Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Фізико-технічний інститут

**ЛАБОРАТОРНА РОБОТА №1**

З дисципліни«Криптографія»

«Експериментальна оцінка ентропії на символ джерела відкритого тексту»

Виконали:

студенти 3 курсу ФТІ

групи ФБ-73  
Дем’яненко Д.

Проноза А.

Перевірив:

Чорний О.

**Мета роботи:**

Засвоєння понять ентропії на символ джерела та його надлишковості, вивчення та порівняння різних моделей джерела відкритого тексту для наближеного визначення ентропії, набуття практичних навичок щодо оцінки ентропії на символ джерела.

**Порядок виконання роботи:**

0. Уважно прочитати методичні вказівки до виконання комп’ютерного практикуму.

1. Написати програми для підрахунку частот букв і частот біграx`м в тексті, а також підрахунку та за безпосереднім означенням. Підрахувати частоти букв та біграм, а також значення та на довільно обраному тексті російською мовою достатньої довжини (щонайменше 1Мб), де імовірності замінити відповідними частотами. Також одержати значення та на тому ж тексті, в якому вилучено всі пробіли.

2. За допомогою програми CoolPinkProgram оцінити значення , , .

3. Використовуючи отримані значення ентропії, оцінити надлишковість російської мови в різних моделях джерела.

**Хід роботи:**

1)Прочитали методичні вказівки до виконання лабораторної роботи.

2) За допомогою програми CoolPinkProgram оцінили значення , , і запушили відповідні скріншоти на github (пункт 2).

3)Створили додаток у IntelliJ IDEA для виконання завдання 1: написати програму для підрахунку частот букв і частот біграм в тексті, а також підрахунку та за безпосереднім означенням.

4)Запушили код на github

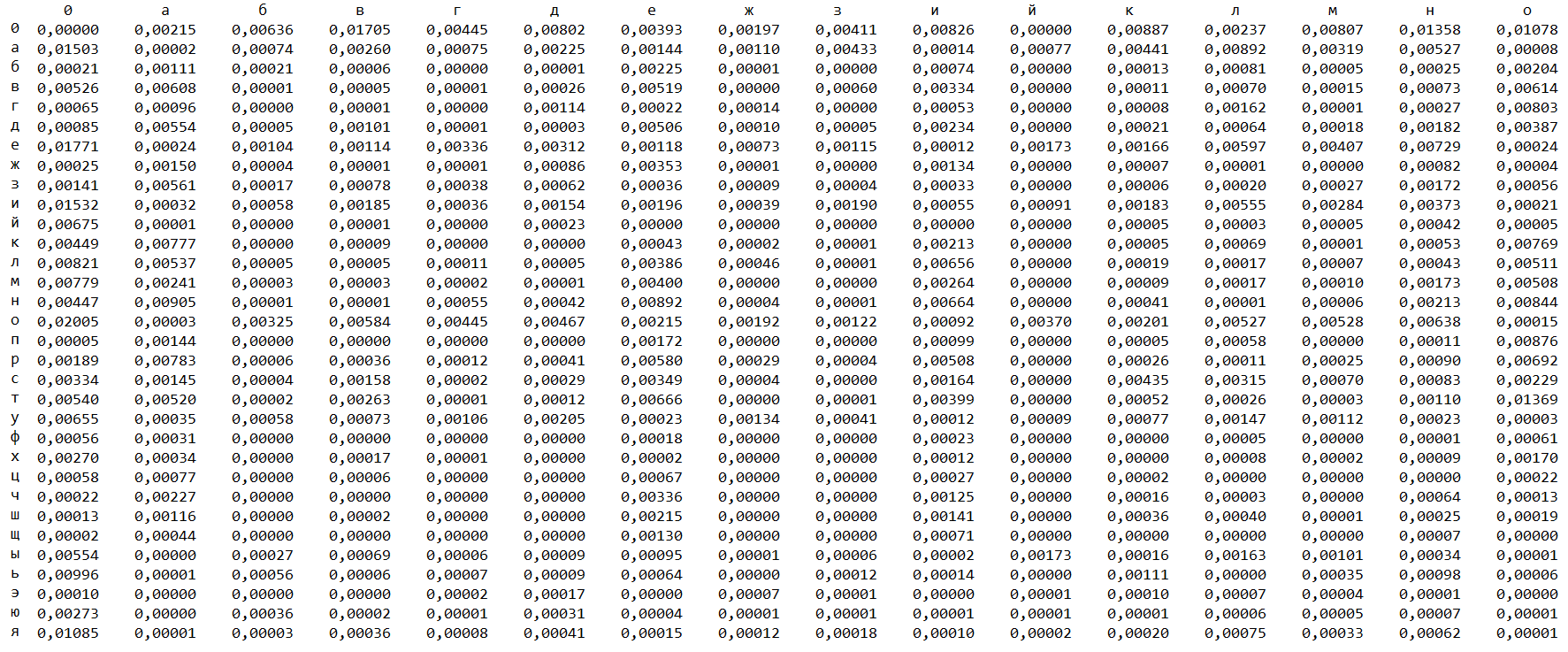
5) Використовуючи отримані значення ентропії, оцінили надлишковість російської мови в різних моделях джерела.

