My Search Engine

ICS-202 Data Structures (Lab)

Semester: 161

Instructor: Faisal Alvi

Mohammed Alshulah

201352890

Source code:

1: import java.io.File;

2: import java.io.FileInputStream;

3: import java.io.FileNotFoundException;

4: import java.io.FileWriter;

5: import java.io.IOException;

6: import java.util.ArrayList;

7: import java.util.Collections;

8: import java.util.HashSet;

9: import java.util.LinkedList;

10: import java.util.List;

11: import java.util.Scanner;

12: import java.util.Set;

13:

14: public class Read {

15:

16:

17:

18: ArrayList<String> fileNames = new ArrayList<String>();

19: ArrayList<Integer> NumberOfWords = new ArrayList<Integer>();

20: List<String> wordsList = new ArrayList<String>() ;

21: LinkedList<List<String>> AllWords = new LinkedList<List<String>>();

22: Set<String> StopWords = new HashSet<String>();

23:

24: String target\_dir = "src/files";

25: File dir = new File(target\_dir);

26: File[] files = dir.listFiles();

27:

28: //insert file names in an array

29: public void fileNames(){

30: for(File f : files){

31: if(!(f.getName().equals(".DS\_Store"))){

32: fileNames.add(f.getName());

33: }

34: }

35: }

36:

37:

38: //count number of words for each file

39: public void numberOfwords() throws FileNotFoundException{

40: for(File f : files){

41: Scanner sc = new Scanner(new FileInputStream(f));

42: int count=0;

43: while(sc.hasNext()){

44: if(!(sc.next().equals("[^A-Za-z]")))

45: count++;

46: }

47: if(!(f.getName().equals(".DS\_Store"))){

48: NumberOfWords.add(count);

49: }

50: sc.close();

51: }

52: }

53:

54:

55:

56: //Store words

57: public void words() throws IOException{

58:

59: for(File f : files){

60:

61: if(f.isFile() && !(f.getName().equals(".DS\_Store"))) {

62: Scanner sc = new Scanner(new FileInputStream(f));

63: readStopWords();

64: wordsList = new ArrayList<String>();

65: while (sc.hasNext()) {

66: String theWord = sc.next().toLowerCase();

67: theWord = theWord.replaceAll("[^A-Za-z]", "");

68: // String addingWord = deletePunctuation(theWord);

69: if(!(StopWords.contains(theWord))){

70: wordsList.add(theWord);

71: Collections.sort(wordsList);

72: }

73: }

74: AllWords.add(wordsList);

75:

76:

77:

78:

79: sc.close();

80: }

81:

82: }

83: }

84:

85: //write words in external file

86: public void writer() throws IOException{

87:

88: FileWriter writer = new FileWriter("src/wordsnew.txt");

89:

90: int fileIndex = 0;

91: for(List<String> str: AllWords) {

92: if(fileNames.size() > fileIndex)

93: writer.write(fileNames.get(fileIndex)+ "\n");

94: for(int j = 0; j < str.size(); j++){

95: writer.write(j +"- "+str.get(j)+ "\n");

96: }

97:

98: writer.write("\n\n");

99: fileIndex++;

100: }

101:

102: writer.close();

103: }

104:

105:

106:

107: //delete stop words

108: public void readStopWords() throws FileNotFoundException{

109: Scanner sc = new Scanner(new FileInputStream("src/StopWords.txt"));

110:

111: while(sc.hasNext()){

112: String newWord = sc.nextLine();

113: newWord.replaceAll("\"", "");

114: StopWords.add(newWord);

115: }

116:

117: sc.close();

118:

119: }

120:

121:

122: //print context of file

123: public String findContext(String FileName, String word, String word2, int operator) throws FileNotFoundException{

124: ArrayList<String> temp = new ArrayList<String>();

125: Scanner sc = new Scanner(new FileInputStream("src/files/"+FileName));

126: String context = null;

127: while(sc.hasNext()){

128: String fileWords = sc.next().replaceAll("[^A-Za-z]", "");

129:

130: temp.add(fileWords.toLowerCase());

131: }

132: if(operator == 1){

133: if(temp.contains(word)){

134: int index = temp.indexOf(word);

135: context = "..."+temp.get(index)+" "+temp.get(index+1)+" "+temp.get(index+2)+ "...";

