601.466/666 ASSIGNMENT 4
Web Robots

Due date: Friday May 1 2020 at 11:59 PM

1 Introduction

In this assignment, you will implement a web crawler. We have given you some starter code, and your job is to fill in missing parts. The code uses the built-in urllib.request library to handle URL opening. To read a webpage, it first creates a request:

```
url = "https://cs.jhu.edu/~yarowsky/cs466.html"
res = urllib.request.urlopen(url)
```

If this request is successful, it returns a response variable, and the HTML can be obtained by calling res.read(). More details can be found by reading the library documentation[1].

We have given you a function parse_links, which uses the BeautifulSoup library to parse html and extract (link, title) pairs. We have also given you a crawl function that implements web crawling using a simple queue-based breadth first search. Finally, an extract_information function extracts desired information from the crawled pages. You can run the code with

```
python hw4.py <starting-url>
```

where starting-url is the link from which you want to start crawling.

Running this code will generate several output files containing the links you visited and the information you extracted.

Tasks

1. Implement the get_nonlocal_links function to extract only links which are non-local or not self-referencing.

   A non-local link is a link referencing a URL from some other domain. For example, if we retrieved a page from https://www.cs.jhu.edu, then the following link

   ```
   <a href="https://www.cs.jhu.edu/research/language-speech-processing/">Language amp; Speech Processing</a>
   ```

[1] https://docs.python.org/3/library/urllib.request.html

is local and the link,

<a href="www.ora.com/index.html">Cool book place</a>

is non-local.

A self-referencing link is a link that points to the same page. For example, if we retrieved a page from https://cs.jhu.edu/∼yarowsky/cs466.html, then the following links are self-referencing:

<a href="cs466.htmltest"></a>

<a href="cs466.html"></a>

<a href="../∼yarowsky/cs466.html"</a>

2. Modify the crawl function so that the robot does not visit self-referencing links.
3. Modify the crawl function to accept an argument within_domain, which specifies that the robot should only traverse a single domain. For example, if the starting url was https://cs.jhu.edu/∼yarowsky/cs466.html, the web robot would only visit pages within the cs.jhu.edu domain.
4. Modify the crawl function to take an argument, "wanted_content", which is a list of content types (e.g. postscript content or text/html), and then only visit pages of content types in that list. You can determine the content type with req.headers['Content-Type'].
5. Modify the extract information function to parse through the returned content from all html and plain text web pages using regular expressions to extract and print out the following 3 types of regular format "contact" information found on a web page:

   - All US phone numbers
   - E-mail addresses
   - Addresses of the format "<CITY>, <STATE> <ZIP>". Be prepared for some mild variation such as "San Diego, CA 92122" and "Cambridge, Mass. 02138", but you can pretty much match on any word string of the form "Word, Word 5-digit number".

     The output should be printed as simple (url, type, data) tuples, e.g.
     (http://www.cs.jhu.edu/∼yarowsky, PHONE, 410-516-5372)
     (http://www.cs.jhu.edu/∼yarowsky, EMAIL, yarowsky@jhu.edu)
     (http://www.cs.jhu.edu/∼yarowsky, ADDRESS, Baltimore, MD 21218)

6. Implement parse_links_sorted to rank all links found in a page by some relevancy function of your own design. Incorporate this function into the crawler, so that unvisited links will be sorted based upon your relevancy function to determine where the robot will visit next. The

PriorityQueue class from the built-in queue library may be useful for this part.

In developing your web crawler, please consider the ethical implications which will be discussed in class. We recommend you to first test your crawler on a small, hand-crafted dataset before releasing your crawler into the wild. Ultimately, please only run your crawler on websites within the jhu.edu domain.

Submission

Please submit hw4.py with the above tasks completed to Gradescope by the given deadline.