CSE 7337
Spring 2017
Student Id: 47304053

Somya Singh
Email: somyas@smu.edu

Term Project

Table of Contents

Introduction and KeyPoints

I have split the project into three parts:

- Script 1 (crawler.py) which is responsible for crawling through the website and download the content to local. The Key property of this script are;
    - Uses the Agent-Name: 'SomyaBOTCSE7337' for all its request
    - If the user does not handle http protocol it defaults to http
    - Read robot.txt to look for ALLOWED and DISALLOWED directory
    - It also read robot.txt to look for "crawl-delay" to see if there is any delay. If there no delay then adds 2 sec delay before each request
    - Process the HTTP Response Code
        - If its 404/500: Mark it as Broken Link
        - If get 401/403/503: Stop the script
        - If its 301 then redirect to the new url
    - Near Duplicate detection using technique called "shingling" and Jaccard coefficient of 80%.
    - Handling the navigation restriction only to /~fmoore/.
    - Printing Size of the directory after each download
    - Listing All
        - All the pages visited
        - All Graphics file
        - All PDF
        - All Duplicate Pages
        - All broken url
        - All the files which are downloaded
        - All the links that are outside /~fmoore/ directory
- Script 2 (indexing.py) which read through all the files downloaded by previous script and performing following operation;
    - Eliminating stop word by reading "StopWordlist.txt" that I have kept in the folder
    - Performing Stemming
    - Storing it in a schema which contain following columns;
        - Document ID which is unique per document
        - Title of the page

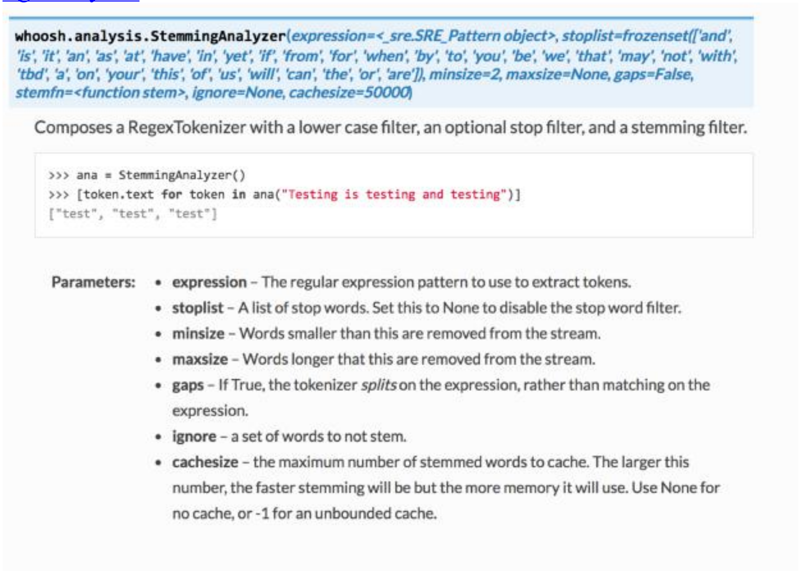
Somya Singh
Email: somyas@smu.edu

  - Path of the URL
  - Content (after stop word and punctuation elimination and stemming) (stored in lowercase)
- Script 3 (outputverification.py) this is used to find the top 20 words in the content and its document frequency.

Key Python Packages

The project is build using python3 with three key packages;

1. Requests (http for human): to handle HTTP request and response. The Package allows program to send an HTTP request to server and process/read the response from the server
2. Beautifulsoup4: to handle parsing of the HTML content. The packages gives the program ability to parse the HTML content.
3. Whoosh: to handle storing and reading data in and from dictionatry. It also provides API for stemming and calculating term/document frequency. The API used for this effort are;
   a. StemmingAnalyzer. (Whoosh API analysis module)
   http://whoosh.readthedocs.io/en/latest/api/analysis.html#whoosh.analysis.StemmingAnalyzer



whoosh.analysis.StemmingAnalyzer(*expression=<_sre.SRE_Pattern object>, stoplist=frozenset(['and', 'is', 'it', 'an', 'as', 'at', 'have', 'in', 'yet', 'if', 'from', 'for', 'when', 'by', 'to', 'you', 'be', 'we', 'that', 'may', 'not', 'with', 'tbd', 'a', 'on', 'your', 'this', 'of', 'us', 'will', 'can', 'the', 'or', 'are']), minsize=2, maxsize=None, gaps=False, stemfn=<function stem>, ignore=None, cachesize=50000*)

Composes a RegexTokenizer with a lower case filter, an optional stop filter, and a stemming filter.

```
>>> ana = StemmingAnalyzer()
>>> [token.text for token in ana("Testing is testing and testing")]
["test", "test", "test"]
```

Parameters:
- **expression** – The regular expression pattern to use to extract tokens.
- **stoplist** – A list of stop words. Set this to None to disable the stop word filter.
- **minsize** – Words smaller than this are removed from the stream.
- **maxsize** – Words longer that this are removed from the stream.
- **gaps** – If True, the tokenizer *splits* on the expression, rather than matching on the expression.
- **ignore** – a set of words to not stem.
- **cachesize** – the maximum number of stemmed words to cache. The larger this number, the faster stemming will be but the more memory it will use. Use None for no cache, or -1 for an unbounded cache.

   b. Reading (Whoosh API rading module)
   http://whoosh.readthedocs.io/en/latest/api/reading.html
   doc_frequency: Which tells user how many document the term appears in (http://whoosh.readthedocs.io/en/latest/api/reading.html#whoosh.reading.IndexReader.doc_frequency)
   most_frequent_terms: This returns top 'number' most frequent term in the given list of content
   (http://whoosh.readthedocs.io/en/latest/api/reading.html#whoosh.reading.IndexReader.most_frequent_terms)

Somya Singh
Email: somyas@smu.edu

Functions used:

1. read_robot: the input is Request URL and output will be if the URL is ALLOWED or DISALLOWED for the userAgent and if any crawl-delay is defined
2. processing_response_code: this function is to handle HTTP response code
3. get_shingles: to obtain shingles value
4. jaccard: get jaccard value
5. check_for_duplicate: This is to make decision of duplication using shingles and jaccard
6. trade_spider: This is main crawling spider function which go through site
7. manipulation_of_url: this is to handle the non-standard robot.txt file location
8. get_tree_size: To get the size of folder where the content is saved.

