# Ranking Webpages

Sofia Huang

due 10/20/2022

## 1. Data Collection

**Download the HTML content of the 1000 unique URIs you gathered in HW2 and strip out HTML tags (called ”boilerplate”) so that you are left with the main text content of each webpage.**

First, I made a function to read the .txt file I had saved of unique URIs and store them in a list. Then, I made another function to use the boilerpy3 library to extract the main text content of the URIs. I wrote the main text content into a separate file for each URI. I used hashlib’s MD5 function to hash the URI for the filenames of the text content files. If nothing was extracted or there was an error parsing the HTML, I skipped that URI until I had tried the process for 1000 unique links.

```python
def read_unique_links_file(txt_filename):
    # check if we are in the correct directory
    os.chdir("/Users/sofiahuang/Documents/WM/FALL2022/DATA440")
    summary_filename = os.path.join(os.getcwd(), txt_filename)
    # read the resolved links txt file into a list to be returned
    try:
        summary_file = open(summary_filename, 'r')
        url_data = summary_file.read()
        unique_list = url_data.split("\n")
        summary_file.close()
        print(str(len(unique_list)))
        return unique_list
    except Exception as e:
        print(e)
        return []

def extract_text(uri_list, raw_directory, processed_directory):
    link_index = 1
    num_processed = 0
    extractor = extractors.ArticleExtractor()
    for uri in uri_list:
        if link_index == 1000: break
        try:
            # hash URI to create filename
            filename = hashlib.md5(uri.encode('utf-8')).hexdigest()
            if os.path.exists(os.path.join(processed_directory,
    filename)):
            # file has already been created, go to the next uri
                print('file already exists, skipping uri')
                link_index += 1
                continue
            # request URI
            resp = requests.get(uri, timeout=5)
            raw_html = resp.text
            # pass HTML to Extractor
            content = extractor.get_content(resp.text)
            # check if any text was obtained, if not, skip
            if (content == ''):
                print('nothing extracted from: ' + uri)
                link_index += 1
                continue
            else:
                print('content from: ' + uri)
            # open txt file and write raw text content to it
            raw_html_file = codecs.open(os.path.join(raw_directory, str(filename)), "w", encoding='utf8')
            raw_html_file.write(uri + "\r\n")
            raw_html_file.write(raw_html)
            raw_html_file.close()
            # open txt file and write processed text content to it
            text_content_file = codecs.open(os.path.join(processed_directory, str(filename)), "w", encoding='utf8')
            text_content_file.write(uri + "\r\n")
            text_content_file.write(content)
            text_content_file.close()
            # keep track of how many URI text content obtained
            num_processed+=1
            link_index+=1
        except Exception as e:
            print('error requesting URI or extracting content, skipping to next.')
            print(e)
            link_index += 1
            continue
    print('{} URIs stripped.'.format(num_processed))
```

**Listing 1:** Extracting main text content of each webpage

**Q: How many of your 1000 URIs produced useful text? If that number was less than 1000, did that surprise you?**

776 of the 1000 URIs produced useful text. The rest had either HTML parsing errors or boilerpy3 was unable to obtain the text and produced an empty file. This did not surprise me as boilerpy3 is not a perfect package and will not always work and some HTML might be all boilerplate. This is due to the fact that different domains have different ways of writing their HTML so it is difficult to have one process of stripping the tags when the structure can differ from one URI to another.

## 2. Rank with TF-IDF

**Choose a query term (e.g., "coronavirus") that is not a stop word (e.g., "the"), or super-general (e.g., "web"), or in HTML markup (e.g., "http") that is found in at least 10 of your documents. If the term is present in more than 10 documents, choose any 10 English-language documents from different domains from the result set.**

**As per the example in the Module 06 slides, compute TF-IDF values for the query term in each of the 10 documents and create a table with the TF, IDF, and TF-IDF values, as well as the corresponding URIs. (If you are using LaTeX, you should create a LaTeX table. If you are using Markdown, view the raw version of this file to see how to generate a table.) Rank the URIs in decreasing order by TF-IDF values.**

The formulas I used are as follows:

TF = occurrence in doc / words in doc

IDF = $\log_2$(total docs in corpus / docs with term)

TF-IDF = TF * IDF

To obtain the values needed to calculate the TF-IDF, I used the function `os.listdir(directory)` to convert the folder of processed HTML files into a list that I can iterate through. To pick the 10 URIs to calculate the TF-IDF for, I used only the URIs that had more than 15 instances of the term so I knew the query term was actually relevant to the webpage and to limit the amount of URIs I had to choose from. Then, I chose the first 10 of those with different domains.

