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

Журавлёв В.А.

Проверил:

Слинко Е.В.

Брест, 2021

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

**Код программы:**

package org.example.models;

import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.\*;

import java.util.stream.Collectors;

import java.util.stream.Stream;

@Component

public class ResourceStore {

private List<Resource> resources;

public ResourceStore() {

resources = new ArrayList<>();

}

@PostConstruct

public void fillResources() 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<>();

Map<String, Integer> wordsWithNumberOfFilesContainingWord = 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.stream(resultStringBuilder.toString().split("[ \n,.;:\"'»«()—!0-9\\[\\]]"))

.filter(word -> !word.isEmpty())

.map(String::toLowerCase)

.collect(Collectors.toList());

for (String word : words) {

wordsWithNumberOfFilesContainingWord.computeIfPresent(word, (key, value) -> value + 1);

wordsWithNumberOfFilesContainingWord.putIfAbsent(word, 1);

}

filesWithWords.put(file, words);

return words.stream();

} catch (Exception e) {

e.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);

}

}

double totalNumberOfFiles = files.size();

for (Map.Entry<File, List<String>> entry : filesWithWords.entrySet()) {

Map<String, Integer> wordsInFileWithCount = new HashMap<>();

for (String newWord : entry.getValue()) {

if (!wordsInFileWithCount.containsKey(newWord)) {

int count = 0;

for (String word : entry.getValue()) {

if (word.equals(newWord))

count++;

}

wordsInFileWithCount.put(newWord, count);

}

}

Map<String, Double> wordsWithWeight = new HashMap<>();

//слово - 5 раз (Q)

for (String word : entry.getValue()) {

double numberOfFilesWithWord = wordsWithNumberOfFilesContainingWord.get(word);

double B = Math.log(totalNumberOfFiles / numberOfFilesWithWord);

double frequencyOfWordInFile = wordsInFileWithCount.get(word);

double A = frequencyOfWordInFile \* B;

wordsWithWeight.putIfAbsent(word, Math.abs(A));

}

resources.add(Resource.builder()

.filePath(entry.getKey().getAbsolutePath())

.wordsWithWeight(wordsWithWeight)

.build());

}

}

public List<String> findMostSuitable(final String request){

String[] words = request.split("[ \n,.;:\"'»«()—!0-9\\[\\]]");

Map<Resource, Double> resourcesWithTotalWeight = new HashMap<>();

for (Resource resource : resources) {

Double resourceWeight = 0D;

Map<String, Double> wordsWithWeight = resource.getWordsWithWeight();

for (String word : words) {

word = word.toLowerCase();

if(wordsWithWeight.containsKey(word)){

resourceWeight += wordsWithWeight.get(word);

}

}

resourcesWithTotalWeight.put(resource, resourceWeight);

}

return resourcesWithTotalWeight.entrySet()

.stream()

.sorted((e1, e2) -> e2.getValue().compareTo(e1.getValue()))

.filter(entry -> entry.getValue().compareTo(0D) > 0)

.map(entry -> entry.getKey().getFilePath())

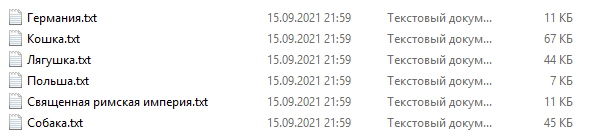
.collect(Collectors.toList());