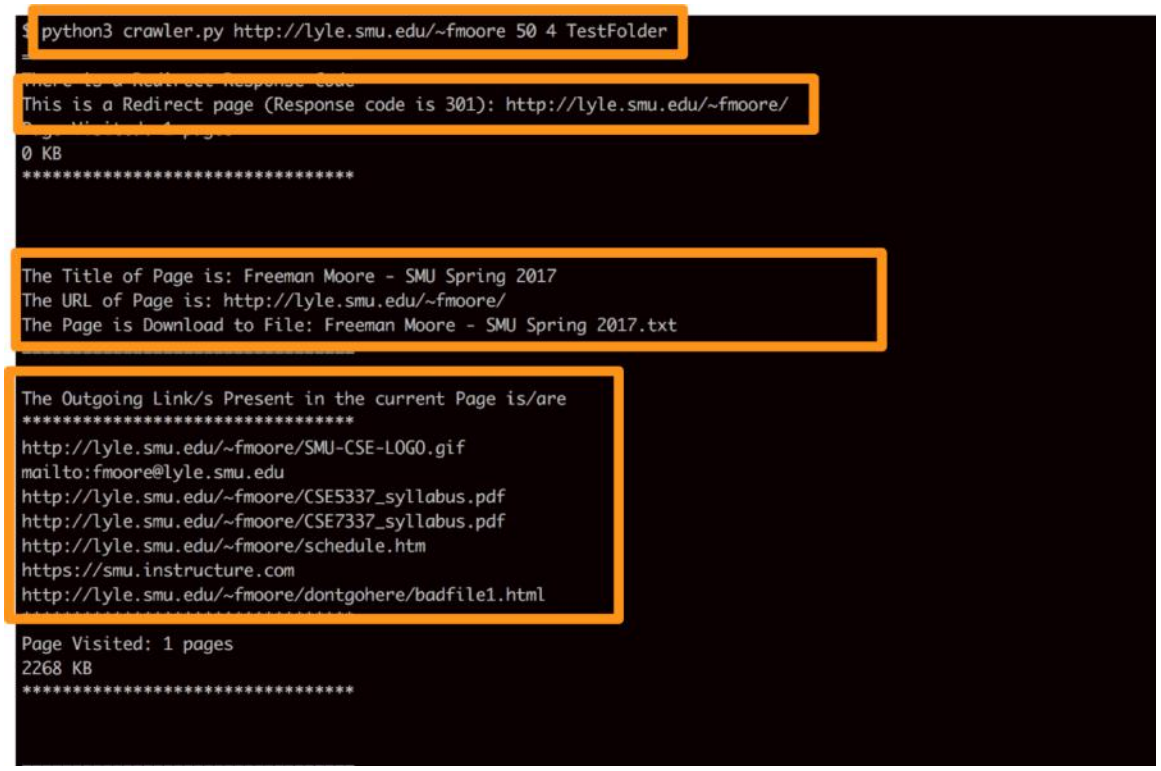
Project Answers

1. Identify the key properties of a web crawler. Describe in detail how each of these properties is implemented in your code.
   The Key feature of crawler implemented in this projects are:
   A. Robustness: The Web contains servers that create spider traps, which are generators of web pages that mislead crawlers into getting stuck fetching an infinite number of pages in a particular domain. Crawlers must be designed to be resilient to such traps. I have achieved this through Page Depth parameter which is taken as an input in program. This is to prevent web crawler to go inside the infinite loop. The crawler is able to handle any of http protocol with any response code. Its able to handle any of redirect response code or error code send by servers without causing any performance impact.
   B. Politeness: Web servers have both implicit and explicit policies regulating the rate at which a crawler can visit them. These politeness policies must be respected. We obtained this by first checking the robots.txt to check what webpages can be crawled and crawl delay. If no crawl delay assigned then its implicitly define delay of 2 sec between every successive request.
   C. Extensible: The crawler architecture be modular. The script is utilizing the functions to perform all actions making it easier to modulize and customize as needed

2. Use your crawler to list the URL of all pages in the test data and report all out-going links of the test data. List Title on each page

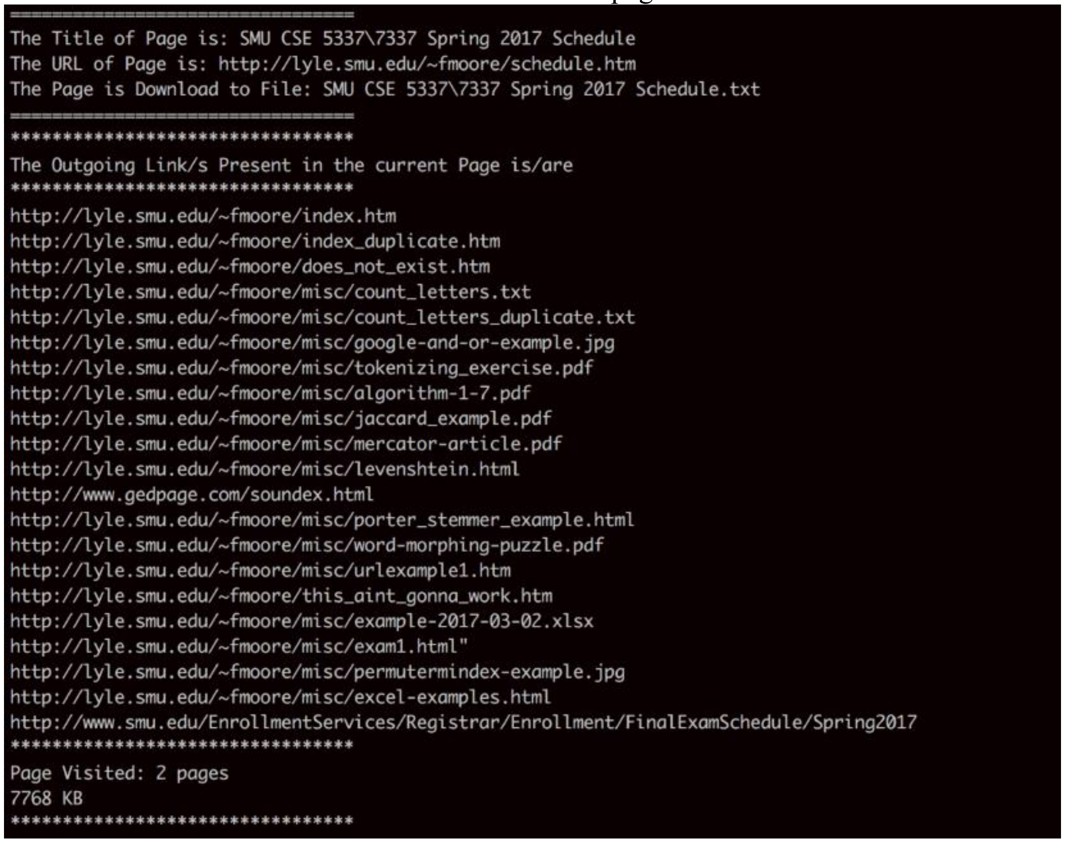
   The screenshot 1 shows the redirect of http://lyle.smu.edu/~fmoore to http://lyle.smu.edu/~fmoore/ followed by all outgoing links. For each page visit Its shows following
   - Title of page
   - URL of page
   - File where it is saved
   - All the outgoing links.

