Golas, Ekal

UNIVERSITY OF TEXAS AT DALLAS | 800 W CaMPBELL RD RICHARDSON TX 75080

HOMEWORK 1

INFORMATION RETRIEVAL



Table of Contents

[Algorithm and Design Decisions 2](#_Toc442091874)

[Problem 1 2](#_Toc442091875)

[Problem 2 3](#_Toc442091876)

[Program Overview 4](#_Toc442091877)

[Running time 4](#_Toc442091878)

[Handling special cases in tokenization 4](#_Toc442091879)

[Major Algorithms and Data Structures 4](#_Toc442091880)

[Output 5](#_Toc442091881)

[Tokenization 5](#_Toc442091882)

[Stemming 6](#_Toc442091883)

[References 8](#_Toc442091884)

# Algorithm and Design Decisions

As discussed in the problem statement (Harabagiu, 2016), the following algorithm is proposed for the solution:-

## Problem 1

1. The main function validates the command-line arguments using a third party library, Apache CLI (Apache, 2015) and then passes the path to the Cranfield (Cranfield, 2016) data to the Parser class.
2. The parse() function of the Parser class is then invoked with this path and then following actions are performed:-
   1. Each entry in that path is traversed
   2. If the entry is a directory, the parse() function is called recursively on the entries of that directory
   3. If the entry is a file, step 3 is executed
   4. A variable containing count of total files is incremented
3. The parse() function calls the readFile() function which performs the following actions:-
   1. A line of text is read and the transformText() function is called which handles and transforms this text as:-
      1. All the SGML tags are replaced by a space
      2. All the digits in the text are removed
      3. All the special characters are removed.
      4. If the word is of abbreviated form (ex: U.S.A), the dots are removed (USA)
      5. If the word is of possession form (ex: University’s), the possessives is removed (University).
      6. If the word contains only ‘ such as in the case of their’middle’class then the ‘ is replaced by space and taken as different words
      7. All the hyphens are replaced by space and the word is treated as two separate words
      8. All the multiple consecutive spaces are replaced by a single space
      9. The text is converted to lower case
   2. After the text is transformed, the line is split into words by using space as delimiter
   3. For each word, following action is performed:-
      1. The word is stored in a hashmap as a token with the word as the key and its occurrences in data as the value
      2. Each time this word is found, its occurrence is incremented by 1 in the hashmap
      3. A variable storing total count of tokens is incremented
4. Step 1 to 3 are executed until all the files are processed. After step 3, the object of Parser class has a token map that maps each token with its frequency in the data, total number of documents and total number of tokens.
5. This object of Parser class is used to call displayResults() function, which performs the following actions:-
   1. Displays the total number of documents
   2. Uses the OutputFormatter class to construct a table to display the text characteristics, which uses a third party library, Apache Lang (Apache, 2015) and the following is displayed:-
      1. Total number of tokens
      2. Total number of unique tokens
      3. The class TextCharacteristics is used to call the function getCountForFrequencyOne() which returns number of tokens that occur only once in the data
      4. The class TextCharacteristics is used to call the function getTop30MostFrequent() which returns a list of 30 tokens that occur the most in the data in descending order of their frequency
      5. Average number of tokens per document
6. After the execution is complete, the time taken to execute this problem is displayed in milliseconds

## Problem 2

1. After [Problem 1](#_Problem_1) is executed, the token map created is passed to the Stemming class, which wraps a public implementation of the porter stemmer (UCSD, 2010)
2. The stem() function is invoked in the Stemming class, which performs the following actions:-
   1. Gets all the tokens in the token map, and for each token, does the following:-
      1. Splits the token into characters and adds it to the Stemmer
      2. Runs the stemming function
      3. Obtains the stemmed word after stemming
      4. Places this stemmed word in a hashmap with key as the word and its frequency in data as the value
      5. A variable keeping count of total stems is incremented
3. After the above steps, this object of Stemming class is used to call displayResults() function, which performs the following actions:-
   1. Displays the total number of documents
   2. Uses the OutputFormatter class to construct a table to display the text characteristics, which uses a third party library, Apache Lang (Apache, 2015) and the following is displayed:-
      1. Total number of tokens
      2. Total number of unique tokens
      3. The class TextCharacteristics is used to call the function getCountForFrequencyOne() which returns number of tokens that occur only once in the data
      4. The class TextCharacteristics is used to call the function getTop30MostFrequent() which returns a list of 30 tokens that occur the most in the data in descending order of their frequency
      5. Average number of tokens per document

# Program Overview

## Running time

The program took 1500 milliseconds to acquire text characteristics during the test run.