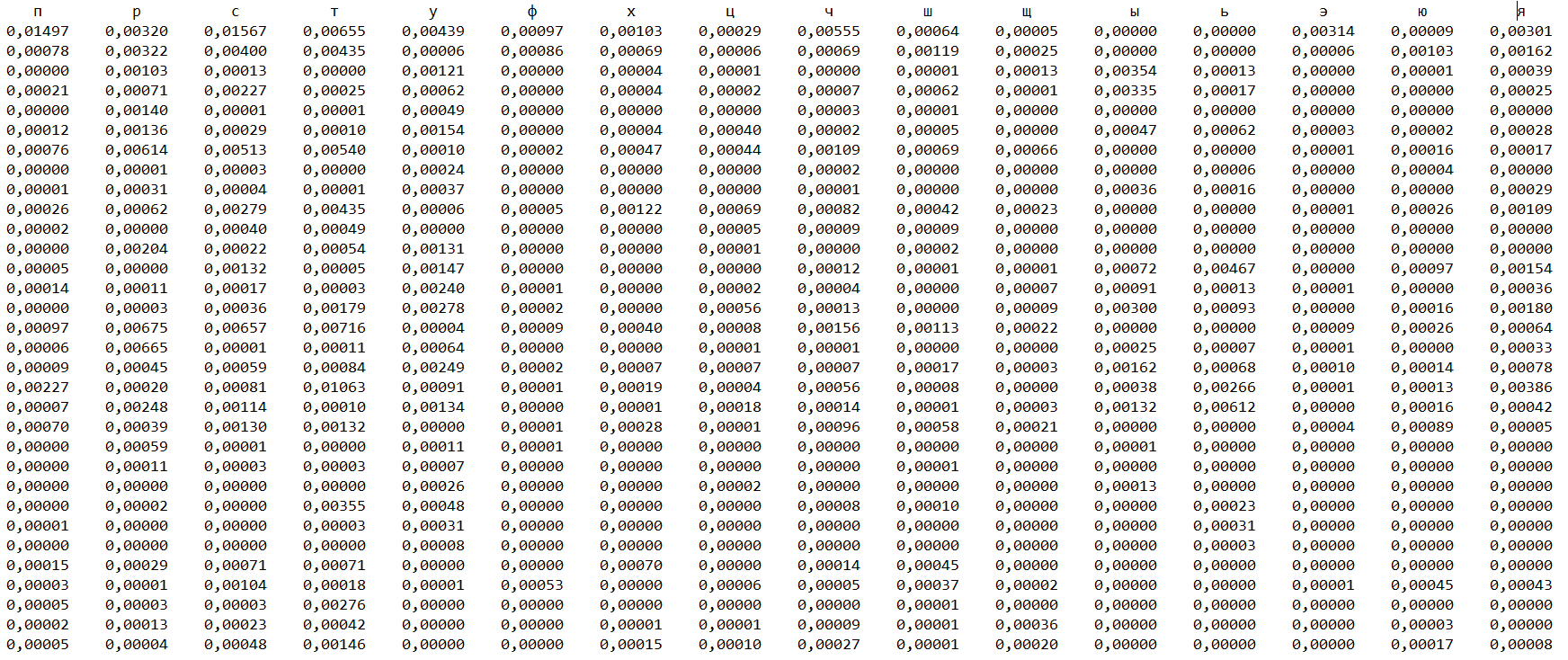
|  |  |
| --- | --- |
| **Ентропія** | |
| Для 1 символу з пробілом | =4.384479655591922 |
| Для 1 символу без пробілу | =4.462782106749086 |
| Для біграми з пробілом з кроком 2 | =3.9815947603632704 |
| Для біграми без пробілу з кроком 2 | = 4.145346121275769 |
| Для біграми з пробілом з кроком 1 | =3.9815564729306763 |
| Для біграми без пробілу з кроком 1 | = 4.145375747460906 |
| Ідеальна ентропія з пробілом | =21.090792787932422 |
| Ідеальна ентропія без пробілу | =20.84044448553777 |

