Since the text file you have provided was very large which my laptop could not take, it gets hanged immediately whenever I run it, therefore I have used certain portion of it, due to which I cannot solved problems for similar data, I tried as below.

**SOURCE CODE**

**import** java.io.BufferedReader;  
**import** java.io.FileReader;  
**import** java.util.\*;  
**import** java.util.Set;  
**import** java.util.List;  
**import** java.util.HashMap;  
**import** java.util.Map;  
  
*/\*\*  
 \* Created by asmita on 6/3/2016.  
 \*/***public class** Probability {  
 **public static float** *gv1*;  
 **public static void** main(String[] args) **throws** Exception  
 {  
 FileReader file = **new** FileReader(**"C:/Users/asmita/Desktop/Text1.txt"**);*// parsed the file using* BufferedReader reader= **new** BufferedReader(file);  
 String text=**""**;  
 String line=reader.readLine();  
 **while**(line!=**null**){*//the parsed text is passed into the text variable* text+=line;  
 line=reader.readLine();  
 }  
 *//System.out.println(text);* String text1=text.toUpperCase();*//all the strings are changed into uppercase  
 //System.out.println(text1);* String text2[]=text1.split(**"[(' ),.!;-]"**);*// text splited* **int** totalWords=text2.**length**;  
 System.***out***.println(totalWords);  
 HashMap<String, Integer> hasmap= **new** HashMap<String, Integer>();*//hasmap is introduced to have key as a word and value as a frequency* **for** (String word:text2)  
 {  
 **if**(word.length()<=2){*//ignored the word whose length is less than 3* **continue**;  
 }  
 Integer existingCount=hasmap.get(word);*//counted the repetation of the words woith get function* hasmap.put(word,existingCount==**null**?1:(existingCount+1));  
 }  
  
  
 Map<Integer, String> map = *sortByValues*(hasmap);*//function is called which uses comparator to sort* **int** m=0;  
 Set set2 = map.entrySet();  
 Iterator iterator2 = set2.iterator();  
 System.***out***.println(**"word\t"**+ **"frequency\t"**+**"Rank"**);  
 **while**(m<20){  
 Map.Entry me2 = (Map.Entry)iterator2.next();  
  
 System.***out***.print(me2.getKey() + **" :"**);  
 System.***out***.print(me2.getValue()+**":"**);  
 System.***out***.println(++m);  
 }  
 *// Calculatimg the releative frequency* System.***out***.println(**"The relative frequency of THE is "**+ *ProbabilityForOne*(hasmap,**"THE"**,totalWords));  
 System.***out***.println(**"The relative frequency of BRAVE is "**+ *ProbabilityForOne*(hasmap,**"BRAVE"**,totalWords));  
 System.***out***.println(**"The relative frequency of HIS is "**+ *ProbabilityForOne*(hasmap,**"HIS"**,totalWords));  
 System.***out***.println(**"The relative frequency of HAVE is "**+ *ProbabilityForOne*(hasmap,**"HAVE"**,totalWords));  
  
  
  
 *//Calculating conditional probability* System.***out***.println(**"-----------Conditional probability--------------"**);  
 System.***out***.println(**"The probability of occurence of MAN after THIS is "**+*ProbabilityForTwo*(hasmap,text2,**"THIS"**,**"MAN"**,totalWords));  
 System.***out***.println(**"The probability of occurence of SHE after HATH is "**+*ProbabilityForTwo*(hasmap,text2,**"HATH"**,**"SHE"**,totalWords));  
 System.***out***.println(**"The probability of occurence of YOU after YET is "**+*ProbabilityForTwo*(hasmap,text2,**"YET"**,**"YOU"**,totalWords));  
 System.***out***.println(**"The probability of occurence of FOR after YOU is "**+*ProbabilityForTwo*(hasmap,text2,**"THEE"**,**"FOR"**,totalWords));  
  
 }  
 **private static** HashMap sortByValues(HashMap map) {  
 List list = **new** LinkedList(map.entrySet());  
 *// Defined Custom Comparator here* Collections.*sort*(list, **new** Comparator() {  
 **public int** compare(Object o1, Object o2) {  
 **return** ((Comparable) ((Map.Entry) (o2)).getValue())  
 .compareTo(((Map.Entry) (o1)).getValue());  
 }  
 });  
  
 *// Here I am copying the sorted list in HashMap  
 // using LinkedHashMap to preserve the insertion order* HashMap sortedHashMap = **new** LinkedHashMap();  
 **for** (Iterator it = list.iterator(); it.hasNext();) {  
 Map.Entry entry = (Map.Entry) it.next();  
 sortedHashMap.put(entry.getKey(), entry.getValue());  
 }  
 **return** sortedHashMap;  
 }  
  
 **private static int** countInPairs(String []str1, String word1, String word2){*// function to count the sequence of two words that are repeated  
 //String t;* **int** count=0;  
 **for** (**int** i = 0; i <str1.**length**-1 ; i++) {  
 *//System.out.println(str1[i]);* **if**(str1[i].equals(word1) && str1[i+1].equals(word2)){  
 count++;  
 }  
 }  
  
 **return** count;  
 }  
 *//function to calculate probability of one word* **private static float** ProbabilityForOne( HashMap<String,Integer> map,String str, **int** totalCount){  
 **int** countWord1=map.get(str);  
 **float** p1=(countWord1/(**float**)totalCount);  
 **return** p1;  
 *// gv1=p1;  
 //System.out.println("The relative frequency of the word "+ str + " is:"+ p1);* }  
 *//function to calculate probability of two word* **private static float** ProbabilityForTwo(HashMap<String, Integer> map, String []str,String str1, String str2, **int** totalCount){  
 **int** countInPairs=*countInPairs*(str,str1,str2);  
 **int** countStr1=map.get(str1);  
 **float** p2=(countInPairs/(**float**)countStr1);  
 **return** p2;  
 *//System.out.println("The probability of occurence of "+ str1 + " after " + str2 +" is:"+p2);* }}

**OUTPUT**: Certain portion of output is shown below.

