CMSC 671 : Information Retrieval

Homework-I

Guide: Prof. Pearce Anurag Pawar (LA22171)

Goal: The goal of the assignment is to compare the two approaches used for tokenizing and down casing all the words in a collection of HTML documents.

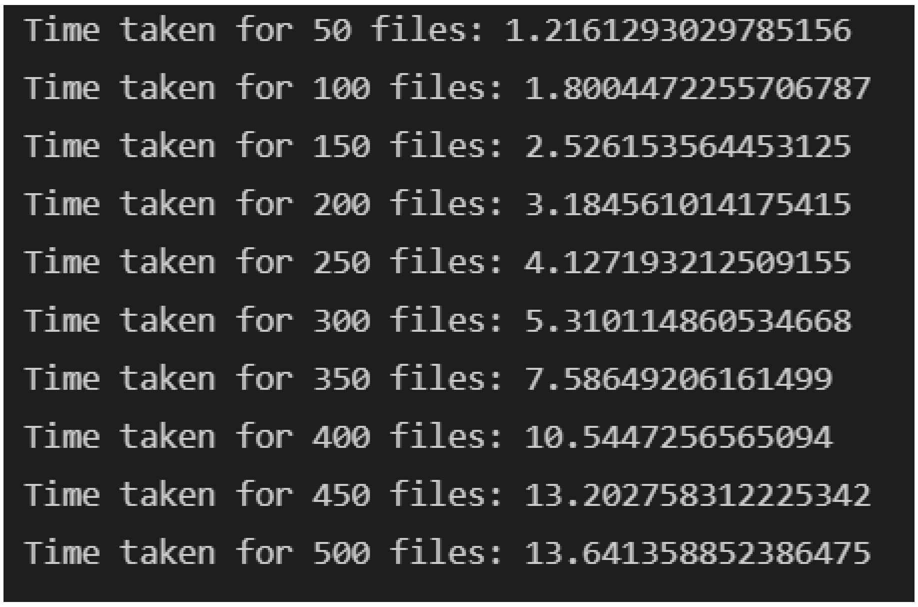
- **Part I : Parsing the HTML document**
- A built-in Python library called '*BeautifulSoap (Version: 4.9.2)*' is used to parse the HTML document.
- It scans complete document from top to down and ignores all the tags used in HTML and appends all the data in a string.
- The mentioned library can also be used for XML document.
- Function used in the code: *def html_cleaner(<location of the file>)*
- **Bottleneck:** As *BeautifulSoap* ignores the tags and append the characters onto a result string, few words are mixed because of this.
- **Example:** Input: line1<br>line2, Output: line1line2.
- **Part II : Cleaning the parsed string**
- To clean the parsed words, a built-in Python library called *RE (Regular expressions)* is used.
- Regular Expression is a search pattern made up of a chain of characters.
- To see if a string contains the supplied search pattern, *RegEx* has been utilized.
- With the help of this library all the special characters have been ignored.
- The encoding parameter is set to *'unicode_encoding'*
- After this, all the upper- case characters are turned to lower-case characters with the help of *lower()* function.

- **Part III - Tokenizing the words from the cleaned document**
- To tokenize the words, a built-python library called 'nltk' is used.
- The 'nltk' library is developed for natural language processing.
- Function used: *nltk.tokenize.word_tokenize(input_text, language='english', preserve_line=False)*
- The *language* parameter in above function is newly developed tokenizing parameter for English language and *preserve_lines* work as a flag to decide whether to sentence tokenize the text or not.

- **Part IV – Writing the output files**
- There are two types of output files generated, one is to represent the data of complete 503 files and another is set of 503 files with their individual output.
- To write the output file, Python OS module is used.