Somya Singh
Email: somyas@smu.edu



```
$ python3 crawler.py http://lyle.smu.edu/~fmoore 50 4 TestFolder

There is a Redirect Response Code
This is a Redirect page (Response code is 301): http://lyle.smu.edu/~fmoore/
Page Visited: 1 pages
0 KB
*********************************


The Title of Page is: Freeman Moore - SMU Spring 2017
The URL of Page is: http://lyle.smu.edu/~fmoore/
The Page is Download to File: Freeman Moore - SMU Spring 2017.txt
=================================

The Outgoing Link/s Present in the current Page is/are
*********************************
http://lyle.smu.edu/~fmoore/SMU-CSE-LOGO.gif
mailto:fmoore@lyle.smu.edu
http://lyle.smu.edu/~fmoore/CSE5337_syllabus.pdf
http://lyle.smu.edu/~fmoore/CSE7337_syllabus.pdf
http://lyle.smu.edu/~fmoore/schedule.htm
https://smu.instructure.com
http://lyle.smu.edu/~fmoore/dontgohere/badfile1.html
*********************************
Page Visited: 1 pages
2268 KB
*********************************


=================================
```

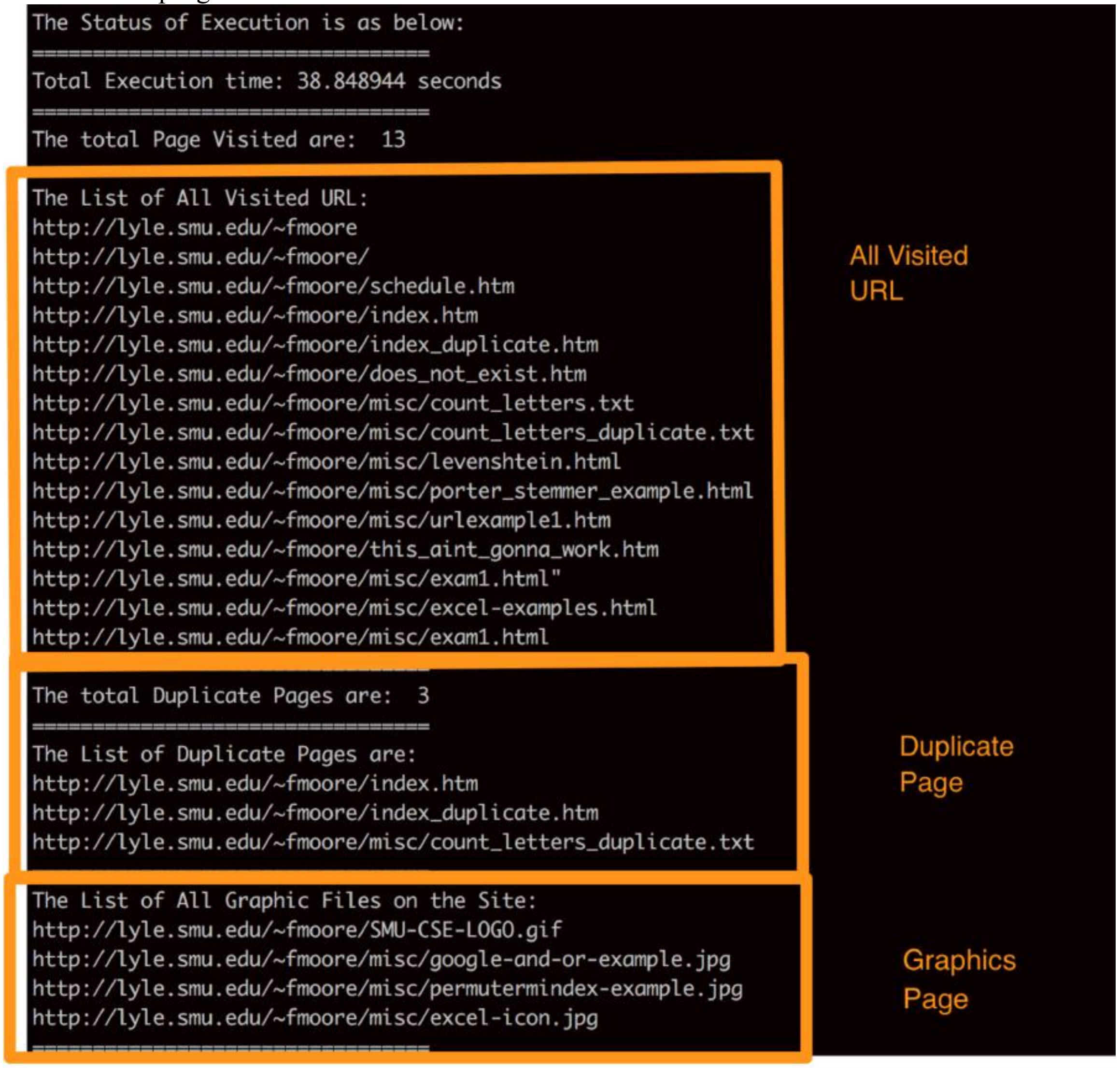
The below screenshot it shows list of all url on that page.



```
=================================
The Title of Page is: SMU CSE 5337\7337 Spring 2017 Schedule
The URL of Page is: http://lyle.smu.edu/~fmoore/schedule.htm
The Page is Download to File: SMU CSE 5337\7337 Spring 2017 Schedule.txt
=================================
*********************************
The Outgoing Link/s Present in the current Page is/are
*********************************
http://lyle.smu.edu/~fmoore/index.htm
http://lyle.smu.edu/~fmoore/index_duplicate.htm
http://lyle.smu.edu/~fmoore/does_not_exist.htm
http://lyle.smu.edu/~fmoore/misc/count_letters.txt
http://lyle.smu.edu/~fmoore/misc/count_letters_duplicate.txt
http://lyle.smu.edu/~fmoore/misc/google-and-or-example.jpg
http://lyle.smu.edu/~fmoore/misc/tokenizing_exercise.pdf
http://lyle.smu.edu/~fmoore/misc/algorithm-1-7.pdf
http://lyle.smu.edu/~fmoore/misc/jaccard_example.pdf
http://lyle.smu.edu/~fmoore/misc/mercator-article.pdf
http://lyle.smu.edu/~fmoore/misc/levenshtein.html
http://www.gedpage.com/soundex.html
http://lyle.smu.edu/~fmoore/misc/porter_stemmer_example.html
http://lyle.smu.edu/~fmoore/misc/word-morphing-puzzle.pdf
http://lyle.smu.edu/~fmoore/misc/urlexample1.htm
http://lyle.smu.edu/~fmoore/this_aint_gonna_work.htm
http://lyle.smu.edu/~fmoore/misc/example-2017-03-02.xlsx
http://lyle.smu.edu/~fmoore/misc/exam1.html"
http://lyle.smu.edu/~fmoore/misc/permutermindex-example.jpg
http://lyle.smu.edu/~fmoore/misc/excel-examples.html
http://www.smu.edu/EnrollmentServices/Registrar/Enrollment/FinalExamSchedule/Spring2017
*********************************
Page Visited: 2 pages
7768 KB
*********************************
```

