Введение в информационный поиск

Лабораторная работа №2

Списки словопозиций

Выполнила: студентка группы
2371 Танькова Наталья Юрьевна

Преподаватель: Зинченко А.С.

4 ноября 2020 г.

1 Ответы на вопросы

1. До внесения изменений:
   (a) В коллекции содержится 678606 термов.
   (b) Размер словаря равен 25964.
2. После добавления обработки стоп-слов:
   (a) В коллекции содержится 673104 терма.
   (b) Размер словаря равен 25812.
3. После добавления обработки термов с помощью алгоритма стемминга Портера:
   (a) В коллекции содержится 463022 терма.
   (b) Размер словаря равен 17419.

2 Результаты запросов

- Результаты запроса "goal"
  - Antony and Cleopatra Entire Play
  - King Lear Entire Play
  - Pericles Entire Play
  - Winter's Tale Entire Play
- Результаты запроса "sun"
  - All's Well That Ends Well Entire Play
  - Antony and Cleopatra Entire Play
  - As You Like It Entire Play
  - Comedy of Errors Entire Play
  - Coriolanus Entire Play
  - Cymbeline Entire Play
  - Hamlet Entire Play
  - Henry IV, part 1 Entire Play
  - Henry IV, part 2 Entire Play
  - Henry V Entire Play
  - Henry VI, part 1 Entire Play
  - Henry VI, part 2 Entire Play
  - Henry VI, part 3 Entire Play
  - Henry VIII Entire Play
  - Julius Caeser Entire Play
  - King John Entire Play
  - King Lear Entire Play
  - Love's Labour's Lost Entire Play
  - Macbeth Entire Play
  - Measure for Measure Entire Play

  - Merchant of Venice Entire Play
  - Merry Wives of Windsor Entire Play
  - Midsummer Night's Dream Entire Play
  - Much Ado About Nothing Entire Play
  - Othello Entire Play
  - Pericles Entire Play
  - Richard II Entire Play
  - Richard III Entire Play
  - Romeo and Juliet Entire Play
  - Taming of the Shrew Entire Play
  - The Tempest Entire Play
  - Timon of Athens Entire Play
  - Titus Andronicus Entire Play
  - Troiles and Cressida Entire Play
  - Twelfth Night Entire Play
  - Two Gentlemen of Verona Entire Play
  - Winter's Tale Entire Play

- Результаты запроса "king"
  - All's Well That Ends Well Entire Play
  - Antony and Cleopatra Entire Play
  - As You Like It Entire Play
  - Coriolanus Entire Play
  - Cymbeline Entire Play
  - Hamlet Entire Play
  - Henry IV, part 1 Entire Play
  - Henry IV, part 2 Entire Play
  - Henry V Entire Play
  - Henry VI, part 1 Entire Play

  - Henry VI, part 2 Entire Play
  - Henry VI, part 3 Entire Play
  - Henry VIII Entire Play
  - Julius Caeser Entire Play
  - King John Entire Play
  - King Lear Entire Play
  - Love's Labour's Lost Entire Play
  - Macbeth Entire Play
  - Measure for Measure Entire Play
  - Merchant of Venice Entire Play
  - Merry Wives of Windsor Entire Play
  - Midsummer Night's Dream Entire Play
  - Much Ado About Nothing Entire Play
  - Othello Entire Play
  - Pericles Entire Play
  - Richard II Entire Play
  - Richard III Entire Play
  - Romeo and Juliet Entire Play
  - Taming of the Shrew Entire Play
  - The Tempest Entire Play
  - Timon of Athens Entire Play
  - Titus Andronicus Entire Play
  - Troiles and Cressida Entire Play
  - Twelfth Night Entire Play
  - Two Gentlemen of Verona Entire Play
  - Winter's Tale Entire Play
- Результаты запроса стоп-слова "about"
  - All's Well That Ends Well Entire Play

