Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа № 1

По дисциплине ЕЯИИС

Тема: «Разработка информационно-поисковой системы и методы оценки качества ее работы»

Выполнил:

Студент 4 курса

Группы ИИ-16

Пешко А.С.

Проверил:

Скарубо А.О.

Брест, 2021

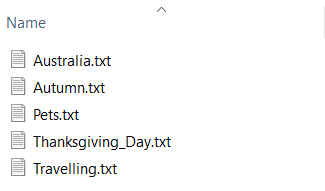
**Цель работы** освоить на практике основные принципы реализации информационно-поисковых систем и методы оценки качества их работы.

**Ход работы:**

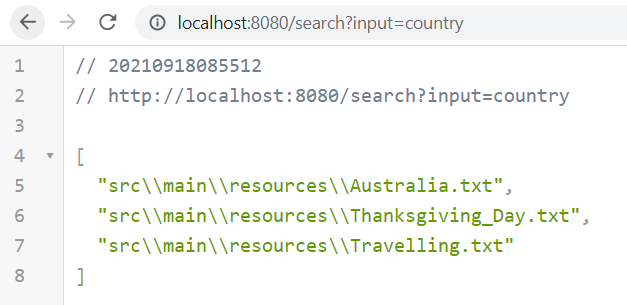
**Код программы:**  
@Component  
public class SearchService {  
  
 private final Map<File, Map<String, Double>> weightWordsInFile = new HashMap<>();  
  
 @PostConstruct  
 public void calculateParameters() throws Exception {  
  
 List<File> files = Files.walk(Paths.get("src/main/resources/"))  
 .filter(Files::isRegularFile)  
 .map(Path::toFile)  
 .collect(Collectors.toList());  
  
 Map<File, List<String>> filesWithWords = new HashMap<>();  
 List<String> allWords = files.stream().flatMap(file -> {  
 try {  
 BufferedReader br = new BufferedReader(new FileReader(file));  
 StringBuilder resultStringBuilder = new StringBuilder();  
  
 String line;  
 while ((line = br.readLine()) != null) {  
 resultStringBuilder.append(line).append(" ");  
 }  
  
 List<String> words = Arrays.asList(resultStringBuilder.toString()  
 .trim()  
 .replaceAll("\\s+", " ")  
 .split("[ ,.;:\"'»«()—!\\[\\]\n]+"));  
 filesWithWords.put(file, words);  
 return words.stream();  
 } catch (Exception var6) {  
 var6.printStackTrace();  
 return Stream.empty();  
 }  
 }).collect(Collectors.toList());  
  
 Map<String, Integer> allWordsWithCount = new HashMap<>();  
 for (String newWord : allWords) {  
 if (!allWordsWithCount.containsKey(newWord)) {  
 int count = 0;  
 for (String word : allWords) {  
 if (word.equals(newWord))  
 count++;  
 }  
 allWordsWithCount.put(newWord, count);  
 }  
 }  
  
 Map<File, Map<String, Integer>> countWordsInFile = new HashMap<>();  
  
 for (Entry<File, List<String>> entry : filesWithWords.entrySet()) {  
 List<String> listWordsInFile = entry.getValue();  
 Map<String, Integer> wordsInFile = new HashMap<>();  
  
 for (String word : listWordsInFile) {  
 int count = 1;  
 if (wordsInFile.containsKey(word)) {  
 count++;  
 }  
 wordsInFile.put(word, count);  
 }  
 countWordsInFile.put(entry.getKey(), wordsInFile);  
 }  
  
 Map<String, Integer> wordWithDocs = new HashMap<>();  
 for (String str : allWords) {  
 int count = 0;  
 wordWithDocs.put(str, count);  
 for (Entry<File, Map<String, Integer>> entry : countWordsInFile.entrySet()) {  
 if (entry.getValue().containsKey(str)) {  
 ++count;  
 wordWithDocs.put(str, count);  
 }  
 }  
 }  
  
 int N = files.size();  
 Map<String, Double> Bi = new HashMap<>();  
  
 for (String word : wordWithDocs.keySet()) {  
 Double b = Math.log((double) N / (double) wordWithDocs.get(word));  
 Bi.put(word, b);  
 }  
  
 for (Entry<File, Map<String, Integer>> entry : countWordsInFile.entrySet()) {  
  
 Map<String, Integer> wordWithCountInFile = entry.getValue();  
 Map<String, Double> weightOfWord = new HashMap<>();  
  
 for (Entry<String, Integer> wordCount : wordWithCountInFile.entrySet()) {  
 String word = wordCount.getKey();  
 Integer count = wordCount.getValue();  
 Double weight = (double) count \* Bi.get(word);  
 weightOfWord.put(word, weight);  
 }  
  
 this.weightWordsInFile.put(entry.getKey(), weightOfWord);  
 }  
 }  
  
 public List<String> findFilesByInput(String word) {  
 List<String> filesWithWord = new LinkedList<>();  
  
 for(Entry<File, Map<String, Double>> entry : weightWordsInFile.entrySet()) {  
 Map<String, Double> wordWithWeight = entry.getValue();  
 Double weight = wordWithWeight.get(word) == null ? 0.0D : wordWithWeight.get(word);  
 if (weight.compareTo(0.0D) > 0) {  
 filesWithWord.add(entry.getKey().getPath());  
 }  
 }  
 return filesWithWord;  
 }  
}