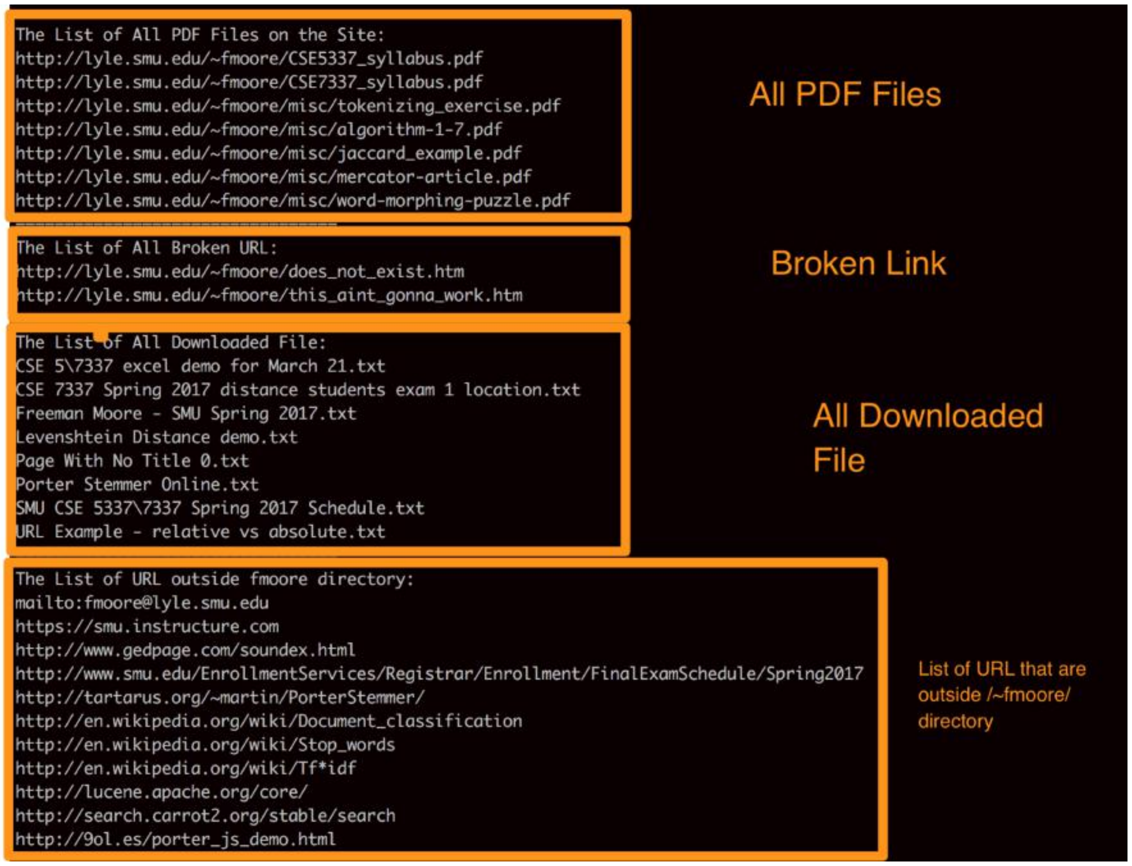
Somya Singh
Email: somyas@smu.edu

At the end of program it list all the url



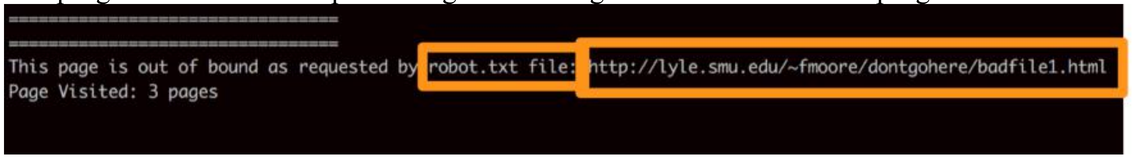
```
The Status of Execution is as below:
==================================
Total Execution time: 38.848944 seconds
==================================
The total Page Visited are:  13
The List of All Visited URL:
http://lyle.smu.edu/~fmoore
http://lyle.smu.edu/~fmoore/
http://lyle.smu.edu/~fmoore/schedule.htm
http://lyle.smu.edu/~fmoore/index.htm
http://lyle.smu.edu/~fmoore/index_duplicate.htm
http://lyle.smu.edu/~fmoore/does_not_exist.htm
http://lyle.smu.edu/~fmoore/misc/count_letters.txt
http://lyle.smu.edu/~fmoore/misc/count_letters_duplicate.txt
http://lyle.smu.edu/~fmoore/misc/levenshtein.html
http://lyle.smu.edu/~fmoore/misc/porter_stemmer_example.html
http://lyle.smu.edu/~fmoore/misc/urlexample1.htm
http://lyle.smu.edu/~fmoore/this_aint_gonna_work.htm
http://lyle.smu.edu/~fmoore/misc/exam1.html"
http://lyle.smu.edu/~fmoore/misc/excel-examples.html
http://lyle.smu.edu/~fmoore/misc/exam1.html
The total Duplicate Pages are:  3
==================================
The List of Duplicate Pages are:
http://lyle.smu.edu/~fmoore/index.htm
http://lyle.smu.edu/~fmoore/index_duplicate.htm
http://lyle.smu.edu/~fmoore/misc/count_letters_duplicate.txt
The List of All Graphic Files on the Site:
http://lyle.smu.edu/~fmoore/SMU-CSE-LOGO.gif
http://lyle.smu.edu/~fmoore/misc/google-and-or-example.jpg
http://lyle.smu.edu/~fmoore/misc/permutermindex-example.jpg
http://lyle.smu.edu/~fmoore/misc/excel-icon.jpg
==================================
```