136: }

137: }

138: else

139: {

140: if(temp.contains(word) && temp.contains(word2)){

141: int index = temp.indexOf(word);

142: if(temp.indexOf(word2) == index+1)

143: context = "..."+temp.get(index)+" "+temp.get(index+1)+" "+temp.get(index+2)+" "+temp.get(index+3)+"...";

144: else

145: context = "...";

146: }

147:

148: }

149:

150: sc.close();

151:

152: return context;

153:

154: }

155:

156:

157:

158:

159: }

1: import java.io.FileNotFoundException;

2: import java.io.IOException;

3: import java.util.ArrayList;

4: import java.util.Collections;

5: import java.util.Hashtable;

6: import java.util.List;

7: import java.util.Scanner;

8:

9: public class Search {

10:

11: Read SearchIn = new Read();

12: Hashtable<Float, Integer> term = new Hashtable<Float,Integer>();

13: ArrayList<Float> frequency = new ArrayList<Float>();

14:

15:

16: int numberOfOcccurence,numberOfOcccurence2 ;

17: int indexOfList ;

18: int fileTotalWords ;

19: float frequencyOfWord,frequencyOfWord2;

20: float productOfrelativeFrequency;

21: float sumOfRelativefrequency;

22:

23:

24:

25:

26:

27: public void main() throws IOException{

28: SearchIn.fileNames();

29: SearchIn.words();

30: SearchIn.writer();

31: SearchIn.numberOfwords();

32: System.out.println("Search Engine.... ");

33: System.out.println("instruction: use AND, OR, NOT as operators ");

34:

35:

36: Scanner sc = new Scanner(System.in);

37: String searchLine =null;

38: Boolean condition = true;

39:

40: do{

41: System.out.println("\nEnter a search term: ");

42: searchLine = sc.nextLine();

43:

44: //takes the words in the input scanner

45: String[] a = searchLine.split(" ");

46:

47: //recognize the operators in term

48: if(a.length == 1){

49: oneWord(a[0]);

50: condition = false;

51: }

52: else if(a.length == 2){

53: notOperator(a[1]);

54: condition = false;

55: }

56: else if(a.length == 3){

57: if(a[1].equals("AND")){

58: andOperator(a[0],a[2]);

59: condition = false;

60: }

61: else{

62: orOperator(a[0],a[2]);

63: condition = false;

64: }

65:

66: }

67: else{

68: andNotOperator(a[0],a[2],a[4]);

69: condition = false;

70: }

71:

72:

73: }while(!condition);

74:

75:

76: //clse the scanner

77: sc.close();

78:

79:

80: }

81:

82: //one word term

83: public void oneWord(String word) throws FileNotFoundException{

84:

85: //find the word and store its information to a hashtable

86: for(List<String> list : SearchIn.AllWords){

87: if(list.contains(word)){

88: numberOfOcccurence = Collections.frequency(list, word);

89: indexOfList = SearchIn.AllWords.indexOf(list);

90: fileTotalWords = SearchIn.NumberOfWords.get(indexOfList);

91: frequencyOfWord = (float)numberOfOcccurence/(float)fileTotalWords;

92: term.put(frequencyOfWord, indexOfList);

93: }

94: }

95:

96: /// adding frequency of words to an array to sort them

97: for(Float frequencyOfWord : term.keySet()){

98: frequency.add(frequencyOfWord);

99: Collections.sort(frequency);

100: }

101:

102: //print the file name sorted by relative frequency

103: for(int i =frequency.size()-1; i > 0; i--){

104: String fileName = SearchIn.fileNames.get(term.get(frequency.get(i)));

105: String someText = SearchIn.findContext(fileName,word,null,1);

106: System.out.println(fileName +" " + someText);

107: }

108:

109: term.clear();

110: frequency.clear();

111: }

112:

113: //term with and operator

114: public void andOperator(String word, String word2) throws FileNotFoundException{

115:

116:

117: for(List<String> list : SearchIn.AllWords){

118: if(list.contains(word) && list.contains(word2)){

119: numberOfOcccurence = Collections.frequency(list, word);

120: numberOfOcccurence2 = Collections.frequency(list, word2);

121: indexOfList = SearchIn.AllWords.indexOf(list);

122: fileTotalWords = SearchIn.NumberOfWords.get(indexOfList);

123: frequencyOfWord = (float)numberOfOcccurence/(float)fileTotalWords;

