Pokhara University

Everest Engineering Collage

Sanapa-2, Lalitpur

Lab Report on “Vector Space Model”

Submitted to

Shailesh Pandey

of

IT Department

Submitted By

Bhupendra Prasad Bhat-[20120010]

Class Roll No :-04

Submitted Date

2023/07/2

WordCount.java

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;
public class WordCount {
    //Data structure to map a word to its frequency i.e. a Map
    HashMap<String,Integer> wordMap = new HashMap<String,Integer>();//    KEY VALUE ie word & its count
    HashMap<String,Integer> pairMap = new HashMap<String,Integer>();//KEY VALUE ie word & its count

    public void readFile(String file) {
        String word = "";
        //var for pairMap
        String prevWord = "";
        String nextWord = "";
        String tempKey ="";
        Scanner sc;
        try {
            sc = new Scanner(new File("C:\\Users\\bhupe\\Downloads\\"+file));
            //Split the text into words using whitespace character
            sc.useDelimiter(" ");//("\\s +") for multiple spaces ie one of more occurance of space
            //Get a word at a time
            while(sc.hasNext()) {
                word = sc.next(); //Get a word
                word = word.toLowerCase(); //Convert words to lowercase
                word = word.replaceAll("[,;.]", ""); //Delete punctuation
                //Store words as keys and frequencies as values
                if(!word.equals("")) {
                    if(wordMap.containsKey(word))
                        wordMap.put(word, wordMap.get(word) + 1);
                    else
                        wordMap.put(word, 1);
                }
            }

            //for keyMap
            sc = new Scanner(new File("C:\\Users\\bhupe\\Downloads\\"+file));
            //Split the text into words using whitespace character
            sc.useDelimiter(" ");//("\\s +") for multiple spaces ie one of more occurance of space
            //Get a word at a time
            while(sc.hasNext()) {
                nextWord = sc.next(); //Get a word
                nextWord = nextWord.toLowerCase(); //Convert words to lowercase
                nextWord = nextWord.replaceAll("[,;]", ""); //Delete punctuation
                //Store words as keys and frequencies as values
                if(!nextWord.equals("")&& !prevWord.equals("")) {
                    tempKey = prevWord + ":" +nextWord;
                    if(pairMap.containsKey(tempKey))
                        pairMap.put(tempKey, pairMap.get(tempKey) + 1);
                    else
                        pairMap.put(tempKey, 1);
                }
                prevWord = nextWord;
            }


            sc.close();
        } catch (FileNotFoundException e) {
            System.out.println("File does not exist!" + e);
        }
    }
    public void printPairFreq() {
        System.out.print("check");
        for(String pairWord: pairMap.keySet()) {
                    System.out.print("check");

            int count = pairMap.get(pairWord);
            System.out.print("{"+pairWord+", "+count+"}"+"\n");
        }
    }

    public void printWordFreq() {

        for(String word: wordMap.keySet()) {
            int count = wordMap.get(word);
            System.out.print("{"+word+", "+count+"}"+"\n");
        }
    }

   double probWord(String word){//tf
       double p;//probality
       int total_count=0;
       int word_count=0;
       for(String wordInMap: wordMap.keySet()) {
            total_count=total_count+1;
        }
       word_count = wordMap.get(word);
       p = word_count*1.0/total_count*1.0; // type casting
       return p;
   }


   double probOfWordPair(String first,String second){
       String word = first + ":" +second;
       double p;//probality
       int total_count=0;
       int wordPair_count=0;
       for(String wordInMap: pairMap.keySet()) {
            total_count=total_count+1;
        }
       wordPair_count = pairMap.get(word);
       p = wordPair_count*1.0/total_count*1.0; // type casting
       return p;
   }

```

```java

    double idf(String[] arr,String term){
        int length = arr.length;
        double idft=0;
        double p;
        int dft=0;
        for (int i = 0; i < length; i++) {
             readFile(arr[i]+".txt");//file name
             p=probWord(term);
             if(p>0){
                 dft++;
             }
         }

        idft=Math.log(length/dft); //length is total number of
files & dft is the total number of files that have the
particular tern that is passed in the function

        return idft;
    }


    double tf_idf(String[] arr,String term){
        double overAllProb=0;
        double p;
        for(int i = 0; i < arr.length; i++) {
             readFile(arr[i]+".txt");//file name
             p=probWord(term);
             if(p>0){
                 overAllProb+=p;
             }
         }

        double tf = overAllProb/arr.length;
        double idf = idf(arr, term);//this logic is worng

        return tf*idf;
    }


    public void main(String[] args) {
        WordCount unigram = new WordCount();
        unigram.readFile("file.txt");
        //unigram.printWordFreq();
        //unigram.printPairFreq();
        double pWord = unigram.probWord("that");
        System.out.print(pWord+"\n");
        double pWordPair = unigram.probOfWordPair("is","the");
        System.out.print(pWordPair+"\n");

        String[] numbers = {"file1","file2","file3"};
        double idf = idf(numbers,"the");
        System.out.println(idf);

        double id_idf = tf_idf(numbers,"the");
        System.out.println(id_idf);

    }
}

```