Somya Singh
Email: somyas@smu.edu



```
The List of All PDF Files on the Site:
http://lyle.smu.edu/~fmoore/CSE5337_syllabus.pdf
http://lyle.smu.edu/~fmoore/CSE7337_syllabus.pdf
http://lyle.smu.edu/~fmoore/misc/tokenizing_exercise.pdf
http://lyle.smu.edu/~fmoore/misc/algorithm-1-7.pdf
http://lyle.smu.edu/~fmoore/misc/jaccard_example.pdf
http://lyle.smu.edu/~fmoore/misc/mercator-article.pdf
http://lyle.smu.edu/~fmoore/misc/word-morphing-puzzle.pdf
===================================
The List of All Broken URL:
http://lyle.smu.edu/~fmoore/does_not_exist.htm
http://lyle.smu.edu/~fmoore/this_aint_gonna_work.htm
The List of All Downloaded File:
CSE 5\7337 excel demo for March 21.txt
CSE 7337 Spring 2017 distance students exam 1 location.txt
Freeman Moore - SMU Spring 2017.txt
Levenshtein Distance demo.txt
Page With No Title 0.txt
Porter Stemmer Online.txt
SMU CSE 5337\7337 Spring 2017 Schedule.txt
URL Example - relative vs absolute.txt
The List of URL outside fmoore directory:
mailto:fmoore@lyle.smu.edu
https://smu.instructure.com
http://www.gedpage.com/soundex.html
http://www.smu.edu/EnrollmentServices/Registrar/Enrollment/FinalExamSchedule/Spring2017
http://tartarus.org/~martin/PorterStemmer/
http://en.wikipedia.org/wiki/Document_classification
http://en.wikipedia.org/wiki/Stop_words
http://en.wikipedia.org/wiki/Tf*idf
http://lucene.apache.org/core/
http://search.carrot2.org/stable/search
http://9ol.es/porter_js_demo.html
```

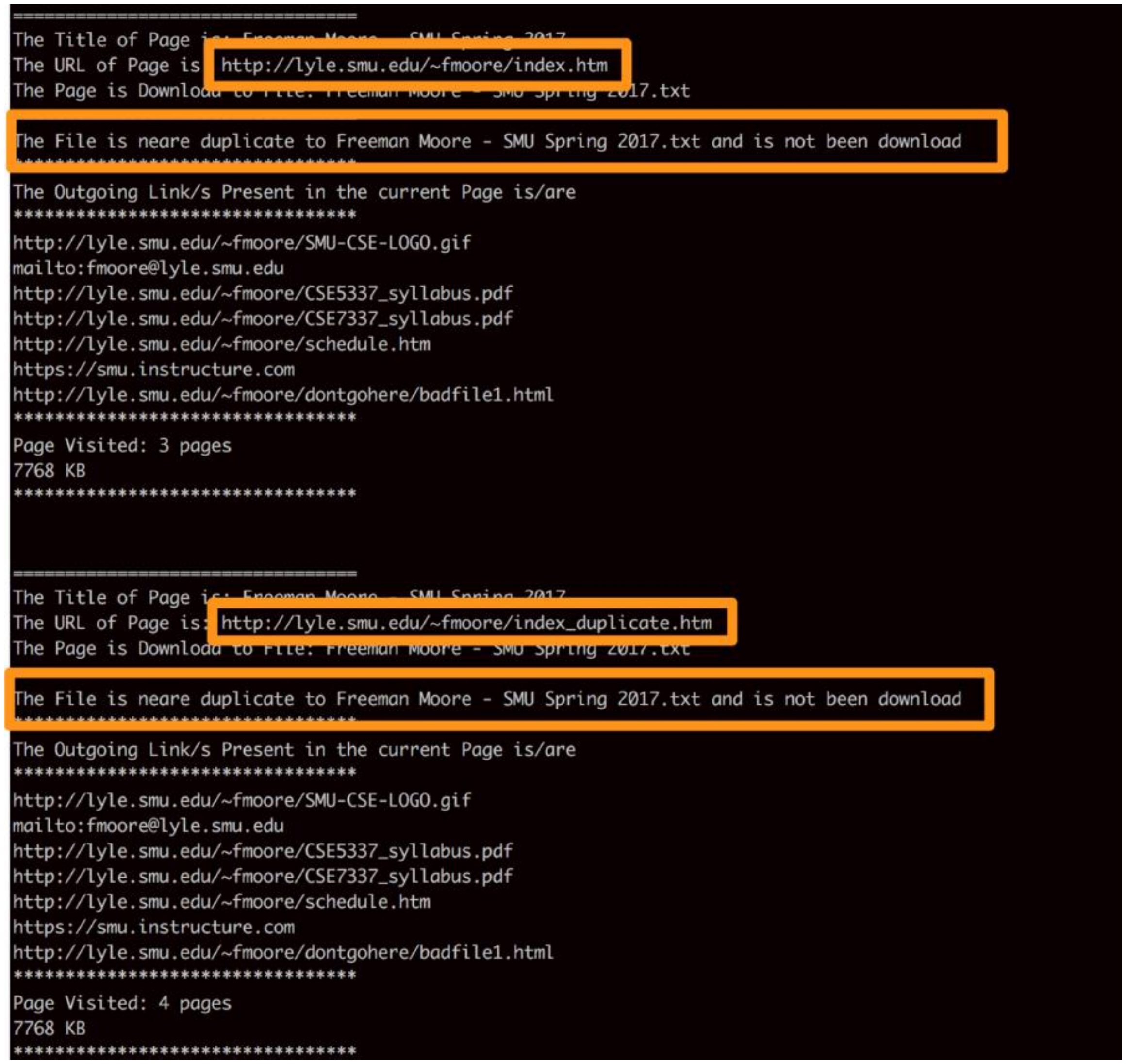
The program is also able to preventing crawler to go to a DISALLOWED program.



```
===================================
===================================
This page is out of bound as requested by robot.txt file: http://lyle.smu.edu/~fmoore/dontgohere/badfile1.html
Page Visited: 3 pages
```