For TF, to obtain the 'occurrence in doc' of the query term, I used the Unix command `cat doc_name | grep -c coronavirus` and to find the total 'words in doc', I used the Python function `len(f.read().split())`. This works by reading the file and putting the content into an array, splitting it by the spaces, meaning each index is a word. So taking the length of that array, gives the word count of the file.

For IDF, I used the number Prof. Nwala provided for the corpus size of Google, 35B. And for the number of 'docs with term', I kept track of how many documents had at least 1 instance of the query term while I was computing the number of occurrences of the term in each document. There were 197 documents that contained the query term 'Ukraine'.

| TF-IDF | TF | IDF | URI |
|---|---|---|---|
| 0.515 | 0.0188 | 27.405 | https://consortiumnews.com/2022/10/05/british-intelligence-predicted-ukraine-war-30-years-ago/ |
| 0.457 | 0.017 | 27.405 | https://www.birmingham.ac.uk/research/perspective/divided-ukraine-connolly.aspx |
| 0.411 | 0.015 | 27.405 | https://www.economist.com/by-invitation/2022/06/10/allowing-ukraine-into-the-eu-is-not-the-right-move-for-now-say-luuk-van-middelaar-and-hans-kribbe |
| 0.387 | 0.0141 | 27.405 | https://news.yahoo.com/russian-rockets-slam-ukrainian-city-074329066.html?soc_src=social-sh&soc_trk=tw&tsrc=twtr |
| 0.344 | 0.0126 | 27.405 | https://abcnews.go.com/International/wireStory/kremlin-regions-ukraine-folded-russia-friday-90683980?utm_source=dlvr.it&utm_medium=twitter |
| 0.265 | 0.010 | 27.405 | https://foreignpolicy.com/2022/09/28/russia-ukraine-war-nato-eastern-flank-military-kaliningrad-baltic-finland/ |
| 0.244 | 0.009 | 27.405 | https://en.wikipedia.org/wiki/Allegations_of_genocide_of_Ukrainians_in_the_2022_Russian_invasion_of_Ukraine |
| 0.223 | 0.008 | 27.405 | https://greatpowerrelations.com/game-changers-in-ukraine-crisis/ |
| 0.221 | 0.008 | 27.405 | https://www.cnn.com/europe/live-news/russia-ukraine-war-news-10-06-22#h_a7e50cc7320607f236dc692cc8414cdd |
| 0.198 | 0.007 | 27.405 | https://moderndiplomacy.eu/2018/06/04/how-and-why-the-u-s-government-perpetrated-the-2014-coup-in-ukraine/ |

**Table 1:** 10 Hits for the term "Ukraine", ranked by TF-IDF.

```python
def count_query_term(directory):
    num_docs_w_term = 0
    # list of all files with stripped text content
    html_list = os.listdir(directory)
    os.chdir("/Users/sofiahuang/Documents/WM/FALL2022/DATA440")
    # creating new folder for documents with query terms
    query_directory = os.path.join(os.getcwd(),'query')
    if not os.path.exists(query_directory):
        os.mkdir(query_directory)
    # loop through stripped text content files and
    # see how many of the query term are in each
    for doc in html_list:
        os.chdir(directory)
        output = os.popen('cat ' + doc + ' | grep -c Ukraine').read()
```

```
     # count the number of docs that have the query term
     if int(output) > 0: num_docs_w_term += 1
     # if there is more than 15 instances, copy file to folder created earlier
     if int(output) >= 15:
         f = open(doc, 'r')
         uri = f. readline()
         num_words = len(f.read().split())
         print('Term occurence: {} - Word count: {} - URI: {}'.format(str(output),num_words,uri))
         copyfile(os.path.join(directory,doc), os.path.join(query_directory,doc))
 print('Number of documents with query term: {}'.format(num_docs_w_term))
```

**Listing 2:** Obtaining files containing the query term ”Ukraine”

## References

- StackOverflow - Assign Standard Ouput To A Variable `https://stackoverflow.com/questions/3503879/assign-output-of-os-system-to-a-variable-and-prevent-it-from-being-displayed-on`
- StackOverFlow - Copy File From One Location To Another `https://stackoverflow.com/questions/52851994/copy-a-file-from-one-location-to-another-in-python`
- Python List Files In A Directory `https://www.stackvidhya.com/python-list-files-in-directory/`
- Python MD5 Hash `https://www.studytonight.com/python-howtos/how-to-get-md5-sum-of-a-string-in-python`
- StackOverflow - Insert Line Break in Latex Table Cell `https://stackoverflow.com/questions/3068555/how-to-insert-manual-line-breaks-inside-latex-tables`
- StackOverflow - Read Only First Line in File `https://stackoverflow.com/questions/1904394/read-only-the-first-line-of-a-file`