– Antony and Cleopatra Entire Play
– As You Like It Entire Play
– Comedy of Errors Entire Play
– Coriolanus Entire Play
– Cymbeline Entire Play
– Hamlet Entire Play
– Henry IV, part 1 Entire Play
– Henry IV, part 2 Entire Play
– Henry V Entire Play
– Henry VI, part 1 Entire Play
– Henry VI, part 2 Entire Play
– Henry VI, part 3 Entire Play
– Henry VIII Entire Play
– Julius Caeser Entire Play
– King John Entire Play
– King Lear Entire Play
– Love's Labour's Lost Entire Play
– Macbeth Entire Play
– Measure for Measure Entire Play
– Merchant of Venice Entire Play
– Merry Wives of Windsor Entire Play
– Midsummer Night's Dream Entire Play
– Much Ado About Nothing Entire Play
– Othello Entire Play
– Pericles Entire Play
– Richard II Entire Play
– Richard III Entire Play
– Romeo and Juliet Entire Play
– Taming of the Shrew Entire Play

  - The Tempest Entire Play
  - Timon of Athens Entire Play
  - Titus Andronicus Entire Play
  - Troiles and Cressida Entire Play
  - Twelfth Night Entire Play
  - Two Gentlemen of Verona Entire Play
  - Winter's Tale Entire Play
- Результаты запроса "Romeo and Juliet"
  - Romeo and Juliet Entire Play
- Результаты запроса "sun and road and goal"
  - Antony and Cleopatra Entire Play
  - King Lear Entire Play
  - Pericles Entire Play
  - Winter's Tale Entire Play
- Результаты запроса "Romeo and Juliet and king"
  - Romeo and Juliet Entire Play
- Результаты запроса "fled and thrown and leaden and twain"
  - Othello Entire Play