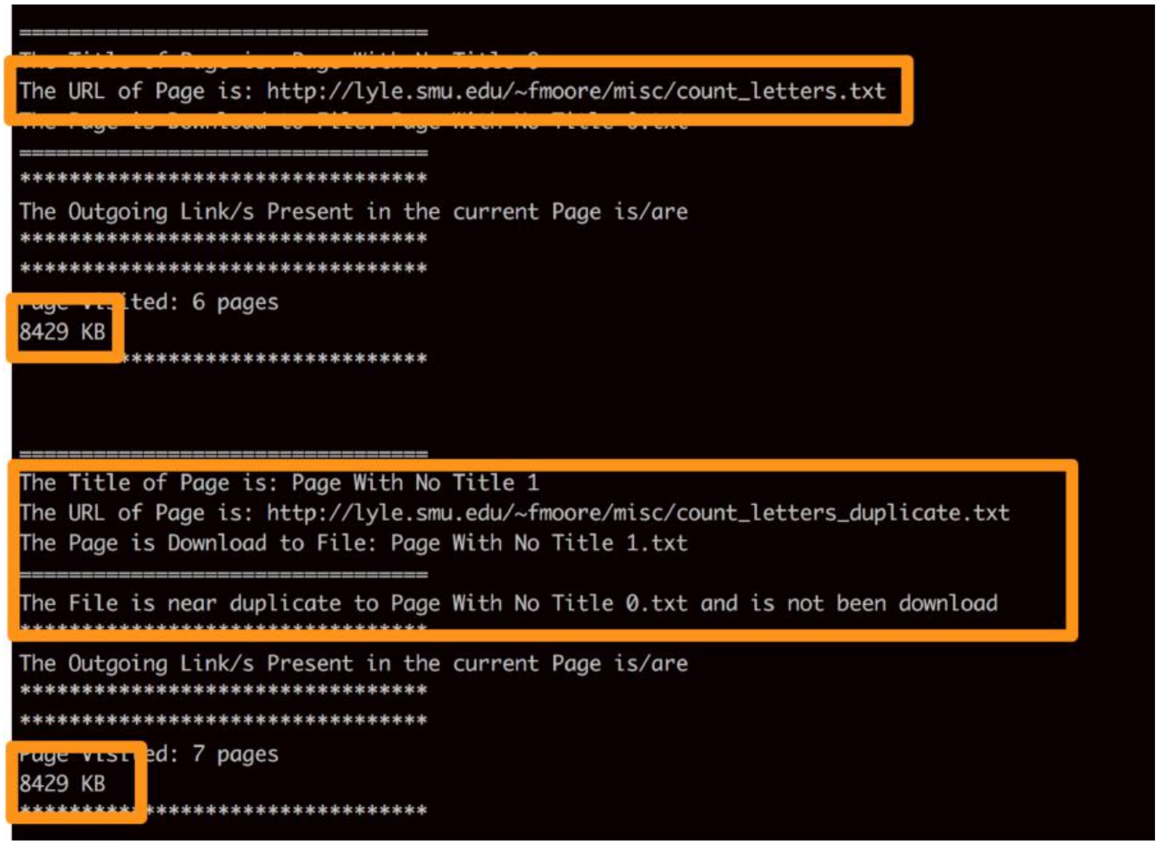
3. Implement duplicate detection, and report if any URLs refer to already seen content.
   I am performing near duplicate detection using shingling (shingle size 5) and calculating Jaccard coefficient .
   (threshold > 80(duplicate data then discard))
   Every Page is checked with all other pages downloaded to make sure the same content is not there.
   Also to handle the duplicate URL I am using an array "visited" which keep track of all URL that are already visited.

Somya Singh
Email: somyas@smu.edu



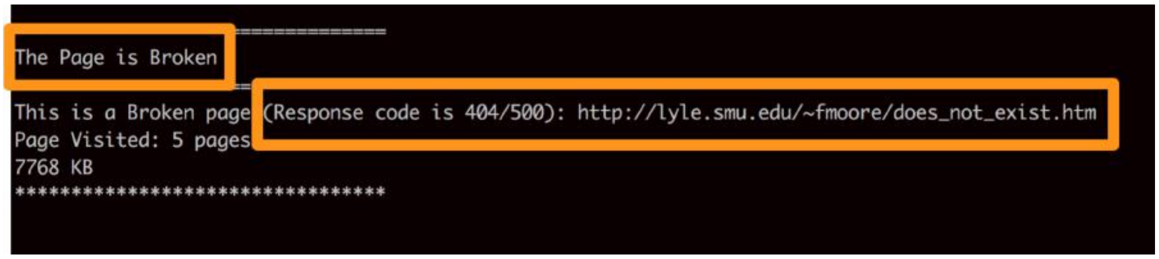
```
=================================
The Title of Page is: Freeman Moore - SMU Spring 2017
The URL of Page is: http://lyle.smu.edu/~fmoore/index.htm
The Page is Download to File: Freeman Moore - SMU Spring 2017.txt

The File is neare duplicate to Freeman Moore - SMU Spring 2017.txt and is not been download
*********************************
The Outgoing Link/s Present in the current Page is/are
*********************************
http://lyle.smu.edu/~fmoore/SMU-CSE-LOGO.gif
mailto:fmoore@lyle.smu.edu
http://lyle.smu.edu/~fmoore/CSE5337_syllabus.pdf
http://lyle.smu.edu/~fmoore/CSE7337_syllabus.pdf
http://lyle.smu.edu/~fmoore/schedule.htm
https://smu.instructure.com
http://lyle.smu.edu/~fmoore/dontgohere/badfile1.html
*********************************
Page Visited: 3 pages
7768 KB
*********************************


=================================
The Title of Page is: Freeman Moore - SMU Spring 2017
The URL of Page is: http://lyle.smu.edu/~fmoore/index_duplicate.htm
The Page is Download to File: Freeman Moore - SMU Spring 2017.txt

The File is neare duplicate to Freeman Moore - SMU Spring 2017.txt and is not been download
*********************************
The Outgoing Link/s Present in the current Page is/are
*********************************
http://lyle.smu.edu/~fmoore/SMU-CSE-LOGO.gif
mailto:fmoore@lyle.smu.edu
http://lyle.smu.edu/~fmoore/CSE5337_syllabus.pdf
http://lyle.smu.edu/~fmoore/CSE7337_syllabus.pdf
http://lyle.smu.edu/~fmoore/schedule.htm
https://smu.instructure.com
http://lyle.smu.edu/~fmoore/dontgohere/badfile1.html
*********************************
Page Visited: 4 pages
7768 KB
*********************************
```

Somya Singh
Email: somyas@smu.edu



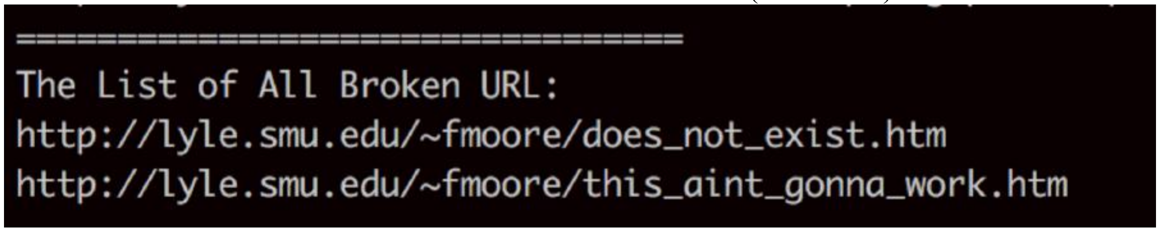
```
=================================
The Title of Page is: Page With No Title 0
The URL of Page is: http://lyle.smu.edu/~fmoore/misc/count_letters.txt
The Page is Download to File: Page With No Title 0.txt
=================================
*********************************
The Outgoing Link/s Present in the current Page is/are
*********************************
*********************************
Page Visited: 6 pages
8429 KB
*********************************


=================================
The Title of Page is: Page With No Title 1
The URL of Page is: http://lyle.smu.edu/~fmoore/misc/count_letters_duplicate.txt
The Page is Download to File: Page With No Title 1.txt
=================================
The File is near duplicate to Page With No Title 0.txt and is not been download
*********************************
The Outgoing Link/s Present in the current Page is/are
*********************************
*********************************
Page Visited: 7 pages
8429 KB
*********************************
```