|  |  |
| --- | --- |
| **Надлишковість російської мови** | |
| Для символів з пробілами | R=0.792114042384381 |
| Для символів без пробілів | R=0.7858595525711539 |
| Дял біграм з пробілами з кроком 2 | R=0.8112164487889033 |
| Для біграм без пробілів з кроком 2 | R=0.8010912807472781 |
| Для біграм з пробілами з кроком 1 | R= 0.8112182641513213 |
| Для біграм без пробілу з кроком 1 | R= 0.8010898591756241 |

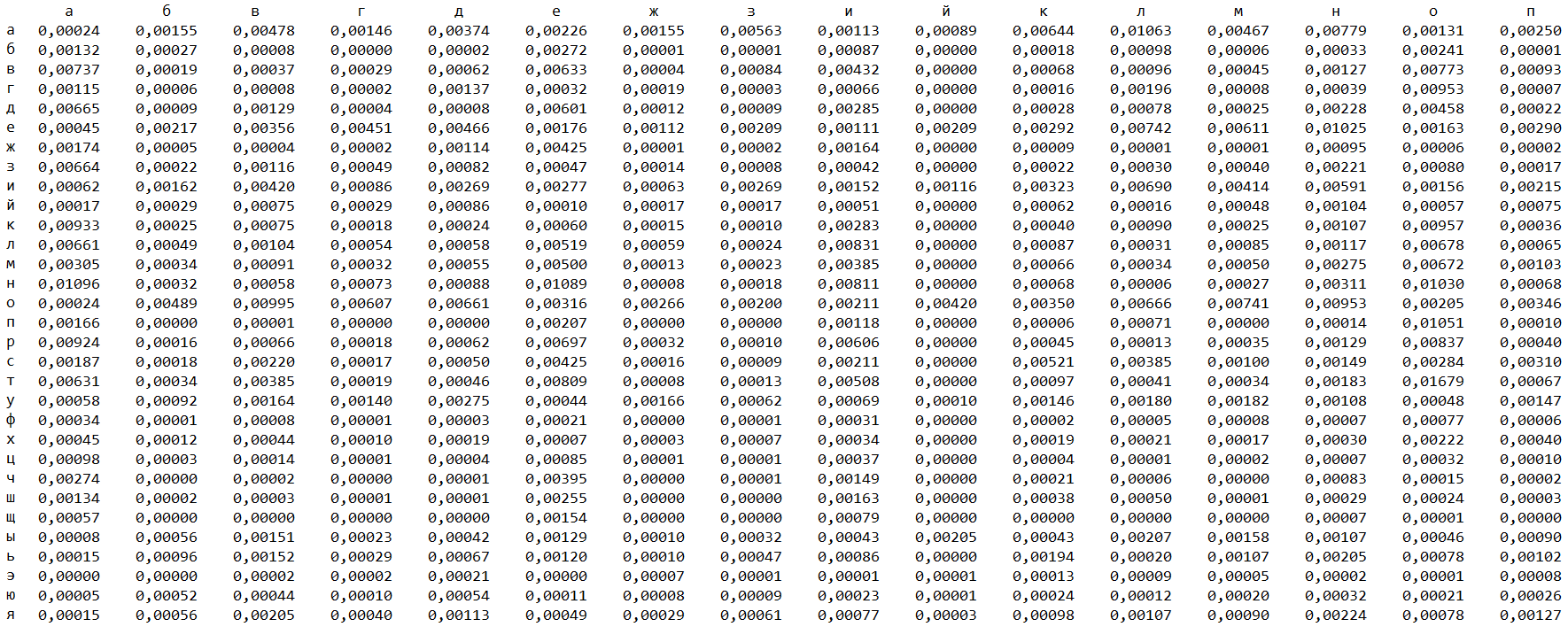
|  |  |  |
| --- | --- | --- |
| **Частоти літер** | | |
|  | З пробілом | Без пробілу |
| 0 | 0,159306574 |  |
| О | 0,093166256 | 0,110820726 |
| Е | 0,071908723 | 0,085535013 |
| А | 0,069836969 | 0,083070673 |
| Т | 0,053201592 | 0,063282988 |
| Н | 0,053071744 | 0,063128534 |
| И | 0,052689810 | 0,062674227 |
| С | 0,045863812 | 0,054554741 |
| Л | 0,041706871 | 0,049610084 |
| Р | 0,038422153 | 0,045702931 |
| В | 0,037241428 | 0,044298465 |
| М | 0,028416212 | 0,033800921 |
| К | 0,028182932 | 0,033523436 |
| Д | 0,027149518 | 0,032294195 |
| У | 0,023826293 | 0,028341239 |
| П | 0,021819463 | 0,025954126 |
| Я | 0,017310588 | 0,020590845 |
| Ь | 0,017146263 | 0,020395381 |
| Ы | 0,015892106 | 0,018903569 |
| Г | 0,015790914 | 0,018783201 |
| Б | 0,014485713 | 0,017230672 |
| З | 0,014226016 | 0,016921764 |
| Ч | 0,012566642 | 0,014947949 |
| Й | 0,008880738 | 0,010563587 |
| Ж | 0,008871783 | 0,010552935 |
| Ш | 0,006717642 | 0,007990596 |
| Х | 0,005439306 | 0,006470024 |
| Ю | 0,005008568 | 0,005957663 |
| Э | 0,003522027 | 0,004189431 |
| Ц | 0,003069796 | 0,003651505 |
| Ф | 0,002652938 | 0,003155654 |
| Щ | 0,002608610 | 0,003102927 |