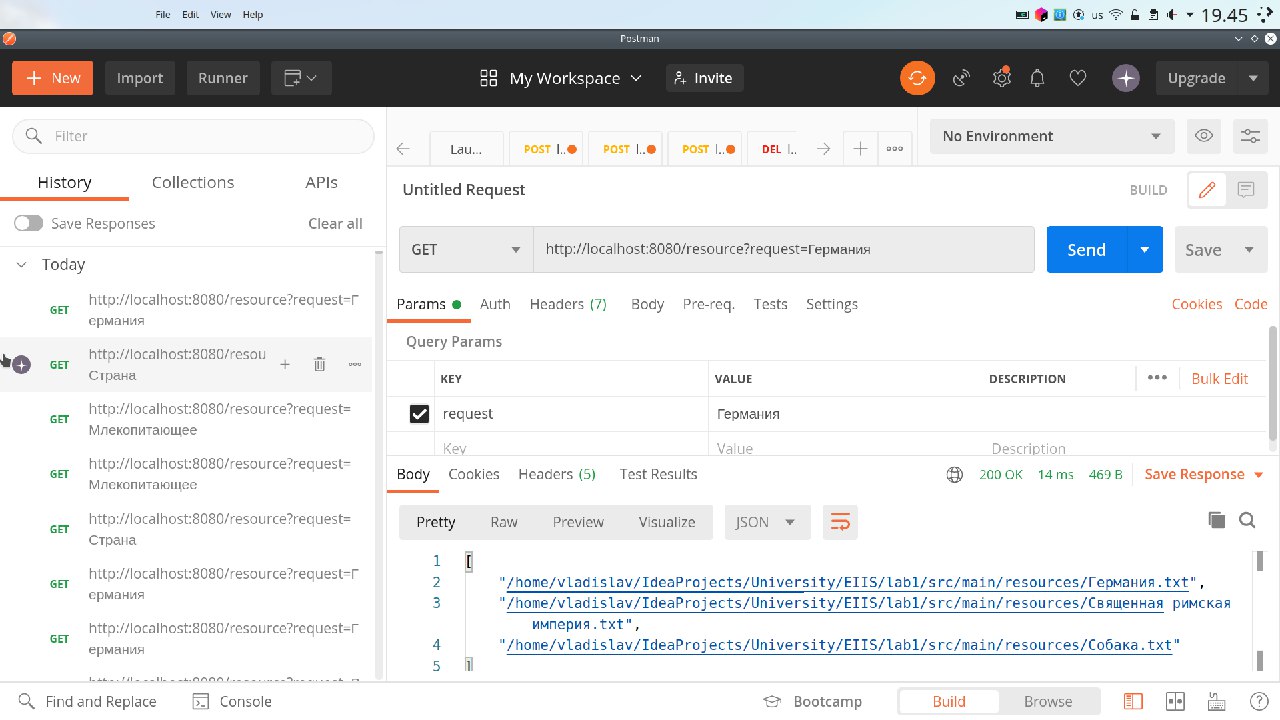
}

}

**Тестирование:**

Файловая система содержит файлы:

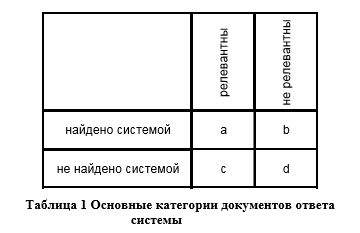


При запросе “Германия”  


Система отдает найденные файлы в порядке важности:

1. Германия.txt
2. Священная римская империя.txt
3. Собака.txt

Во всех трех файлах так или иначе упоминается “Германия”, но вряд ли мы хотели найти текст про Собак. Далее результаты “Германия.txt” и “Священная римская империя.txt” – считаем за правильную работу системы, “Собака.txt” – ошибка. Так же “Польша.txt” так же связан с Германией, и это файл можно считать за тот, который мы хотели бы получить.



Исходя из данных выше:

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

Метрики:

* 1. Полнота: r = a/(a+c);  
     r = 2/3 = 0.66
  2. Точность: p = a/(a+b)

P = 2/3 = 0.66

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

Accuracy = 4/6 = 0.66

* 1. Ошибка: error = (b+c)/(a+b+c+d)