4. Use your crawler to list all broken links within the test data
The crawler read the Http response code to make the necessary steps. If any page response with HTTP response code of 404 the crawler reports it as broken page and move to next url.



```
=================================
The Page is Broken
=================================
This is a Broken page (Response code is 404/500): http://lyle.smu.edu/~fmoore/does_not_exist.htm
Page Visited: 5 pages
7768 KB
*********************************
```

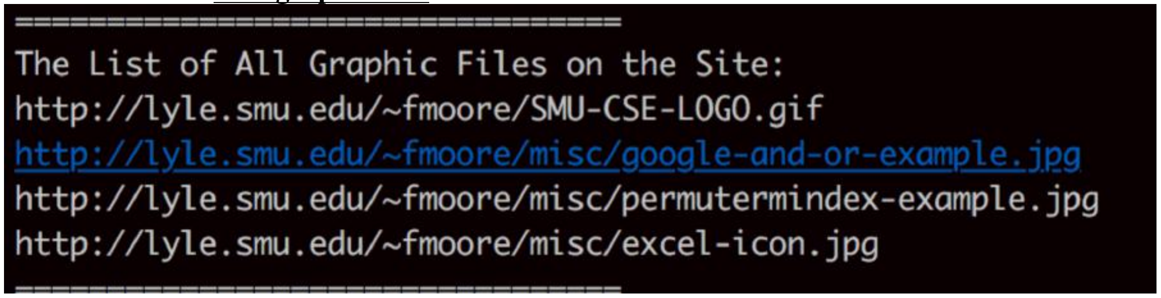
At the end I list all broken links. In total I see two broken links (HTTP 404)



```
=================================
The List of All Broken URL:
http://lyle.smu.edu/~fmoore/does_not_exist.htm
http://lyle.smu.edu/~fmoore/this_aint_gonna_work.htm
```

Somya Singh
Email: somyas@smu.edu

5. How many graphic files are included in the test data?
There are total of **four graphic files** that I am able to see in total.



```
==================================
The List of All Graphic Files on the Site:
http://lyle.smu.edu/~fmoore/SMU-CSE-LOGO.gif
http://lyle.smu.edu/~fmoore/misc/google-and-or-example.jpg
http://lyle.smu.edu/~fmoore/misc/permutermindex-example.jpg
http://lyle.smu.edu/~fmoore/misc/excel-icon.jpg
==================================
```

6. Have your crawler save the words from each page of type (.txt, .htm, .html). Make sure that you do not save HTML markup. Explain your definition of "word". In this process, give each page a unique document ID. Implement Stemming

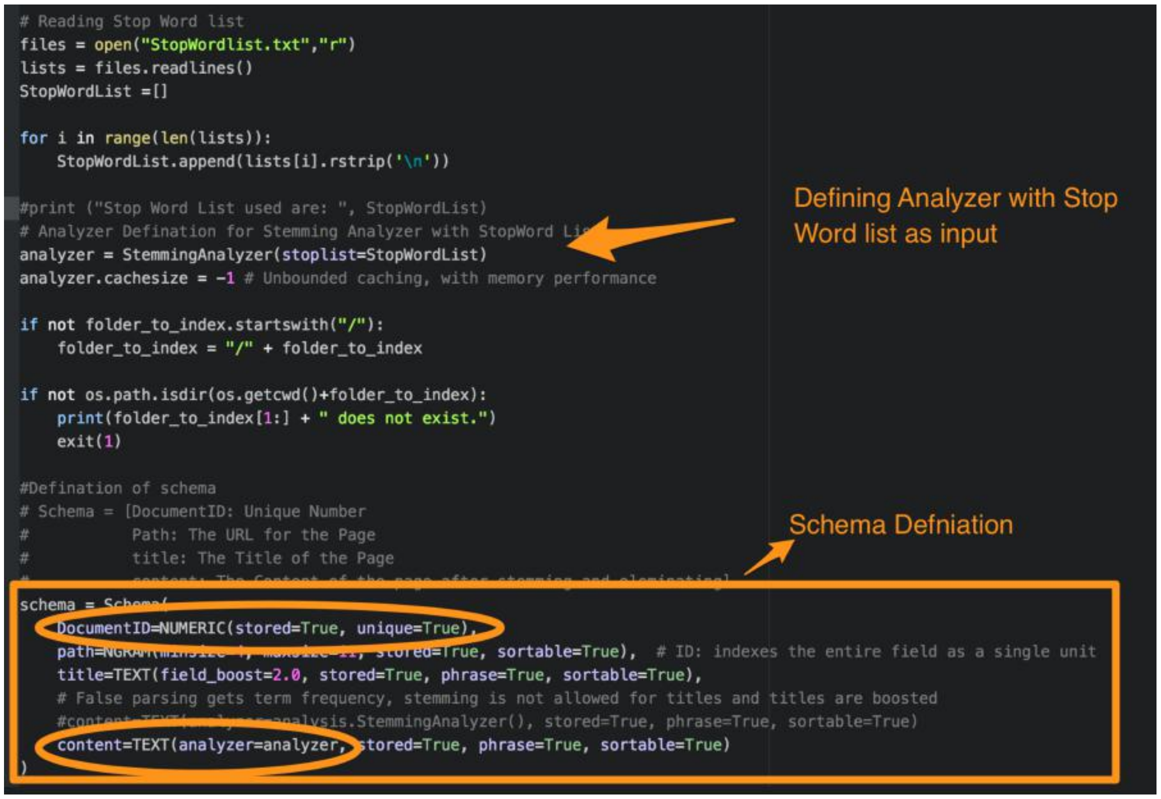
As an input to crawler.py user inputs

- the url to be crawled,
- number of pages user wants,
- page depth and
- output folder where user wants to download the content.

Test folder is the place where all the files identified with their title are saved by the web crawler after performing the crawling. The Test Folder is used by Indexing.py as an input to do indexing. **First it perform stop word elimination by reading StopWordlist.txt** present inside Crawler folder. Then Stemming using Whoosh StemmingAnalyzer it stemmed the token. After performing Normalization, stemmed words are stored in schema. Schema contain following field with

- **DocumentID - Unique document id for each file in Test Folder**
- content field - Processed text after stop word elimination and stemmed words
- path field - Url
- Title field – title and unique Doc Id for each file in Test Folder .

