Henry Warren

CSE 5337

3/18/18

Project 1 – Web Crawler (Extended Write-Up)

The program behavior of my project by file in detail:

* + *master.py*:
    - Entry point and manager of the program.
    - Contains a “master” class that instantiates instances of the other classes and controls the overarching data flow in the program.
    - Master object contains a “run” method that is a program loop that continues indefinitely until the page visit limit parameter N is met or the URL frontier runs out of URLs.
  + *spider.py*:
    - Contains a class that simply fetches the url delegated to it by master, gets the html page source, and passes it to the parser for parsing.
  + *parser.py*:
    - Contains a class that serves a couple functions:
      * First, the parser extracts all values of ‘href’ attributes found in ‘a’ tags in html source, and passes these to the URL filter.
      * Second, if the html source is from a page with a desired file type for text parsing (.htm, .txt, .php, etc) it will grab all plain text from the html source and pass it to the tokenizer for text processing.
  + *tokenizer.py*:
    - The tokenizer serves a couple important functions:
      * First, the tokenizer identifies words in text via the regular expression “*[A-z][^.?!\s]\*[A-z\d]\b”*. (this regex implements the precise definition of a “word” provided in the project requirements).
      * Once we have a list of all words in the page source, the tokenizer “tokenizes” them by casting them to lower case and eliminating stopwords from the set.
      * Remaining tokens are returned to the parser.
  + *indexer.py*:
    - Contains a class that, given a list of tokens, indexes these tokens in a nested dictionary in the form *{docID : {token : tf}}*.
    - The indexer also contains a function for returning the N most frequent terms by either term frequency or document frequency (default mode is document frequency).
  + *pagearchive.py*:
    - Contains a class that archives page contents for the purpose of duplicate detection.
    - Accomplishes dup-detection by taking the md5 hash of a page source and storing this hash in a dictionary with this md5 hash as the key, and the page’s URL as the value; *{hash\_of\_page\_source : url}*.
    - This class contains a function to detect exact duplicates by simply hashing a page’s contents and comparing it to the archive keys. If the hash is not currently a key in the archive, the hash is added and the page is considered unique. Otherwise, a flag is raised for a duplicate.
  + *urlfilter.py*:
    - Contains a class that vets found urls based on multiple factors:
      * Whether or not a link has already been visited (and if it has, how long has it been since it was visited. If it’s been visited recently and therefore is not “fresh”, don’t visit it).
      * If it is an outgoing link (since we need to stay on your website).
      * If it is a bad link (404).
      * If it is an image file.
    - All of these cases are logged and passed back to master. If a url passes this vetting, it is added to the url frontier.
    - The URL filter also casts relative page links to absolute url paths before passing them on to the frontier.
  + *urlfrontier.py*:
    - Class containing a queue that found urls are added to (after being passed through the urlfilter).
    - If the queue runs empty, the program prints its findings and exits.