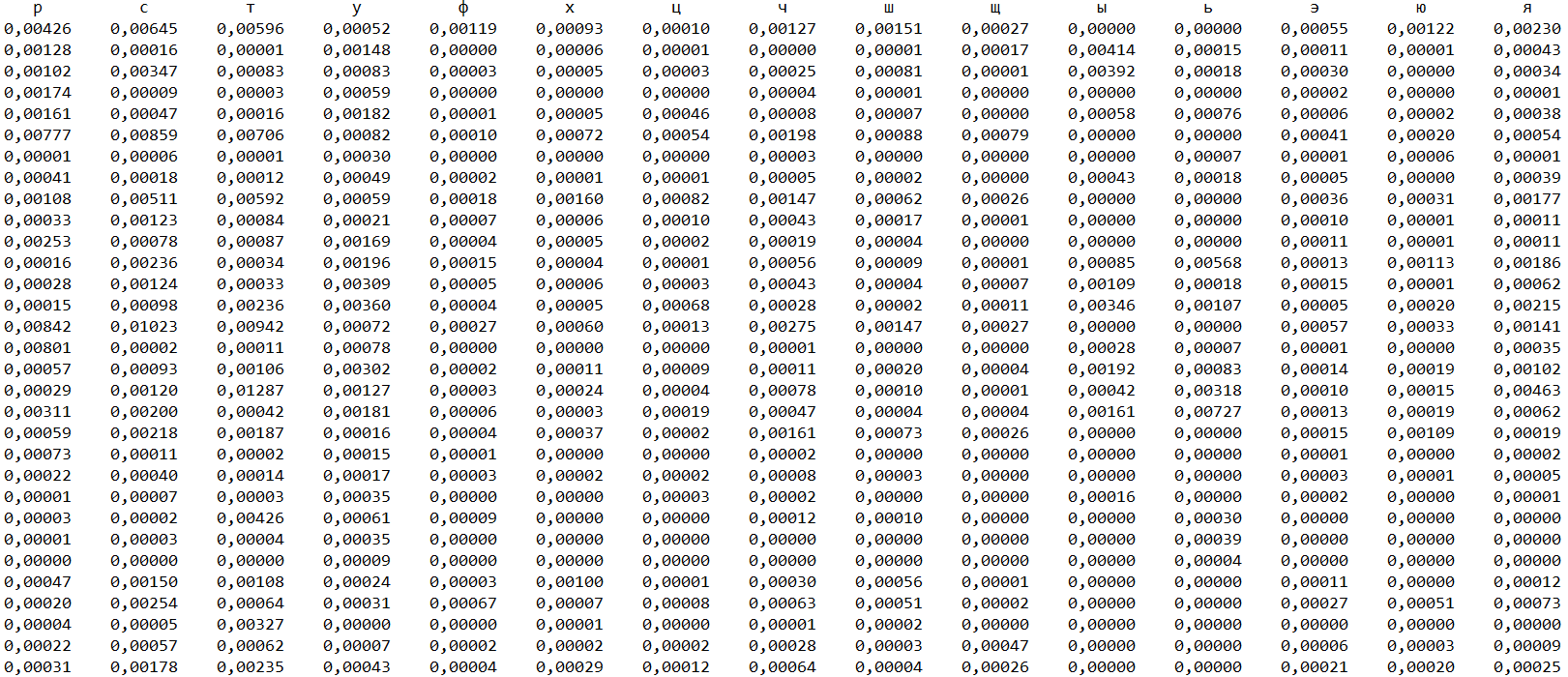
**Частота біграм з пробілом і кроком 2**