3 Программный код

```
[language=Java, caption={Индексация коллекции документов},label=invertedindex]
package com.company;

import org.jsoup.Jsoup;
import org.jsoup.safety.Whitelist;
import org.jsoup.nodes.Document;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.*;

class Pair {
    int termFrequency;
    LinkedList<Integer> list;

    public Pair(int docID) {
        termFrequency = 1;
        list = new LinkedList<Integer>();
        list.add(docID);
    }

    void addDocument(int docID) {
        this.termFrequency++;
        if (this.list.getLast() != docID) {
            this.list.add(docID);
        }
    }

    void print() {
        for (Iterator I = this.list.iterator();
             I.hasNext();) {
            System.out.print(I.next() + " ");
        }
        System.out.println("");
    }
}
```

```
public class InvertedIndex {
    List<String> documents = new ArrayList<String>();
    Map<String, Pair> index = new HashMap<>();
    int countTokens = 0;
    LinkedList<String> stopWords;

    public InvertedIndex(String path) throws FileNotFoundException {
        stopWords = new LinkedList();
        File file = new File(path);
        Scanner in = new Scanner(file);
        while (in.hasNext()) {
            stopWords.add(in.next());
        }
    }

    public void indexDocument(String path) throws IOException {
        if (!documents.contains(path)) {
            Integer docId = documents.size();
            documents.add(docId, path);
            //Document doc = (Document) Jsoup.parse(file, "UTF-8");
            File input = new File(path);
            Document doc = (Document) Jsoup.parse(input, "UTF-8");
            String content = doc.body().text().toLowerCase();
            String[] words = content.split("[^a-zA-Z0-9_']+");
            boolean isExist = false;
            for (int i = 0; i < words.length; i++) {
                Pair idx = index.get(words[i]);
                isExist = false;
                Stemmer stemmer = new Stemmer();
                stemmer.add(words[i].toCharArray(), words[i].length());
                stemmer.stem();
                String term = stemmer.toString();
                if(idx == null){
                    idx = new Pair(docId);
                    if (!this.stopWords.contains(term)) {
                        index.put(term, idx);
                        isExist = true;
```

```
                }
            }
            if (!this.stopWords.contains(term) &&(isExist == true ||
                    idx.list.getLast() != docId)){
                idx.addDocument(docId);
            }
        }
        countTokens += index.size();
        System.out.printf("| %2d | %60s | %5d |%n", docId, path,
                index.size());
    }
}

public void indexCollection(String folder) throws IOException {
    File dir = new File(folder);
    String[] files = dir.list();
    for (int i = 0; i < files.length; i++) {
        this.indexDocument(folder + "\\" + files[i]);
    }
}

public static LinkedList<Integer> getIntersection(
        List<Integer> list1, List<Integer> list2) {
    LinkedList<Integer> intersection = new LinkedList();
    Iterator<Integer> i = list1.iterator(), j = list2.iterator();
    int element1 = i.next();
    int element2 = j.next();
    int k = 2;
    int size = list1.size() + list2.size();
    do {
        if (element1 == element2) {
            if (intersection.isEmpty()) {
                intersection.add(element1);
            } else if (intersection.getLast() != element1) {
                intersection.add(element1);
            }
            if (i.hasNext()) {
                element1 = i.next();
```

```
            }
            if (j.hasNext()) {
                element2 = j.next();
            }
        }
        if (element1 < element2 && i.hasNext()) {
            element1 = i.next();
        }
        if (element2 < element1 && j.hasNext()) {
            element2 = j.next();
        }
        k++;
    } while (k <= size);
    return intersection;
}

public List<Integer> getAllDoc(){
    LinkedList<Integer> list = new LinkedList<>();
    for (int i = 0; i < documents.size(); i++){
        list.add(i);
    }
    return list;
}
public List<Integer> executeQuery(String query) {
    query = query.toLowerCase();
    String s[] = query.split(" and ");
    List<Integer> documents = new LinkedList();
    if(s.length<2)
        if(stopWords.contains(query))
            documents = getAllDoc();
        else
            documents = this.index.get(query).list;
    else if(this.index.get(s[0]) != null &&
            this.index.get(s[1]) != null){
        int i = 2;
        if(stopWords.contains(s[0])){
            if (stopWords.contains(s[1])){
                documents = getAllDoc();
```

```
            } else {
                documents = index.get(s[1]).list;
            }
        } else
            documents = this.getIntersection(this.index.get(s[0]).list,
                    this.index.get(s[1]).list);

        while(i < s.length && documents != null){
            if (!stopWords.contains(s[i]))
                documents = this.getIntersection(documents,
                        this.index.get(s[i]).list);
            i++;
        }
    }
    return documents;
}

public void printDocuments(List query){
    Integer docId;
    if(query!=null){
        for(Iterator i = query.iterator();i.hasNext();){
            docId = (Integer)i.next();
            String s = this.documents.get(docId);
            System.out.println(s.substring(s.indexOf('\\') + 1,
                    s.indexOf('.')));
        }
    }
}

public static void main(String[] args) throws IOException {
    InvertedIndex myIndex = new InvertedIndex("stop_words.txt");
    myIndex.indexCollection("collection_html");
    System.out.println("Count tokens " + myIndex.countTokens);
    System.out.println("Size " + myIndex.index.size());
    String query1 = "goal";
    String query2 = "sun";
    String query3 = "king";
    String query4 = "about";
```

```
        String query5 = "romeo and juliet";
        String query6 = "goal and sun and about";
        String query7 = "romeo and juliet and king";
        String query8 = "fled and thrown and leaden and twain";
        System.out.println("");
        System.out.println(query1);
        myIndex.printDocuments(myIndex.executeQuery(query1));
        System.out.println("");
        System.out.println(query2);
        myIndex.printDocuments(myIndex.executeQuery(query2));
        System.out.println("");
        System.out.println(query3);
        myIndex.printDocuments(myIndex.executeQuery(query3));
        System.out.println("");
        System.out.println(query4);
        myIndex.printDocuments(myIndex.executeQuery(query4));
        System.out.println("");
        System.out.println(query5);
        myIndex.printDocuments(myIndex.executeQuery(query5));
        System.out.println("");
        System.out.println(query6);
        myIndex.printDocuments(myIndex.executeQuery(query6));
        System.out.println("");
        System.out.println(query7);
        myIndex.printDocuments(myIndex.executeQuery(query7));
        System.out.println("");
        System.out.println(query8);
        myIndex.printDocuments(myIndex.executeQuery(query8));
    }
}
```