**Тестирование:**

Файловая система содержит файлы:



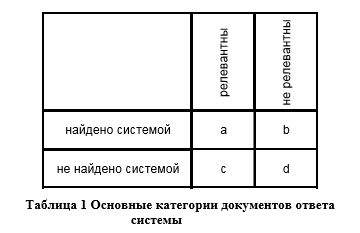
Для поиска передаем слово “country”



Система отдает найденные файлы в порядке важности:

1. Australia.txt
2. Thanksgiving\_day.txt
3. Travelling.txt

Во всех трех файлах упоминается “country”, но не все результаты поиска являются релевантными для нас (напрмер текст про День Благодарения). Таким образом “Australia.txt” и “Travelling.txt” – считаем за правильную работу системы, “Thanksgiving\_day.txt” – за неправильную. Так же в системе были файлы “Autumn.txt” и “Pets.txt”, которые не были найдены программой, но и не являютмся релевантными для нас.



Исходя из данных выше:

a = 2, b = 1, c = 0, d = 2;

Метрики:

* 1. Полнота: **r** = a/(a+c);  
     r = 2/2 = 1
  2. Точность: **p** = a/(a+b)

P = 2/3 = 0.66

* 1. Аккуратность: **accuracy** = (a+d) / (a+b+c+d)

Accuracy = 4/5 = 0.8

* 1. Ошибка: **error** = (b+c)/(a+b+c+d)

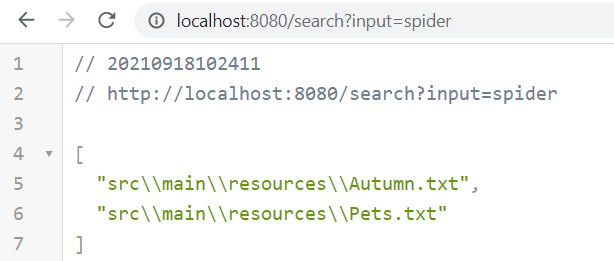
Error = 1/5 = 0.2

* 1. F-measure: **F** = 2/((1/p) + (1/r))

F = 2/(1/0.66 + 1/1) = 0.795

Для построения графика точности/полноты нам нужны ещё примеры запросов:

**2. Запрос “spider”**

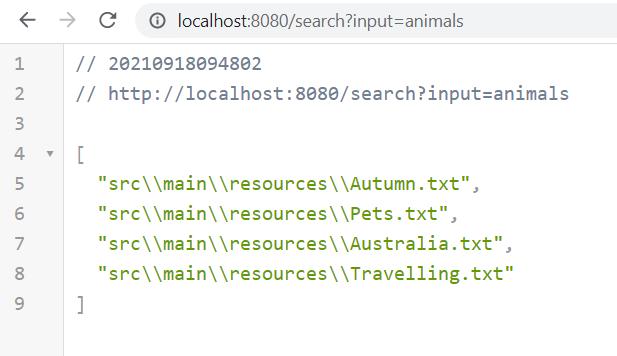


a = 1, b = 1, c = 1, d = 2;

2.1 Полнота **r** = a/(a+c) = 1/(1+1) = 0.5

2.2. Точность **p** = a/(a+b) = 1/(1+1) = 0.5

**3. Запрос “animals”**

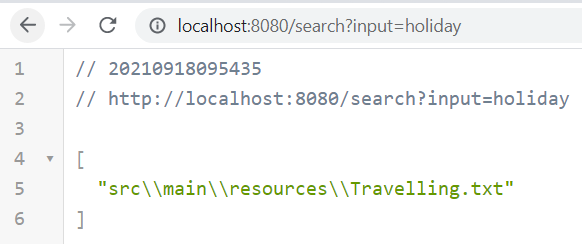


a = 1, b = 3, c = 0, d = 1;

3.1 Полнота **r** = a/(a+c) = 1/(1+0) = 1

3.2. Точность **p** = a/(a+b) = 1/(1+3) = 0.25

1. **Запрос “holiday”**

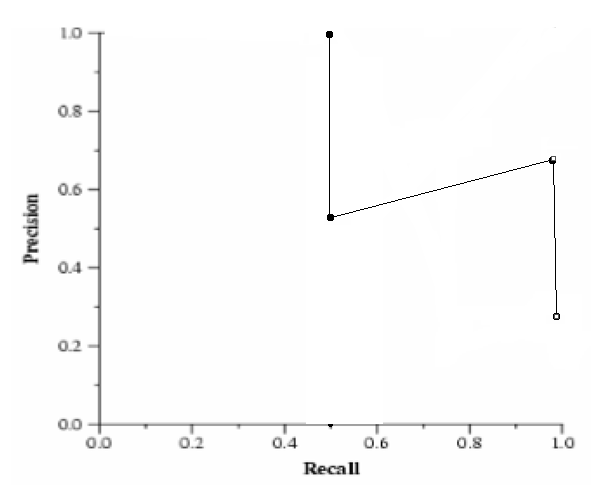


a = 1, b = 0, c = 1, d = 3;

4.1 Полнота **r** = a/(a+c) = 1/(1+1) = 0.5

4.2. Точность **p** = a/(a+b) = 1/(1+0) = 1

График:



На небольшом количестве файлов и простых запросах система показывает себя хорошо.

**Вывод:** освоили на практике основные принципы реализации информационно-поисковых систем и методы оценки качества их работы.