not care about. I also downcase all words to get rid of arbitrary duplicates. This way I don’t end up with two different tokens like ‘The’ and ‘the’. I use regex to distinguish these different words.

1. Use your crawler to create a list of all pages in the test data and report all out-going links of the test data. [20 points]

See links.txt

1. Use your crawler to list all broken links in the test data. [10 points]

See broken\_links.txt

1. How many JPG files are included in the test data? [10 points]

3 JPG files (printed to console when run from terminal)

1. Have your crawler save the words from each page of type (.txt, .htm, .html). Make sure that you do not save HTML markup. In this process, give each page a unique document ID. [25 points]

See doc\_id.txt

1. Create a word frequency list from the saved words. Report the 20 most common words with its document frequency. [15 points]

See freq.txt