****

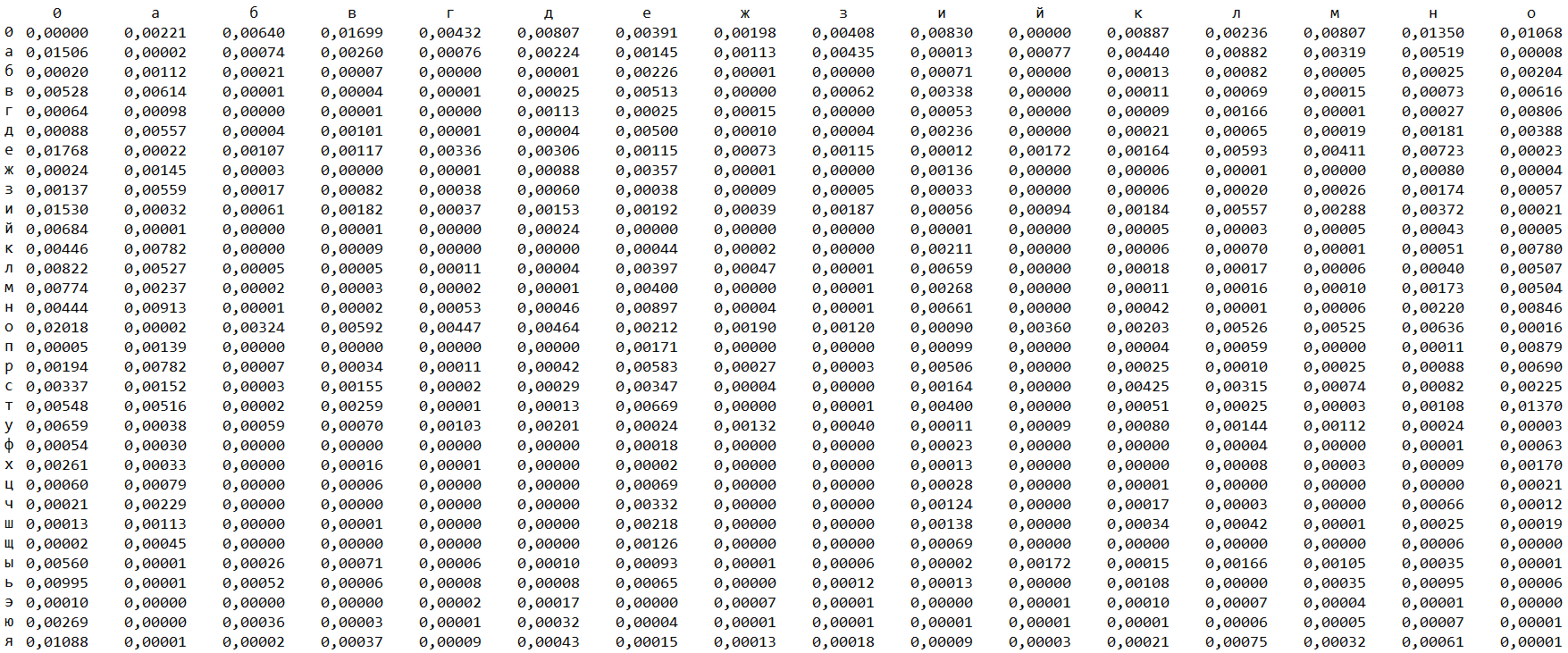
****