124: frequencyOfWord2 = (float)numberOfOcccurence2/(float)fileTotalWords;

125: productOfrelativeFrequency = frequencyOfWord\*frequencyOfWord2;

126:

127: //put things in hashtable

128: term.put(productOfrelativeFrequency,indexOfList);

129: }

130: }

131:

132: //sort the frequencies

133: for(Float productOfrelativefrequency : term.keySet()){

134: frequency.add(productOfrelativefrequency);

135: Collections.sort(frequency);

136: }

137:

138: //print files name

139: for(int i =frequency.size()-1; i > 0; i--){

140: String fileName = SearchIn.fileNames.get(term.get(frequency.get(i)));

141: String someText = SearchIn.findContext(fileName,word,word2,2);

142: System.out.println(fileName +" "+ someText);

143:

144: }

145:

146: //clear hashtable and array after printing

147: term.clear();

148: frequency.clear();

149: }

150:

151:

152: //term with or operator

153: public void orOperator(String word, String word2){

154:

155: for(List<String> list : SearchIn.AllWords){

156: if(list.contains(word) && list.contains(word2)){

157: numberOfOcccurence = Collections.frequency(list, word);

158: numberOfOcccurence2 = Collections.frequency(list, word2);

159: indexOfList = SearchIn.AllWords.indexOf(list);

160: fileTotalWords = SearchIn.NumberOfWords.get(indexOfList);

161: frequencyOfWord = (float)numberOfOcccurence/(float)fileTotalWords;

162: frequencyOfWord2 = (float)numberOfOcccurence2/(float)fileTotalWords;

163: sumOfRelativefrequency = frequencyOfWord+frequencyOfWord2;

164:

165: //put things in hashtable

166: term.put(sumOfRelativefrequency,indexOfList);

167: }

168: }

169:

170: //sort the frequencies

171: for(Float sumOfRelativefrequency : term.keySet()){

172: frequency.add(sumOfRelativefrequency);

173: Collections.sort(frequency);

174: }

175:

176: //print files name

177: for(int i =frequency.size()-1; i >0; i--)

178: System.out.println(SearchIn.fileNames.get(term.get(frequency.get(i))));

179:

180:

181: //clear hashtable and array after printing

182: term.clear();

183: frequency.clear();

184: }

185:

186:

187: //term with not operator

188: public void notOperator(String word){

189:

190: for(List<String> list : SearchIn.AllWords){

191: if(!(list.contains(word))){

192: indexOfList = SearchIn.AllWords.indexOf(list);

193: System.out.println(SearchIn.fileNames.get(indexOfList));

194: }

195: }

196: }

197:

198:

199: //term with and not operators

200: public void andNotOperator(String word, String word2,String word3){

201:

202: for(List<String> list : SearchIn.AllWords){

203: if(list.contains(word) && list.contains(word2) && !(list.contains(word3)) ){

204: numberOfOcccurence = Collections.frequency(list, word);

205: numberOfOcccurence2 = Collections.frequency(list, word2);

206: indexOfList = SearchIn.AllWords.indexOf(list);

207: fileTotalWords = SearchIn.NumberOfWords.get(indexOfList);

208: frequencyOfWord = (float)numberOfOcccurence/(float)fileTotalWords;

209: frequencyOfWord2 = (float)numberOfOcccurence2/(float)fileTotalWords;

210: productOfrelativeFrequency = frequencyOfWord\*frequencyOfWord2;

211:

212: term.put(productOfrelativeFrequency,indexOfList);

213: }

214: }

215:

216: for(Float productOfrelativeFrequency : term.keySet()){

217: frequency.add(productOfrelativeFrequency);

218: Collections.sort(frequency);

219: }

220:

221: //print files name

222: for(int i =frequency.size()-1; i >0; i--)

223: System.out.println(SearchIn.fileNames.get(term.get(frequency.get(i))));

224:

225:

226: //clear hashtable and array after printing

227: term.clear();

228: frequency.clear();

229: }

230: }

231:

232:

233:

1: import java.io.IOException;

2:

3: public class tester {

4: public static void main(String[] args) throws IOException{

5: Search newSearch = new Search();

6:

7: //notify the user

8: System.out.println("Building...");

9:

10: //run the search engine

11: newSearch.main();

12:

13:

14: }

15: }

Output screenshots:









