**CMSC 476 Information Retrieval**

**Phase 1**

**Due: 2/19/2021 @ 11:59 pm**

**By: April Zhu**

**Objectives**

The objectives of this homework are to compare two approaches to tokenize and downcase all words in a collection of HTML documents.

**My Approach**

Run: time python cmsc476\_p1.py /Users/haojunzhu/desktop/p1\_files /Users/haojunzhu/desktop/tokenFiles

I implemented the program in Python, and used tokenizer which is a function nltk.word\_tokenize (text) in NLTK library. The tokenizer splits tokens based on white space and punctuation, it doesn’t handle the HTML5 Entity Names well, such as á which is &aacute, I found 3288 aacute words in the files.

First, I used regular expression operations to remove tags in html files, and then change some common abbreviations such as ’t ’ve ’ll ’d to ‘not’, ‘have’, ‘will’, ‘would’, next filtered special symbols such as double quotes, single quotes, periods, commas and extra spaces. I didn’t add any extra operation to handle the numbers, but the tokenizer removed all the numbers by default.

Next, store all the padded text in a list, and write to the correspond directory to correspond file.

Lastly, used dictionary structure to store and count the frequency of the words , loop the word list to check if the word as a key in the dictionary, if not then dic[word] =1, else if there is already in the dic, then dic[word]+=1, and after the loop finished use sorted () function to sort the dictionary in order of frequency and in alphabetic order, and store the two sorted result into separate file.

**My Partner’s Approach**

Sanaa Mironov has1@umbc.edu

Python based BeautifulSoup library for html parsing and joining for tokenizing.

It also doesn't handle tokens with non-alphabetical characters well, but I do think it’s better than what NLTK do, since I didn’t find there are so many aacute in her top 50 alphabetically first tokens. It splits them and discards the half that keeps the non-alpha char.

**Incorrectly Tokenized**

There are a lot domain names didn’t remove correctly, such as ‘com’, ‘edu’, ‘net’, ‘gov’, ‘org’, also as I mentioned before the HTML5 Entity Names such as &aacute &eacute, only the ‘&’ will be removed, and the rest of word doesn’t make any sense; In addition, there many single character words such as ‘s’, ‘n’, ‘k’, I think some of them might be the word that linked with punctuations, and when punctuation is removed, the tokenizer will count one word as separate words, and there a lot long words, they are the url in the website that not include in the tag and the tokenizer will separate the url into ‘www’ ‘some longword’ ‘domain’.

**Efficiency**

|  |  |  |
| --- | --- | --- |
| Number of files | User | CPU |
| 50 | 5.98 | 6.695 |
| 100 | 6.82 | 7.533 |
| 150 | 7.49 | 8.210 |
| 200 | 8.27 | 8.771 |
| 250 | 8.61 | 9.116 |
| 300 | 9.20 | 9.735 |
| 350 | 11.04 | 11.668 |
| 400 | 11.76 | 12.409 |
| 450 | 12.28 | 12.978 |
| 500 | 13.15 | 13.857 |

**Comparison with Partner**

For the frequency of very first words, we have very similar number, while I have less number of ‘s’ because when I padding the text I have padded some of ’s into is or replaced it with a space; While I have aacute and eacute at 24th and 25th, my partner didn’t have these words, I think the reason is that the BeautifulSoup method removed those using its html symbol and tag library.

From the perspective of efficiency, my partner’s method used less time to finish the process of 500 files, and we have very similar result either in total numbers of words or the top/last 50 words in both docs, I would say my partner’s approach is better.

While I compare the result of other students’ result they sent out in discord, I and my partner have relatively greater number in total words, and relative faster, but only the top frequent 5-10 words are similar. I think the efficiency is also affected by the hardware, and the language we are implementing.