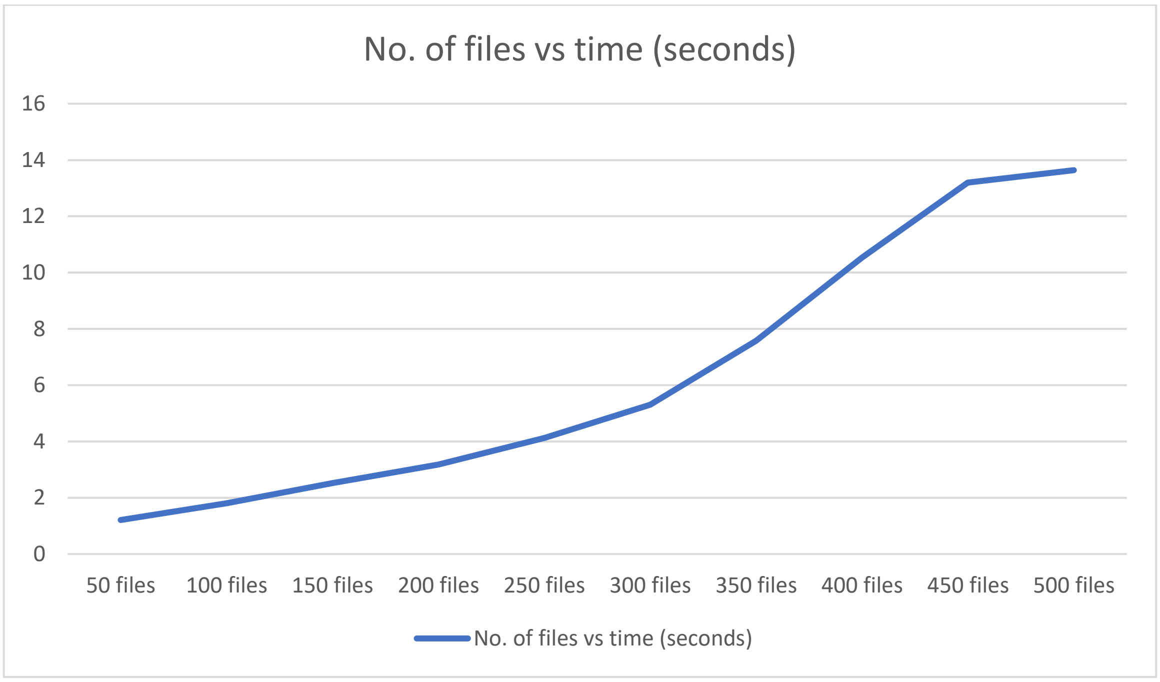
- **Part V: Performance analysis**

- Below is the screenshot depicting the time (in seconds) required to process particular number files.



```
Time taken for 50 files: 1.2161293029785156
Time taken for 100 files: 1.8004472255706787
Time taken for 150 files: 2.526153564453125
Time taken for 200 files: 3.184561014175415
Time taken for 250 files: 4.127193212509155
Time taken for 300 files: 5.310114860534668
Time taken for 350 files: 7.58649206161499
Time taken for 400 files: 10.5447256565094
Time taken for 450 files: 13.202758312225342
Time taken for 500 files: 13.641358852386475
```

- Below is a graph representing the above data.



- There are three FOR loops used in the code, complexity analysis as follows:

1. First FOR loop iterates through all the 503 files and reads them i.e., O(no. of files).
2. Second FOR loop is iterating through each one of the 503 files and cleaning and tokenizing the words i.e., O(no. of words).
3. Third FOR loop is again going through all the words and pushing them into a result string i.e. ,O(no. of words).

Part VI: Comparison with other approach (Compared with Bhushan Mahajan's approach)

- He has performed all the operations in Python with the help of build-in libraries.
- To parse the HTML document, he has used a built-in library called ‘*html2text’*.
- We have used same approach to clean the parsed string i.e., Regular Expressions.
- A GenSim is used to tokenize the words.
- Bhushan’s performance table as follows:

Number of documents processed	CPU execution time (seconds)
53	0.4519
103	1.2247
153	1.6595
203	2.1807
253	2.5102
303	2.9911
353	3.3196
403	3.8139
453	4.3097
503	4.6922

- **Steps to execute the program**

- Install the required libraries from the requirement.txt file.
- *python3 -m pip install -r requirement.txt*
- Run program: Python3 <file_name>.

- **References**

- https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- https://docs.python.org/3/library/re.html
- https://www.w3schools.com/python/python_regex.asp
- https://www.nltk.org/api/nltk.tokenize.html