**Comprehensive Practice**

**Section One: Vocabulary Comparison**

**Group ID : 1**

**Group Leader: Imraul Emmaka (201739060027)**

**Group Members:**

**Istiak Alam Fahim (201739060036)**

**Md. Mamun Hasan (201739060034)**

**Mugisha Seif (201739060021)**

**Mustansir Nawaz (201739060007)**

**Mohammad Kamal Lale (201739060006)**

**Major: Computer Science**

**Supervisor: Shan He**

**School of Computer Science**

**Southwest Petroleum University**

**Lab Report Of Comprehensive Practice 1**

**Part 1:**

**Tasks to be carried out:**

 Read two text files

 Find overlapped words/ common words between the two files

 Find unique words between those two files

 Calculating percentage of the overlaps of both files

 Comparing their percentage

**Procedure:**

 Create two ArrayList: list1, list2

 Read two text files using scanner

 Put the strings of the two files into the two ArrayLists consecutively using the method fileToArray():

fileToArray(list1), fileToArray(list2)

/\* for the sake of the task, we get rid of all the stop words from both \* ArrayList, as we can consider them as noise.

\*/

 Create an ArrayList of stop words, and remove them using removeAll() method

 getRidOfDuplicates(list1), getRidOfDuplicates(list2)

 New empty ArrayList uniqueWords = getUniqueWords(list1)

New empty ArrayList overlappedWords = getOverlappedWords(list1)

 Percentage\_book1 = (overlappedWords.size/list1.size())\*100

 Percentage\_book2 = (overlappedWords.size/list2.size())\*100

**key Algorithms:**

**getUniqueWords(ArrayList list):**

* uniqueWords = new empty list

//merge list1 and list2 into a new list

* New ArrayList mergedList = mergeTwoArrayList(list1, list2)
* Collections.sort(mergedList)
* current = list.get(0)
* For i = 1 to mergeList.size() :
  + If list.get(i) is not equal to current :
    - current = list.get(i)
    - If i<list.size()-1 and current is not equal to list.get(i+1) :
      * UniqueWords.add(current)
  + Else if i==list.size()-1 :
    - UniqueWords.add(current)
* Return uniqueWords

**getOverlappedWords(ArrayList list):**

* overlaps = new empty list

//merge list1 and list2 into a new list

* New ArrayList mergedList = mergeTwoArrayList(list1, list2)
* Collections.sort(mergedList)
* Prev = list.get(0)
* For i=1 to list.size() :
  + current = list.get(i)
  + If current equals prev:
    - overlaps.add(prev)
  + Prev = current
* Return overlaps

**getRidOfDuplicates(ArrayList list):**

* noDuplicates = new empty list
* Collections.sort(list)
* Prev = list.get(0)
* For i=1 to list.size() :
  + current = list.get(i)
  + If current not equals prev:
    - noDuplicates.add(prev)
  + Prev = current
  + Else if i==list.size()-1
    - noDuplicates.add(prev)
* Return noDuplicates

**Part 2: Application of the Program**

The goal of our task is to answer the question: “Researchers in the humanities often perform such comparisons of vocabulary in selections of text to answer questions like, ‘Did Christopher Marlowe’ actually write Shakespeare's plays?" which if we generalize stands as Did any author x write author y’s book?

Also we can look at the question as are the writers of the both books same?

To find the answer to this question we analyze both books with our program. We compare both file and the words in the both book. We find out how many words are common, and how many words are unique between them. Of course we get rid of all the stop words and all the repeating words in the book itself first. The number of stop words that we’ve used here is 1527. We divide the number of common words by the size of both books and calculate percentage for both books. Here the numerator – number of overlaps is same for both books, so the percentage depends on the book size. The bigger the book is the smaller the percentage is.

We’ve used many different files to analyze the and support our answer. We’ve used books with same writer and books with different writer.

**Let’s see the output for two different books by Shakespeare:**

Book 1: Antony and Cleopatra

Book 2: As You Like It

The size of words in book 1: 28400

The size of words in book 2: 26544

After getting rid of all the stop words and duplicate of the book itself:

The size of words in book 1: 4822

The size of words in book 2: 3789

number of unique words: 5764

number of common words: 1422

Percentage of common words in book1 = 29.48

Percentage of common words in book1 = 37.53

**Let’s see another example for same writer – Edwin Abott Abott**

Book 1: Flatland: A Romance of Many Dimensions

# Book 2: Philochristus: Memoirs of a Disciple of the Lord

The size of words in book 1: 36958

The size of words in book 2: 134497

After getting rid of all the stop words and duplicate of the book itself:

The size of words in book 1: 4961

The size of words in book 2: 6510

number of unique words: 6448

number of common words: 2511

Percentage of common words in book1 = 50.61

Percentage of common words in book1 = 38.57

**Now Let’s look at output for two different books by different authors:**

Book 1: As you like it

Book 2: Silanus

The size of words in book 1: 26544

The size of words in book 2: 138367

After getting rid of all the stop words and duplicate of the book itself:

The size of words in book 1: 3789

The size of words in book 2: 8388

number of unique words: 7710

number of common words: 2233

Percentage of common words in book1 = 58.93

Percentage of common words in book1 = 26.62

**Now mobydick and Hamlet**

Book 1: Hamlet

Book 2: Mobydick

The size of words in book 1: 32328

The size of words in book 2: 222148

After getting rid of all the duplicate of the book itself:

The size of words in book 1: 5325

The size of words in book 2: 17144

number of unique words: 17622

number of common words: 2423

Percentage of common words in book1 = 45.50

Percentage of common words in book1 = 14.13

**Finding:**

Here we can see that the difference between percentages for books by different authors is high, and for books by same author is low.

So we can state this hypothesis: If the percentages of the overlap word is two books are same, then they have same author or perhaps one author is very much influenced by the other.

In our case for Hamlet and mobydick, the difference between the percentages is 32.31, which is pretty high. So we can conclude that they do not have the same author.

**Source code:**

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Scanner;

/\*\*

\*

\* **@author** emmaka

\*

\*/

**public** **class** Comprehensive\_1 {

**public** **static** **void** main(String[] args) **throws** FileNotFoundException {

/\*

\* create 2 ArrayList: list1 and lis2

\* list1 to store book1 and list2 to store book2

\*/

ArrayList<String> list1 = **new** ArrayList<String>();

ArrayList<String> list2 = **new** ArrayList<String>();//Edwin Abott Abott//flatland.txt

Scanner scanner1 = **new** Scanner(**new** File("C:\\Users\\Lab\\workspace\\Test\\src\\As You Like.txt"));

Scanner scanner2 = **new** Scanner(**new** File("C:\\Users\\Lab\\workspace\\Test\\src\\silanus.txt"));//Edwin Abott Abott//Silanus.txt

//add all the words from text files to ArrayList

*fileToArrayList*(scanner1, list1);

*fileToArrayList*(scanner2, list2);

System.***out***.println("list1 size: "+list1.size());

System.***out***.println("list2 size: "+list2.size());

//---------------------------------------------------------------------------------

//get rid of the duplicates

list1 = *getRidOfDuplicates*(list1);

list2 = *getRidOfDuplicates*(list2);

System.***out***.println("list1 size after getting "

+ "rid of all the duplicates: "+list1.size());

System.***out***.println("list2 size after getting "

+ "rid of all the duplicates: "+list2.size());

// ArrayList countOfWordsList1 = wordOccurrenceCounter(list1);

// System.out.println(countOfWordsList1);

// System.out.println(list2);

// ArrayList countOfWordsList2 = wordOccurrenceCounter(list2);

// System.out.println(countOfWordsList2);

//create an arraylist to store the uniqueWords

ArrayList<String> uniqueWords = *getUniqueWords*(list1, list2);

System.***out***.println("number of unique words: " + uniqueWords.size());

//create an ArrayList overlaps to store the overlaps

ArrayList<String> overlapedWords = *getOverlappedWords*(list1, list2);

System.***out***.println("number of overlapped words: "+overlapedWords.size());

//Find the percentage of overlap

**double** overlap\_number = overlapedWords.size();

**double** percentage\_book1 = (overlap\_number / list1.size()) \* 100;

**double** percentage\_book2 = (overlap\_number / list2.size()) \* 100;

//System.out.println("sum = "+sum);

System.***out***.println("Percentage of overlap in book 1 = "+percentage\_book1);

System.***out***.println("Percentage of overlap in book 2 = "+percentage\_book2);

}

**static** **void** printKmostOccurence(ArrayList<String> list, **int** k) {

ArrayList countsArrayList = *wordOccurrenceCounter*(list);

**for** (**int** i = 0 ; i < 2\*k; i++) {

//System.out.println("Frequency of word "++);

}

}

/\*\*

\*

\* **@param** sc

\* **@param** list

\* Adds strings from a file to an Arraylist

\*/

**static** **void** fileToArrayList(Scanner sc, ArrayList<String> list) {

**while**(sc.hasNext()) {

String name = sc.useDelimiter("[^a-zA-Z']+").next().toLowerCase();

list.add(name);

}

}

/\*\*

\* This method takes in two ArrayList as formal parameter and adds

\* list2 at the back of list1. Then returns list1.

