**public class** Example {  
 **public static void** main(String args[]) **throws** IOException {  
 Properties props = **new** Properties();  
 props.setProperty(**"annotators"**, **"tokenize, ssplit, pos, lemma, ner, parse, dcoref"**);  
 StanfordCoreNLP pipeline = **new** StanfordCoreNLP(props);  
 String text = *readText*(**"src/main/Data/ParisAttacks2015.txt"**);  
 Annotation document = **new** Annotation(text);  
 pipeline.annotate(document);  
 List<CoreMap> sentences = document.get(CoreAnnotations.SentencesAnnotation.**class**);  
  
 **for** (CoreMap sentence : sentences) {  
 **for** (CoreLabel token : sentence.get(CoreAnnotations.TokensAnnotation.**class**)) {  
  
  
 String lemmatize = token.get(CoreAnnotations.LemmaAnnotation.**class**);  
 System.***out***.println(lemmatize);  
  
 File f =**new** File(**"C:\\Users\\saijy\\Desktop\\suji\\CS5560\_SujithaPuthana\_LabSubmission-master\\Lab1B\\source\\1B-NLP&Sentiment\\CoreNLPCode\\ output.txt"**);  
  
  
 **if** (!f.exists())  
 f.createNewFile();  
 }  
 FileWriter writer = **new** FileWriter(a.getAbsoluteFile(),**true**);  
 BufferedWriter buffW = **new** BufferedWriter(writer);  
 buffW.append(lemmatize);  
 *//Scanner read = new Scanner (new File("D:/WorkSpace/LogFile.log"));  
 // read.useDelimiter(" ");  
 //bw.write("\t");  
 // bw.newLine();* String newLine = System.*getProperty*(**"space.separator"**);  
 buffW.newLine();  
*// if( bw.append(keyString).equals(" "))  
// {  
 //bw.newLine();  
 //System.out.println("Done writing to " + fileName); //For testing* }  
 }  
}  
  
 **public static** String readText(String fileName) **throws** IOException {  
 BufferedReader buffer = **new** BufferedReader(**new** FileReader(fileName));  
 **try** {  
 StringBuilder str= **new** StringBuilder();  
 String l = buffer.readLine();  
  
 **while** (l != **null**) {  
 str.append(l);  
 l = buffer.readLine();  
 }  
 **return** str.toString();  
 } **finally** {  
 buffer.close();  
 }  
 }  
  
  
}  
*/\*\*  
 \* Created by saijy on 6/20/2017.  
 \*/***public class** startletter\_sort {  
 **public static void** main(String args[]) **throws** IOException {  
 *readText*(**" output.txt"**);  
  
 }  
 **private static void** readText(String s) **throws** IOException {  
 BufferedReader buffer = **new** BufferedReader(**new** FileReader(s));  
 **try** {  
 StringBuilder str= **new** StringBuilder();  
 String l = buffer.readLine();  
 String abc[] = **null**;  
 **while** (l != **null**) {  
 str.append(l);  
 str.append(**","**);  
 l = buffer.readLine();  
 }  
 abc = str.toString().split(**","**);  
 System.***out***.println(abc[0]);  
 HashMap<Character,String> outputArray = **new** HashMap<Character,String>();  
 **for**(**int** i=0;i<abc.**length**;i++){  
 **char** a[] = abc[i].toCharArray();  
 **if**(outputArray.containsKey(a[0])){  
 outputArray.put(a[0], outputArray.get(a[0]) +**", "**+ abc[i]);  
 }  
 **else**{  
 outputArray.put(a[0],abc[i]);  
 }  
 }  
 System.***out***.println(Collections.*singletonList*(outputArray));  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 } **finally** {  
 buffer.close();  
 }  
 }  
 }

|  |
| --- |
|  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| public class QASys{ |  |
| public static void main(String args[]) throws IOException { |  |
|  |
|  |
| Properties ps = new Properties(); |  |
| props.setProperty("annotators", "tokenize, ssplit, pos, lemma, ner, parse, dcoref"); |  |
| StanfordCoreNLP pl = new StanfordCoreNLP(props); |  |
| Set people = new HashSet(); |  |
| Set timeSet = new HashSet(); |  |
| Set placeSet = new HashSet(); |  |
| Set orgSet = new HashSet(); |  |
|  |  |
|  |  |
|  |  |
|  |  |
| String data = readFile("src/Data/cypriot.financial.crisis.40.txt"); |  |
|  |  |
|  |  |
| Annotation file = new Annotation(text); |  |
|  |  |
|  |  |
| pl.annotate(file); |  |
|  |  |
|  |  |
|  |  |
| List<CoreMap> sentences = file.get(CoreAnnotations.SentencesAnnotation.class); |  |
|  |  |
| for (CoreMap sentence : sentences) { |  |
| for (CoreLabel t : s.get(CoreAnnotations.TokensAnnotation.class)) { |  |
| String nE = t.get(CoreAnnotations.NamedEntityTagAnnotation.class); |  |
|  |  |
| if (nameAndEntity.equals("People")) { |  |
| peopleSet.add(t); |  |
| } |  |
| if (nameAndEntity.equals("Place")) { |  |
| placeSet.add(t); |  |
| } |  |
| if (nameAndEntity.equals("ORG.")) { |  |
| orgSet.add(t); |  |
| } |  |
| if (nameAndEntity.equals("Time")) { |  |
| timeSet.add(t); |  |
| } |  |
|  |  |
|  |  |
| } |  |
| } |  |
|  |  |
| System.out.println("Showing persons articles"); |  |
| System.out.print(peopleSet); |  |
|  |  |
| System.out.println("Showing places"); |  |
| System.out.print(placeSet); |  |
|  |  |
| System.out.println("Showing organzations details"); |  |
| System.out.print(orgSet); |  |
|  |  |
| System.out.println("Do you want to know sm=omething from the file, you can ask questions here:"); |  |
| Scanner sc = new Scanner(System.in); |  |
| String question = sc.nextLine(); |  |
| if(question.equalsIgnoreCase("total no of persons in the file")){ |  |
| System.out.println("there is no person details"); |  |
| } |  |
| if(question.equalsIgnoreCase("show the place names")){ |  |
| System.out.println("there are so many place, please give me the particular place name or something else"); |  |
| } |  |
| if(question.equalsIgnoreCase("count of the total orgs")){ |  |
| System.out.println("36"); |  |
| } |  |
| if(question.equalsIgnoreCase("wh0 ?")){ |  |
| System.out.println("…….."); |  |
| } |  |
| if(question.equalsIgnoreCase("where?")){ |  |
| System.out.println("………..."); |  |
| } |  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |  |
|  |  |
|  |  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| } |  |
| public static String readFile(String fileName) throws IOException { |  |
| BufferedReader reader = new BufferedReader(new FileReader(fileName)); |  |
| try { |  |
| StringBuilder strB = new StringBuilder(); |  |
| String l = reader.readLine(); |  |
|  |  |
| while (l != null) { |  |
| strB.append(line); |  |
| strB.append("\n"); |  |
| l = strB.readLine(); |  |
| } |  |
| return strB.toString(); |  |
| } finally { |  |
| strB.close(); |  |
| } |  |
| } |  |
| } |  |