PROJECT REPORT

Tokenization and Stemming

Bhattacharya, Abhirup

Part 1:

- The first part of this project includes building the tokenizer, that can effectively tokenize the input text based on certain predefined rules and build a dictionary of the token frequencies.
- I have created 2 functions for the implementation, the first includes a tokenize() function, which can tokenize the input sentence, by removing special characters, hyphens, "'s" in possessives and numbers. The tokens are split from the position where such characters are encountered.
- A calling function for the tokenize() function has been defined, which parses the document to extract only the contents of the <TEXT> tag from each of the files in the Cranfield collection, and passes this content to the tokenize() function, one document at a time.
- The tokenize() function then returns a list of tokens to the doctBuild() function, which in turn creates a dictionary of tokens with the tokens as the key and the frequency as the value.
- Apart from this, the dictBuild() function also maintains a count of the total number of documents parsed and the cumulative count of tokens encountered.

The path to the Cranfield collection can be provided as a command line argument while calling the program.

1. *Number of tokens in the Cranfield text collection : 205793*
2. *Number of unique words in the Cranfield text collection : 7159*
3. *Number of words occuring once in Cranfield collection : 2568*
4. *The 30 most frequent words are :*

the : 18693

of : 11356

and : 5711

a : 5333

to : 4404

in : 4254

is : 4110

for : 3268

are : 2427

with : 2084

by : 1703

at : 1592

that : 1570

on : 1556

flow : 1517

be : 1269

an : 1261

as : 1102

pressure : 1092

this : 1080

from : 1067

boundary : 970

which : 969

number : 914

results : 873

it : 855

layer : 842

mach : 733

theory : 706

was : 698

5. *Average words per document in Cranfield collection : 146.995*

Below are the specific information as mentioned in the Homework description

1. **How long the program took to acquire the text characteristics.**
   Answer: The program took a total of 6940 milliseconds to tokenize the documents and acquire text characteristics

2. **How the program handles:**

   A. Upper and lower case words (e.g. "People", "people", "Apple", "apple");

   Answer: All the words have been normalized to lower case before tokenization. Hence the token dictionary only contains lower case words.

   B. Words with dashes (e.g. "1996-97", "middle-class", "30-year", "tean-ager")

   Answer: Words with dashes have been split from the point of occurrence of the dash.

   For eg. Teen-ager will be split into 2 tokens, ‘teen’ and ‘ager’

   C. Possessives (e.g. "sheriff's", "university's") D. Acronyms (e.g., "U.S.", "U.N.")

   3. Briefly discuss your major algorithms and data structures.

   Answer: I have used a python dictionary, which is a hashmap to maintain a count of the tokens and its corresponding frequencies. This enables us to perform taks such as

counting the number of tokens with single occurrence and finding the N most frequent tokens in the dataset.

Output Screenshot

```
================================================================
====================  TOKENIZATION STATS  =====================
================================================================

Time taken for tokenization                              : 6940 milliseconds
Number of tokens  in the Cranfield text collection       : 205793
Number of unique words  in the Cranfield text collection : 7159
Number of words occuring once in Cranfield collection    : 2568

The 30 most frequent words are                           :
 the : 18693
 of : 11356
 and : 5711
 a : 5333
 to : 4404
 in : 4254
 is : 4110
 for : 3268
 are : 2427
 with : 2084
 by : 1703
 at : 1592
 that : 1570
 on : 1556
 flow : 1517
 be : 1269
 an : 1261
 as : 1102
 pressure : 1092
 this : 1080
 from : 1067
 boundary : 970
 which : 969
 number : 914
 results : 873
 it : 855
 layer : 842
 mach : 733
 theory : 706
 was : 698

Average words per document in Cranfield collection       : 146.995
```

Part 2:

- The second part of the project includes stemming the tokens, by running any porter implementation.
- I have used the NLTK Porter Stemmer implementation for this purpose.
- I have reused the tokenizer function to tokenize the Cranfield collection and pass the obtained tokens to the NLTK Porter Stemmer.
- A dictionary of the stemmed words are then created, with the stemmed word itself as the key and the corresponding frequencies as the values.
- The tokenize() function then returns a list of tokens to the doctBuild() function, which in turn creates a dictionary of tokens with the tokens as the key and the frequency as the value.
- Apart from this, the dictBuild() function also maintains a count of the total number of documents parsed and the cumulative count of tokens encountered.
- I have called the tokenize function again in the stemming function (instead of using the token list obtained in the dictBuild() function) so that this function can directly be re-used directly if needed later. Also, to figure out the time taken for stemming, I supposed that it requires us to find the total time taken for the tokenization along with the stemming process.

Below are the characteristics as obtained:

1. *Number of distinct stems in the Cranfield text collection : 4543*
2. *Number of word stems occurring once in Cranfield collection : 1527*
3. *The 30 most frequent stems are :*
   the : 18693
   of : 11356
   and : 5711
   a : 5333
   to : 4404
   in : 4254
   is : 4110
   for : 3268
   are : 2427
   with : 2084
   flow : 1705
   by : 1703
   at : 1592
   that : 1570
   on : 1556
   be : 1366
   an : 1261
   pressur : 1251
   number : 1246
   as : 1102
   thi : 1080
   result : 1073
   from : 1067

it : 1034
boundari : 998
which : 969
layer : 948
effect : 849
method : 824
theori : 793

4. *Average stems per document in Cranfield collection : 3.245*

Output Screenshot:

```
====================================================================
==================  STEMMING STATS  ================================
====================================================================


Time taken for Stemming                                  : 2733 milliseconds
Number of stems in the Cranfield text collection         : 4543
Number of word stems occuring once in Cranfield collection    : 1527

The 30 most frequent stems are                          :
 the : 18693
 of : 11356
 and : 5711
 a : 5333
 to : 4404
 in : 4254
 is : 4110
 for : 3268
 are : 2427
 with : 2084
 flow : 1705
 by : 1703
 at : 1592
 that : 1570
 on : 1556
 be : 1366
 an : 1261
 pressur : 1251
 number : 1246
 as : 1102
 thi : 1080
 result : 1073
 from : 1067
 it : 1034
 boundari : 998
 which : 969
 layer : 948
 effect : 849
 method : 824
 theori : 793

Average stems per document in Cranfield collection          : 3.245
```