\*

\* **@param** list1 ArrayList1, at the back of which list2 gets added

\* **@param** list2 ArrayList2

\* **@return** newArrayList

\*/

**static** ArrayList<String> mergeTwoArrayList(ArrayList<String> list1, ArrayList<String> list2) {

ArrayList<String> newArrayList = **new** ArrayList<String>();

//add words in list1 into newArrayList

**for** (**int** i = 0, len = list1.size(); i < len; i++) {

newArrayList.add(list1.get(i));

}

//add words in list2 into newArrayList

**for** (**int** i = 0, len = list2.size(); i < len; i++) {

newArrayList.add(list2.get(i));

}

**return** newArrayList;

}

/\*\*

\* This method gets rid of the duplicate words

\* if any word has appeared several tiimes, add only one copy into noDuplicates

\* If any word unique, add it to noDuplicates

\* **@param** list

\* **@return** noDuplicates

\*/

**static** ArrayList<String> getRidOfDuplicates(ArrayList<String> list) {

ArrayList<String> noDuplicates = **new** ArrayList<String>();

//sort list

Collections.*sort*(list);

//find duplicates and get rid of 'em

String prev =list.get(0);

**for**(**int** i = 1; i < list.size(); i++) {

String curr = list.get(i);

**if** (!curr.equals(prev))

noDuplicates.add(prev);

prev = curr;

**if** (i == list.size()-1)

noDuplicates.add(prev);

}

**return** noDuplicates;

}

/\*\*

\* This method takes in two ArrayLists and returns a new ArrayList

\* which contains the unique words between them.

\* first it merges the two lists into the mergedList. then sorts them.

\* Then finds the unique words.

\*

\* **@param** list1

\* **@param** list2

\* **@return** uniqueWords

\*/

**static** ArrayList<String> getUniqueWords(ArrayList<String> list1, ArrayList<String> list2) {

ArrayList<String> uniqueWords = **new** ArrayList<String>();

//merge list1 and list2 into mergedList

ArrayList<String> mergedList = *mergeTwoArrayList*(list1, list2);

Collections.*sort*(mergedList);

//look for unique words.

//If it is unique add it to uniqueWords

String curr = mergedList.get(0);

**for**(**int** i = 1, len = mergedList.size(); i < len; i++) {

**if** (!mergedList.get(i).equals(curr)){

curr = mergedList.get(i);

**if**(i < len - 1 && !curr.equals(mergedList.get(i+1)) ){

uniqueWords.add(curr);

}

**else** **if** (i == len-1) {

uniqueWords.add(curr);

}

}

}

**return** uniqueWords;

}

/\*\*

\* This method takes in two ArrayLists and returns a new ArrayList

\* which contains the overlapped words between them.

\* first it merges the two lists into the mergedList. then sorts them.

\* Then finds the overlapped words. Adds them into overlaps

\*

\* **@param** list

\* **@return**

\*/

**static** ArrayList<String> getOverlappedWords(ArrayList<String> list1, ArrayList<String> list2) {

ArrayList<String> overlaps = **new** ArrayList<String>();

//merge list1 and list2 into mergedList

ArrayList<String> mergedList = *mergeTwoArrayList*(list1, list2);

Collections.*sort*(mergedList);

String prev =mergedList.get(0);

**for**(**int** i = 1, n = mergedList.size() - 1; i < n; i++) {

String curr = mergedList.get(i);

**if** (curr.equals(prev))

overlaps.add(prev);

prev = curr;

}

**return** overlaps;

}

/\*\*

\*

\* **@param** list

\* **@return** counterList

\*/

**static** ArrayList wordOccurrenceCounter(ArrayList<String> list) {

ArrayList counterList = **new** ArrayList();

Collections.*sort*(list);

String currString = list.get(0);

**int** counter = 1;

**for** (**int** i = 1, len = list.size(); i < len; i++) {

**if** (list.get(i).equals(currString))

counter++;

**else** {

counterList.add(counter);

counterList.add(currString);

counter = 1;

}

currString = list.get(i);

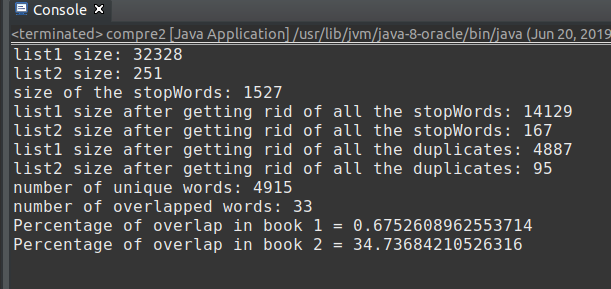
}

**return** counterList;

}

}

**Output:**

****