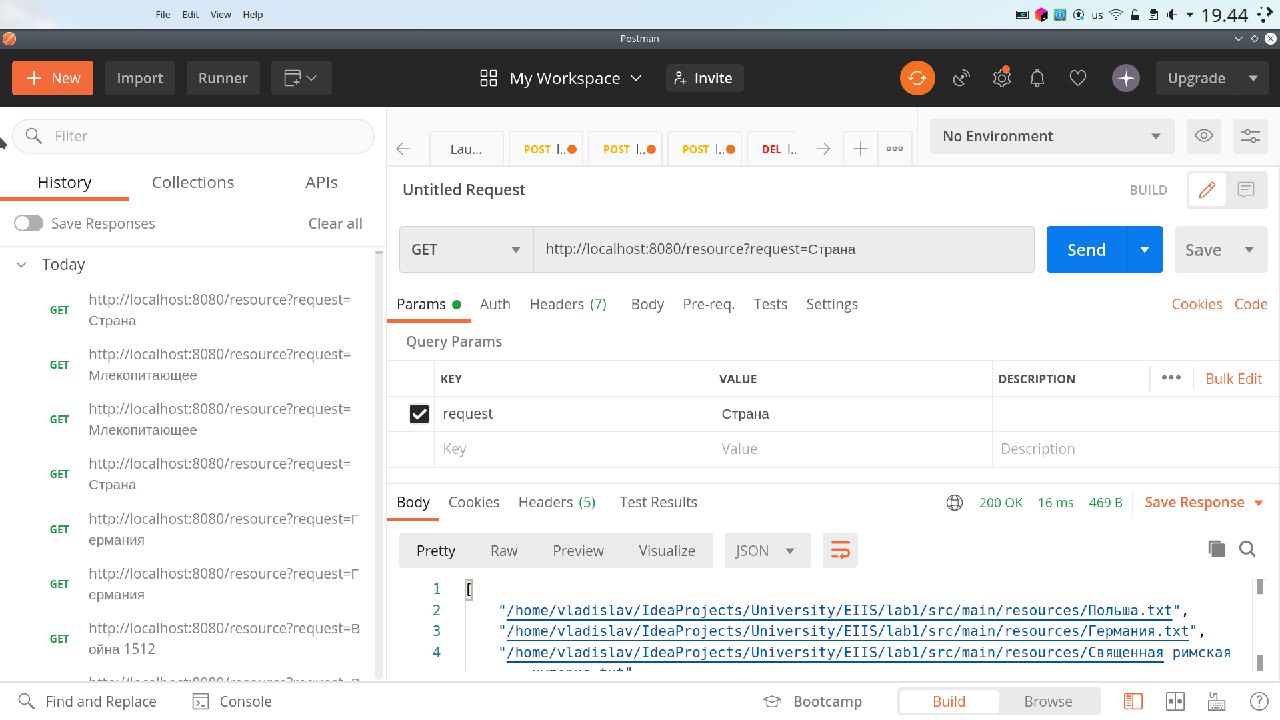
Error = 2/6 = 0.33

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

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

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

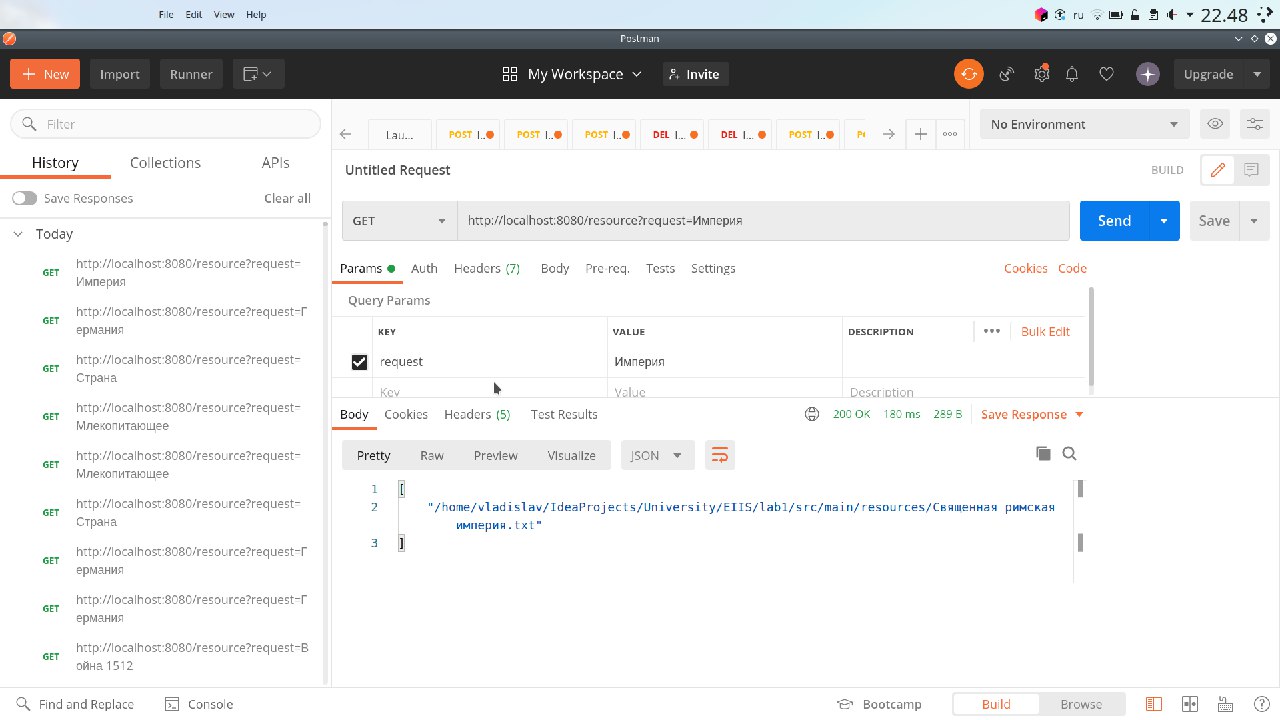
2. Запрос “Страна”



Полнота r = 3/(3+0) = 1

Точность p = 3/(3+0) = 1

3. Запрос “Империя”



Полнота r = ½ = 0.5

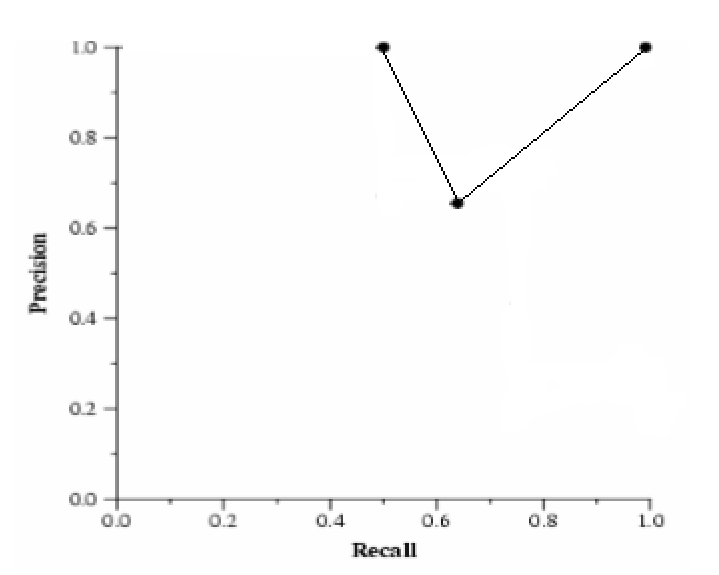
Точность p = 1/1 = 1

1. Запрос “Млекопитающее”

Полнота r = ½ = 0.5

Точность p = 1/1 = 1

График:



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

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