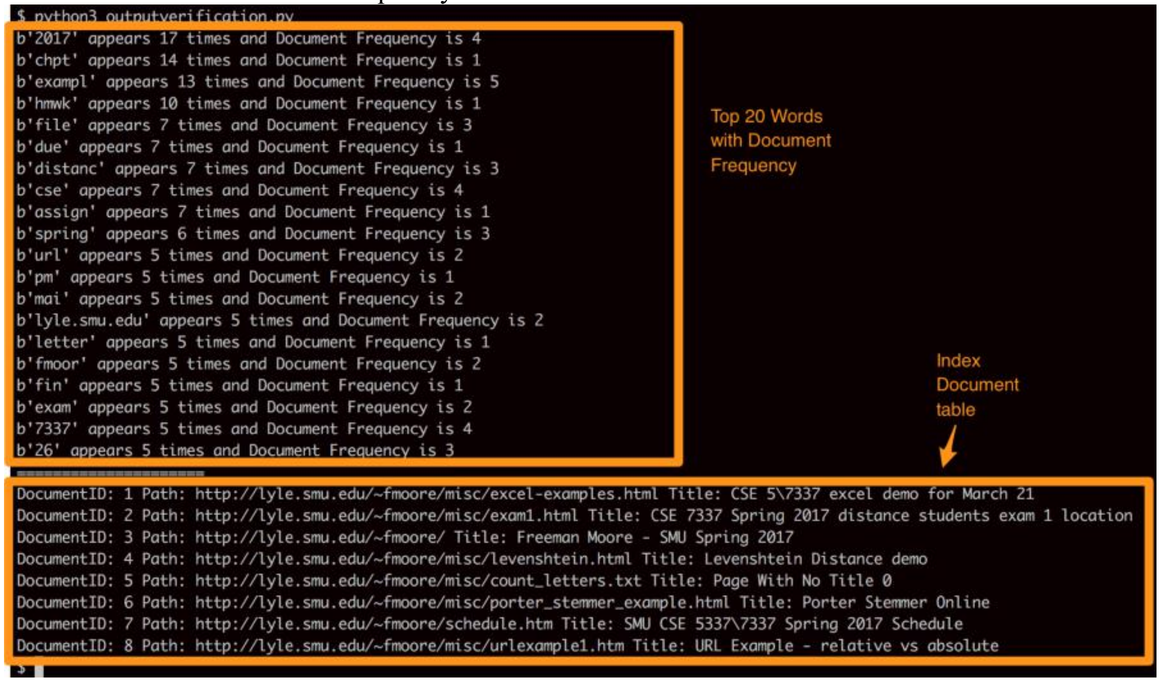
In this Word definition is the token after performing normalization (stop word elimination, punctuation elemenation and stemming) and the stemmed word (include both numbers and words) are obtained and stored in the schema under "content" column in lowercase to allow insensitive searching.



```
# Reading Stop Word list
files = open("StopWordlist.txt","r")
lists = files.readlines()
StopWordList =[]

for i in range(len(lists)):
    StopWordList.append(lists[i].rstrip('\n'))

#print ("Stop Word List used are: ", StopWordList)
# Analyzer Defination for Stemming Analyzer with StopWord Li
analyzer = StemmingAnalyzer(stoplist=StopWordList)
analyzer.cachesize = -1 # Unbounded caching, with memory performance

if not folder_to_index.startswith("/"):
    folder_to_index = "/" + folder_to_index

if not os.path.isdir(os.getcwd()+folder_to_index):
    print(folder_to_index[1:] + " does not exist.")
    exit(1)

#Defination of schema
# Schema = [DocumentID: Unique Number
#           Path: The URL for the Page
#           title: The Title of the Page
#           content: The Content of the page after stemming and eleminating]
schema = Schema(
    DocumentID=NUMERIC(stored=True, unique=True),
    path=NGRAM(minsize=4, maxsize=11, stored=True, sortable=True),  # ID: indexes the entire field as a single unit
    title=TEXT(field_boost=2.0, stored=True, phrase=True, sortable=True),
    # False parsing gets term frequency, stemming is not allowed for titles and titles are boosted
    #content=TEXT(analyzer=analysis.StemmingAnalyzer(), stored=True, phrase=True, sortable=True)
    content=TEXT(analyzer=analyzer, stored=True, phrase=True, sortable=True)
)
```

7. Report the 20 most common words with its document frequency. [15 points] **CSE7337**: words or stemmed words?

   Once indexing is done and all documents are stored in stemmed format below are my top 20 words with its document frequency



```
$ python3 outputverification.py
b'2017' appears 17 times and Document Frequency is 4
b'chpt' appears 14 times and Document Frequency is 1
b'exampl' appears 13 times and Document Frequency is 5
b'hmwk' appears 10 times and Document Frequency is 1
b'file' appears 7 times and Document Frequency is 3
b'due' appears 7 times and Document Frequency is 1
b'distanc' appears 7 times and Document Frequency is 3
b'cse' appears 7 times and Document Frequency is 4
b'assign' appears 7 times and Document Frequency is 1
b'spring' appears 6 times and Document Frequency is 3
b'url' appears 5 times and Document Frequency is 2
b'pm' appears 5 times and Document Frequency is 1
b'mai' appears 5 times and Document Frequency is 2
b'lyle.smu.edu' appears 5 times and Document Frequency is 2
b'letter' appears 5 times and Document Frequency is 1
b'fmoor' appears 5 times and Document Frequency is 2
b'fin' appears 5 times and Document Frequency is 1
b'exam' appears 5 times and Document Frequency is 2
b'7337' appears 5 times and Document Frequency is 4
b'26' appears 5 times and Document Frequency is 3
=====================
DocumentID: 1 Path: http://lyle.smu.edu/~fmoore/misc/excel-examples.html Title: CSE 5\7337 excel demo for March 21
DocumentID: 2 Path: http://lyle.smu.edu/~fmoore/misc/exam1.html Title: CSE 7337 Spring 2017 distance students exam 1 location
DocumentID: 3 Path: http://lyle.smu.edu/~fmoore/ Title: Freeman Moore - SMU Spring 2017
DocumentID: 4 Path: http://lyle.smu.edu/~fmoore/misc/levenshtein.html Title: Levenshtein Distance demo
DocumentID: 5 Path: http://lyle.smu.edu/~fmoore/misc/count_letters.txt Title: Page With No Title 0
DocumentID: 6 Path: http://lyle.smu.edu/~fmoore/misc/porter_stemmer_example.html Title: Porter Stemmer Online
DocumentID: 7 Path: http://lyle.smu.edu/~fmoore/schedule.htm Title: SMU CSE 5337\7337 Spring 2017 Schedule
DocumentID: 8 Path: http://lyle.smu.edu/~fmoore/misc/urlexample1.htm Title: URL Example - relative vs absolute
$
```

Somya Singh
Email: somyas@smu.edu

Reference:

1. https://github.com/divyanshch/Search-Engine-and-Crawler/blob/master/Crawler/crawler.py
2. https://nlp.stanford.edu/IR-book/information-retrieval-book.html
3. https://github.com/Smerity/pubcrawl/blob/master/crawler.py
4. http://www.netinstructions.com/how-to-make-a-simple-web-crawler-in-java/
5. https://github.com/yasserg/crawler4j
6. http://docs.python-requests.org/en/master/user/quickstart/
7. https://www.crummy.com/software/BeautifulSoup/
8. http://whoosh.readthedocs.io/en/latest/index.html
9. https://blog.mafr.de/2011/01/06/near-duplicate-detection/