## Handling special cases in tokenization

1. Upper and lower case words - All the text is converted to lower case words, so that words “People”, “people” and “pEople” are same token (“people”).
2. Words with dashes - Words having dash in the middle (“middle-class”) are separated and taken as two words. So, “middle” and “class” are taken as separate tokens.
3. Possessives: Possessive words (“chandler’s”) are transformed to non-possessive words by truncating “‘s” at the end (“chandler”).
4. Acronyms: Acronyms (“U.S.A”) are transformed by removing dots (“USA”)

## Major Algorithms and Data Structures

The algorithms used are described in the section [Algorithms](#_Algorithm_and_Design). The data structures used are described as follows:-

1. Hashmap to store tokens and stems, where the key is of type String and value is of type integer
   1. Insertion takes linear time if collisions are assumed to not occur
   2. Retrieval takes constant time
   3. To compare and retrieve the top 30 tokens, a class SortableMap is used, which implements Comparator and sorts the keys of a map based on their values. This sorting takes O(n log n) time.
2. ArrayList of Strings to store the top 30 tokens
   1. Insertion is linear time
   2. Retrieval takes constant time

# Output

## Tokenization

###############################################

TOKENIZATION

Total Number of the Documents: 1400

###############################################

1: Total number of tokens: 229225

2: Number of unique words: 9123

3: Number of words that occur only once: 3853

4: 30 most frequent words:

WORD FREQUENCY

the 19451

of 12714

and 6669

a 5937

in 4650

to 4560

is 4113

for 3491

are 2428

with 2263

on 1943

flow 1848

at 1834

by 1756

that 1570

an 1389

be 1271

pressure 1207

boundary 1156

from 1116

as 1113

this 1081

layer 1002

which 975

number 973

results 885

it 855

mach 824

theory 789

shock 712

5: Average number of tokens per document: 163.73214285714286

###############################################

Time taken to acquire the text characteristics is: 1426 milliseconds

## Stemming

###############################################

STEMMING

Total Number of the Documents: 1400

###############################################

1: Total number of tokens: 9123

2: Number of unique words: 6416

3: Number of words that occur only once: 4924

4: 30 most frequent words:

WORD FREQUENCY

gener 15

observ 11

oper 10

comput 9

integr 9

deriv 9

determin 9

indic 8

continu 8

predict 8

separ 8

simul 8

investig 8

compar 7

conduct 7

stabil 7

differ 7

diffus 7

correct 7

depend 7

design 7

develop 7

express 7

origin 7

correl 7

illustr 7

acceler 7

approxim 7

us 7

analyz 6

5: Average number of tokens per document: 6.5164285714285715

###############################################

# References

Apache. (2015, June 17). *Downloads*. Retrieved from Apache Commons: https://commons.apache.org/proper/commons-cli/download\_cli.cgi

Apache. (2015, May 18). *Lang*. Retrieved from Commons: http://commons.apache.org/proper/commons-lang/download\_lang.cgi

Cranfield. (2016, January 27). *Cranfield collection*. Retrieved from UTD csgrads1: /people/cs/s/sanda/cs6322/Cranfield

Harabagiu, S. (2016, Jan 27). *Homework1.* Retrieved from CS 6322: http://www.hlt.utdallas.edu/~sanda/courses/IR/Hm1\_s2016.pdf

UCSD. (2010). *Stemmer.java.* Retrieved from SVN: http://chianti.ucsd.edu/svn/csplugins/trunk/soc/layla/WordCloudPlugin/trunk/WordCloud/src/cytoscape/csplugins/wordcloud/Stemmer.java