Information Retrieval

Homework 1: Write-up

In the program me and my partner developed we decided to use the NLTK package in python. We chose this package because of its many useful features such as its RegexpTokenizer which is used in the tokenizing of lines of the text documents provided. The lines nltk.download(‘punkt’) and nltk.download(‘averaged\_perceptron\_tagger’) download the Punkt tokenizer models and the average perceptron tagger models, which are essential for NLTK to function correctly when tokenizing a text. The line tokenizer = RegexpTokenizer(r’\w+’) removes all punction from the documents, which is necessary is the task of tokenizing the entire document. We tokenize our text documents on the line newtokens = tokenizer.tokenize(line), which tokenizes each line of text into a list of tokens. Me and my partner had decided to keep the numbers present within our documents, our reason for choosing to do that was several documents consisted of lists of numbers representing IP addresses, amounts of items, phone numbers and several other instances of lists of numbers.

The run time of the program is approximately 25-30 seconds, which I believe to be a excellent runtime considering the program goes through 503 html documents, iterating through each line tokenizing them. Then removing html tags, and punctuation. The program then compares the list of tokenized words from each document to the list of stopwords made by myself and my partner. This stopword list contains individual characters from the alphabet both uppercase and lowercase, some basic words such as “the”, ”and”, “to”, “so” , “as” , “of” , “is” , “for”, “by”, “be”, “are”, “was” , “it”, “this”, “an” and “on” , also contains a list of characters that appear frequently in html such as “<”, ”>”, “- -", “ ! ”, “=” , “{“ , “}” and “==”, and it contains a list of items that may be considered punctuation such as “[“ , “]”, “ . ”, “ –“ , “%”, “|”, “ , ” , “(“ , “)” , “:” , “ \* ” , “ ; ”, “$” , “#” , “&”, and “@”. This stopword list aids in the removal of un-necessary characters within our final tokenized list. After iterating through the stoplist and removing un-necessary characters the program will then organize the list alphabetically creating a alphabetical list and by frequency creating a frequency list. Both the alphabetical and frequency list will be inputed into a output file made at program completion, each output file will contain only the top 50 and botton 50 words and frequencies, separated by 3 newline characters within the file. The theoretical time complexity of the program can be separated into separate sections first the tokenization of the files, this can be done in O(n) time, the removal of the stopwords can be done in O(nk) time where n is the size of the tokenized words list and k is the size of the stopwords. The dictionary operation can be done in O(1) time for insertion and removal from the dictionary. The space complexity of the program is reduced with the removal of the stopwords on each iteration of the program and the clearing of the words list on each iteration as well. The Dictionary space complexity will be O(n) where n is the number of unique tokens found in all documents, the output file space complexity will be proportional to the 100 lines of input put into each file, with each input containing a word and a frequency in both the frequency list and the alphabetical ordering.

Below is a graph showing the time taken for each document to be processed and tokenized. As seen in the graph html document that are larger in turn take a longer time to tokenize then the smaller documents as evidenced by the dips and sudden rises in the graph.

A graph of a document number

Description automatically generated

Below is a graph showing the combined time taken for each document to be processed. As shown in the second graph the html document folder tends to have larger documents at the end of the folder, this is evidenced by the steep incline seen on the graph in the second half.A graph of a graph

Description automatically generated

Below is the Alphabetical list made by the program .

0 | 88855

00 | 62773

000 | 215

0000 | 3

0006 | 1

000m | 2

000nemetorszagdeutsche | 1

001 | 2

0010 | 2

00109 | 1

00198 | 1

002 | 1

0020 | 2

0022 | 1

003 | 1

0030 | 1

0033 | 1

0036 | 2

0043 | 2

006 | 1

007 | 1

009 | 1

0095 | 3

00974 | 2

00adria | 1

00kor | 1

01 | 14921

0100 | 2

0100to | 1

0101 | 1

0103 | 1

010a | 1

011 | 1

0116 | 1

012 | 1

01202 | 1

01203 | 1

01223 | 4

013 | 1

014 | 1

0147 | 1

0160 | 1

017 | 2

018 | 2

019 | 1

01992 | 1

01995 | 1

01997 | 1

02 | 4112

021 | 1

így | 1

ígéreteiknek | 1

írja | 2

írott | 1

írások | 2

ítélve | 1

îâ²² | 1

ò | 1

òplain | 1

óta | 1

õ | 1

õket | 1

õrjöngõ | 1

õt | 1

ölj | 1

önazonosságát | 1

önbecsülésének | 1

öndefiníció | 2

önfejlõdés | 1

önfeladó | 1

önfenntartási | 1

öngerjesztõ | 1

öngyarapításparancsa | 1

önhipnózishoz | 1

önismeret | 1

önkifejezés | 1

önkép | 1

önképe | 1

önmeghatározás | 1

önmeghatározással | 1

önmegváltó | 1

önrendelkezésrõl | 1

öntudat | 1

öntudatlan | 1

öntörvényû | 1

önállóság | 1

örököse | 1

összefogó | 1

összetartozás | 1

össznemzeti | 1

összpontosító | 1

ösztönben | 1

ötvözõdési | 1

ötvözõdött | 1

ú | 2

úgy | 4

új | 5

újonnan | 1

útján | 1

ürességet | 1

Below is a list of the frequency of the words

0 | 88855

00 | 62773

01 | 14921

1 | 5876

2 | 4226

02 | 4112

3 | 3134

4 | 2698

5 | 2447

10 | 2016

6 | 1975

03 | 1879

11 | 1847

0s | 1766

8 | 1764

9 | 1720

7 | 1705

12 | 1666

13 | 1652

14 | 1607

15 | 1605

0fn | 1435

65 | 1386

16 | 1344

496 | 1300

62 | 1292

27 | 1217

50 | 1204

17 | 1152

gov | 1133

1996 | 1104

05 | 1104

7pgs | 1101

20 | 1089

70 | 1087

org | 1084

79yr | 1062

53 | 1030

04 | 1017

96 | 1000

75 | 1000

44 | 985

au | 969

80yr | 967

25 | 953

80 | 930

18 | 914

19 | 906

ms | 903

60 | 899

switching | 1

dilorenzo | 1

bragar | 1

wexler | 1

appease | 1

relents | 1

scrutinizes | 1

execs | 1

castanet | 1

wingfield | 1

marimba | 1

spreadsheets | 1

applets | 1

pioneered | 1

debuted | 1

screensaver | 1

onscreen | 1

widgets | 1

onus | 1

machevsky | 1

persist | 1

intelligently | 1

tuner | 1

toolkit | 1

bongo | 1

assembling | 1

activex | 1

eagerly | 1

polese | 1

microsystems | 1

payne | 1

hoff | 1

shaio | 1

kleiner | 1

caufield | 1

byers | 1

intranets | 1

ain | 1

intelliquest | 1

ventured | 1

statistic | 1

exploit | 1

webmasters | 1

bookmarks | 1

freeware | 1

watermarking | 1

pixels | 1

rescanned | 1

photoshop | 1

picturemarc | 1