Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 6 REPORT

Melih Yabaş
161044072

Course Assistant: Ayşe Şerbetçi Turan

1 INTRODUCTION

1.1 Problem Definition

We need to solve a basic Natural Language Processing problem. We have many text file in a directory. We want to read all these files in the directory. There are two query types to make. One of them is bigram that a piece of text consisting of two sequential words which occurs in a given text at least once. The other one is TFIDF(Term frequency-inverse document frequency).

First we must take all the words in data structures. There are 2 hashmap class to do this. We will keep the words in Word HashMap. The key for the word hashmap will be the words and the value will refer to another hashmap (file hashmap) which keeps the occurrences of the word in different files. The key for the file hashmap is the filename and the value is an arraylist containing the word positions in that file.

1.2 System Requirements

This program needs Java Virtual Machine to work properly.

Java Virtual Machine Requirements:

2 Windows 10/8/7/Vista/XP/2000

3 Windows Server 2008/2003

4 Intel and 100% compatible processors are supported.

5 Pentium 166 MHz or faster processor wit at least 64 MB of physical RAM.

6 98 MB for free disk space.

Packes:

7 java.util.*

8 java.util.Iterator

9 java.io

10 java.util.*

11 METHOD

11.1 Problem Solution Approach

Firstly, we need to create a Word HashMap. I override all the method of HashMap class. After that, I write the other Map class File HashMap. To write File HashMap is easier because we use 2 ArrayList to keep keys and values. So, we can start operations.

readDataSet: We need to read all the words in these files. I create a FileMap object and put file names and coordinates in. After that, I put the Word and the FileMap object in the WordMap object. I did override put methods to do that. So there is no problem here.

bigrams: To find bigrams I used the WordMap object. I control all the words and their locations.

tfIDF: This method has a formula. First we need to calculate tf and IDF. To do that I count files, and required words with some loops.

printWordMap: I used the WordMap object and I print all the structure on the screen.

Time Complexities of File_Map Methods:

size()

I returned just fnames size. O(1)

isEmpty()

I control if the fnames list has any element. O(1)

containsKey()

I just get the index of key. O(1)

containsValue()

I just get the index of value. O(1)

get()

I get the element if it is not null. O(1)

put()

It is easy to add an element to an ArrayList. O(1)

remove()

I get all elements another arraylists. (n is arralists's size) O(n)

putAll()

I get all elements of the map and put in the lists. N is Map's size. O(n)

clear()

Just clear arraylists. O(1)

keySet()

I get all the elements with addAll method. O(1)

values()

I get all the elements with addAll method. O(1)

entrySet()

I get all elements from two arraylists and keep them in a Node. After that I add them in a set. (n is map's size) O(n)

Time Complexities of Word_Map Methods:

size()

I returned just keyNum (a class variable). O(1)

isEmpty()

I control if the keyNum is 0. O(1)

containsKey()

I control if there is an element in the index using get() . O(n) (n is array's size. It is worst-case)

containsValue()

I control all the words using linked-list. O(n) (n is number of words)

get()

I get the element if it is not null. O(1)

put()

Normally it has a constant time complexity but, we might have to control all the index. O(n)

putAll()

I get all elements of the map and put in the lists. N is Map's size. O(n)

clear()

I just create the table again. O()

Just clear arraylists. O(1)

keySet()

I get all the elements using Linked-List method. n is number of words O(n)

values()

I get all the elements using Linked-List method. n is number of words O(n)

entrySet()

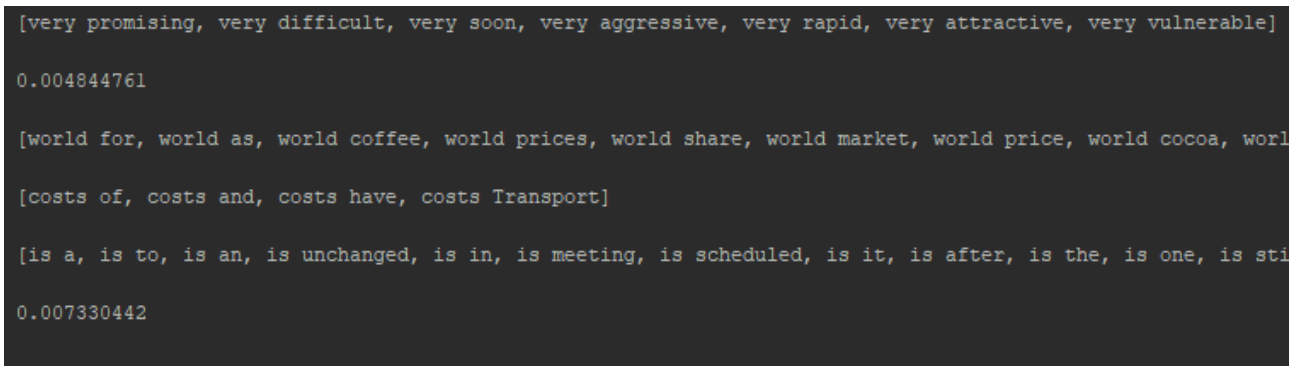
I get all elements from two arraylists and keep them in a Node. After that I add them in a set. (n is map's size) O(n)

12 RESULT

12.1 Test Cases

First I test it with example input file. Then I control some spesific words and files to test it. I check number of words, number of files and the other constant features. Then I decide it is actually working!

12.2 Running Results



Output:

[very promising, very difficult, very soon, very aggressive, very rapid, very attractive, very vulnerable]

0.004844761

[world for, world as, world coffee, world prices, world share, world market, world price, world cocoa, world bank, world made, world markets, world tin, world grain]

[costs of, costs and, costs have, costs Transport]

[is a, is to, is an, is unchanged, is in, is meeting, is scheduled, is it, is after, is the, is one, is still, is not, is no, is how, is now, is are, is set, is only, is that, is possible, is at, is likely, is passed, is estimated, is some, is forecast, is expected, is due, is caused, is more, is depending, is also, is apparent, is slightly, is projected, is ending, is time, is often, is down, is he, is well, is fairly, is foreseeable, is heading, is difficult, is too, is high, is imperative, is going, is being, is put, is keeping, is open, is 112, is defining, is sold, is very, is uncertain, is unlikely, is insufficient, is willing, is proposing, is faced, is currently, is getting, is trying, is basically, is insisting, is unfair, is sending, is planned, is beginning, is affecting, is harvested, is trimming, is improving, is Muda, is great, is aimed, is precisely, is committed, is wrong, is unrealistic, is searching, is showing, is helping, is why, is concerned, is keen, is downward, is sceptical, is favourable, is flowering]

0.007330442