Gebze Technical University
Computer Engineering

CSE 222 - 2019 Spring

HOMEWORK 6 REPORT

OZAN ŞELTE
161044061

Course Assistant: Ayşe Şerbetçi Turan

1 INTRODUCTION

1.1 Problem Definition

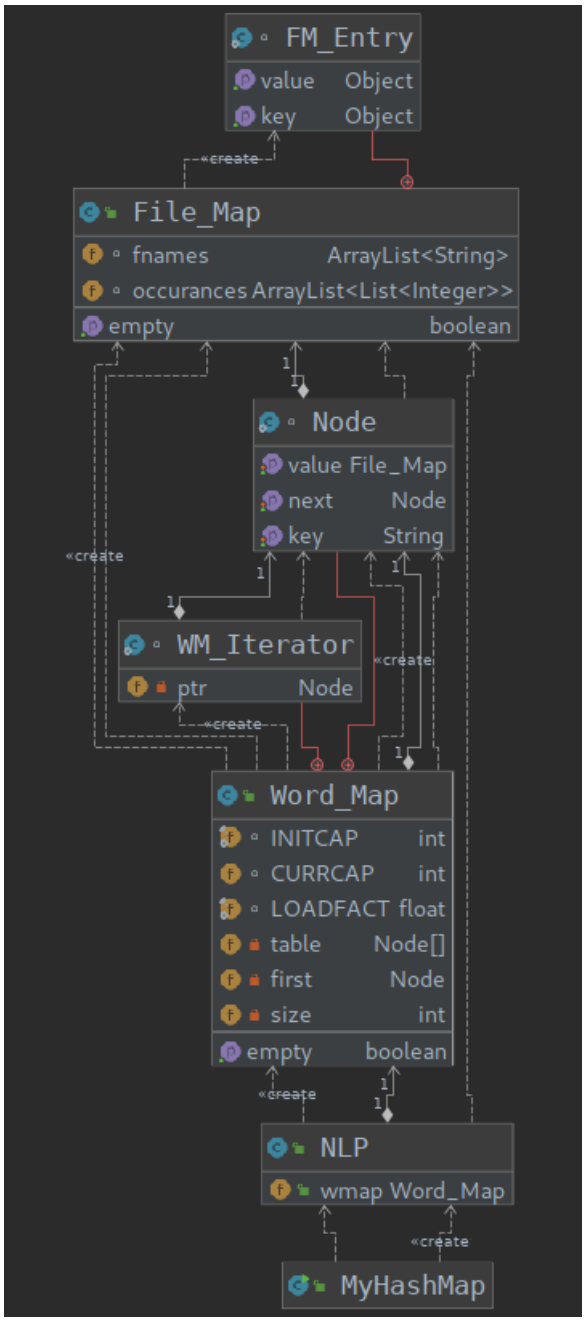
The problem is to calculate the bigram and TFIDF values with our own data structures. In making these calculations, we need to keep every different word in HashMap structures. We are expected to read and understand what we have to do in a separate file.

1.2 System Requirements

A computer, Java Virtual Machine, proper datasets and a query file.

2 METHOD

2.1 Class Diagrams



2.2 Use Case Diagrams

Not required.

2.3 Problem Solution Approach

2.3.1 File_Map

File_Map class has inner FM_Entry class to implement entrySet method. All methods of FM_entry class has O(1) complexity. This File_Map class uses ArrayList so its complexities will not good.

int size - returns ArrayList's size method. It has O(n).

boolean isEmpty - returns ArrayList's isEmpty method. It has O(n).

boolean containsKey - returns ArrayList's containsKey method which calls indexOf method and it has O(n).

boolean containsValue - returns ArrayList's containsKey method which calls indexOf method and it has O(n).

Object get – returns ArrayList's get method. It has O(n).

Object put – calls ArrayList's put methods T(n) = 2*n => O(n).

Object remove – calls ArrayList's remove method, O(n).

void putAll – calls ArrayList's put and Map's get methods, O(n^2).

void clear – calls ArrayList's clear method O(n).

Set<String> keySet – calls HashSet constructor with an ArrayList, O(n^2)

Collection values – calls ArrayList's addAll method with an ArrayList, O(n^2).

2.3.2 Word_Map

Word_Map class has inner WM_Iterator and Node classes to implement iterator method. All methods of the inner classes' methods has O(1) complexity.

Iterator iterator – O(1).

int size – O(1).

boolean isEmpty – O(1).

boolean containsKey – calls get method which has O(1).

boolean containsValue – traverses all nodes one by one, O(n).

File_Map get – uses hash technique, O(1).

File_Map put – uses hash technique, O(1).

void putAll – calls put method for all members of Map, O(n).

void clear – O(1).

Set<String> keySet – traverses all nodes and adds to the Set. Sets' add method has O(n) and traversing has O(n) too so the result is O(n^2).

Collection values – just like the keySet method, it has O(n^2).

int hash – just a basic math operation, O(1).

void checkSize – for every node, calls put method, O(n).

void putEntry – calls get(O(1)) and File_Map's put(O(n)), O(n).

List<Integer> getEntry – returns File_Map's get method, O(n).

2.3.3 NLP

void readDataset – calls enterFile method for all of the files in given directory, O(n).

List<String> bigrams – for each key(word) gets the File_Map and each files' each position it checks and adds to a set, O(n^4).

float tfIDF – uses tf and idf methods, O(n^3).

void printWordMap – uses Word_Map's iterator, O(n).

void enterFile - if n is length of the File, O(n).

String filterWord – if n is length of the String, O(n).

double tf – uses Word_Map's getEntry method and for each key(word) gets the File_Map's fileName lists' size, O(n^2).

double idf – uses Word_Map's keySet and Set's addAll methods, O(n^3).

3 RESULT

3.1 Test Cases

I have used the given query and got the expected results but they have a bit more resolution for floating point number values.

3.2 Running Results

printWorldMap result:

```
anonymity requesting opposed attendees separately unified formulate purpose Several l986
```

Example Screenshot:

```
Please enter the query file or press enter(ex: input.txt):
Size: 7 => [very difficult, very rapid, very vulnerable, very aggressive, very promising,
0.004878173
Size: 13 => [world markets, world bank, world grain, world as, world cocoa, world for, wor
Size: 4 => [costs Transport, costs and, costs of, costs have]
Size: 94 => [is searching, is helping, is flowering, is willing, is scheduled, is unlikely
0.007383948
```