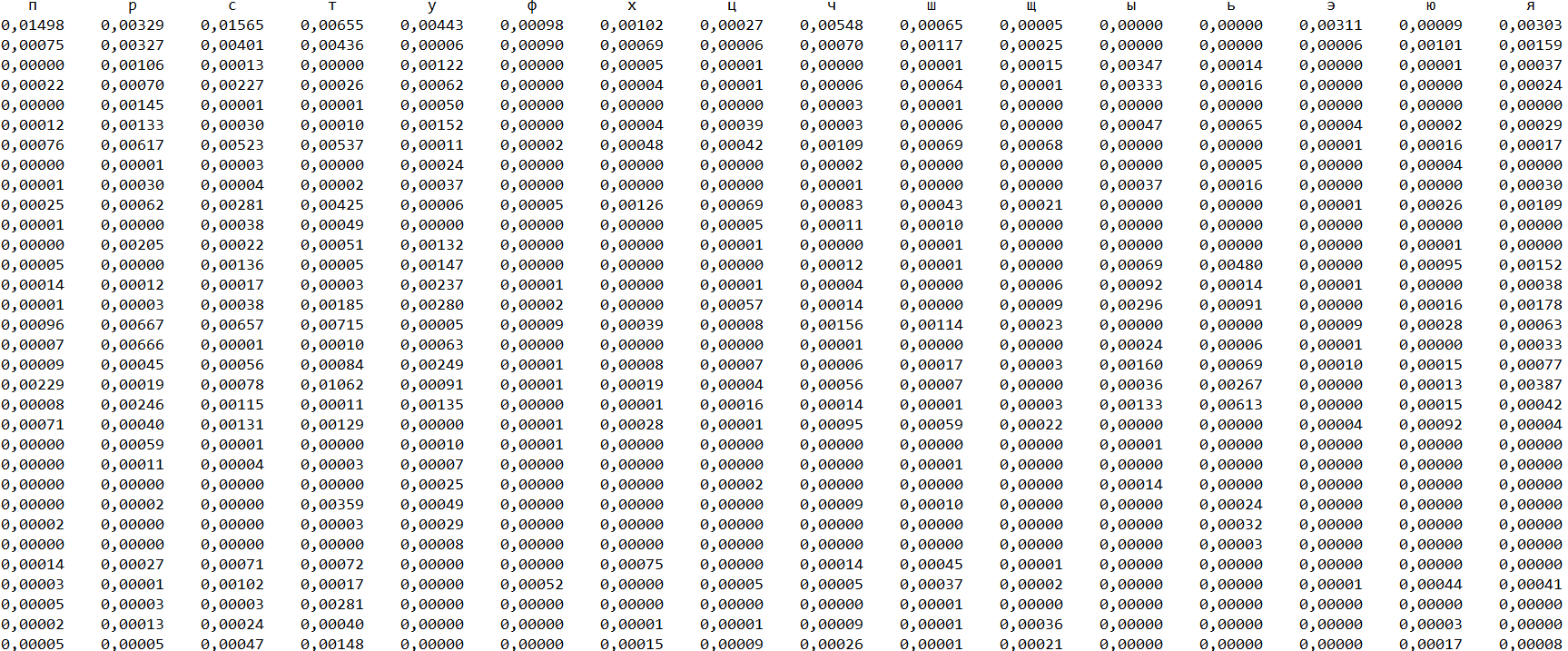
**Частота біграм без пробілу і кроком 2**

****

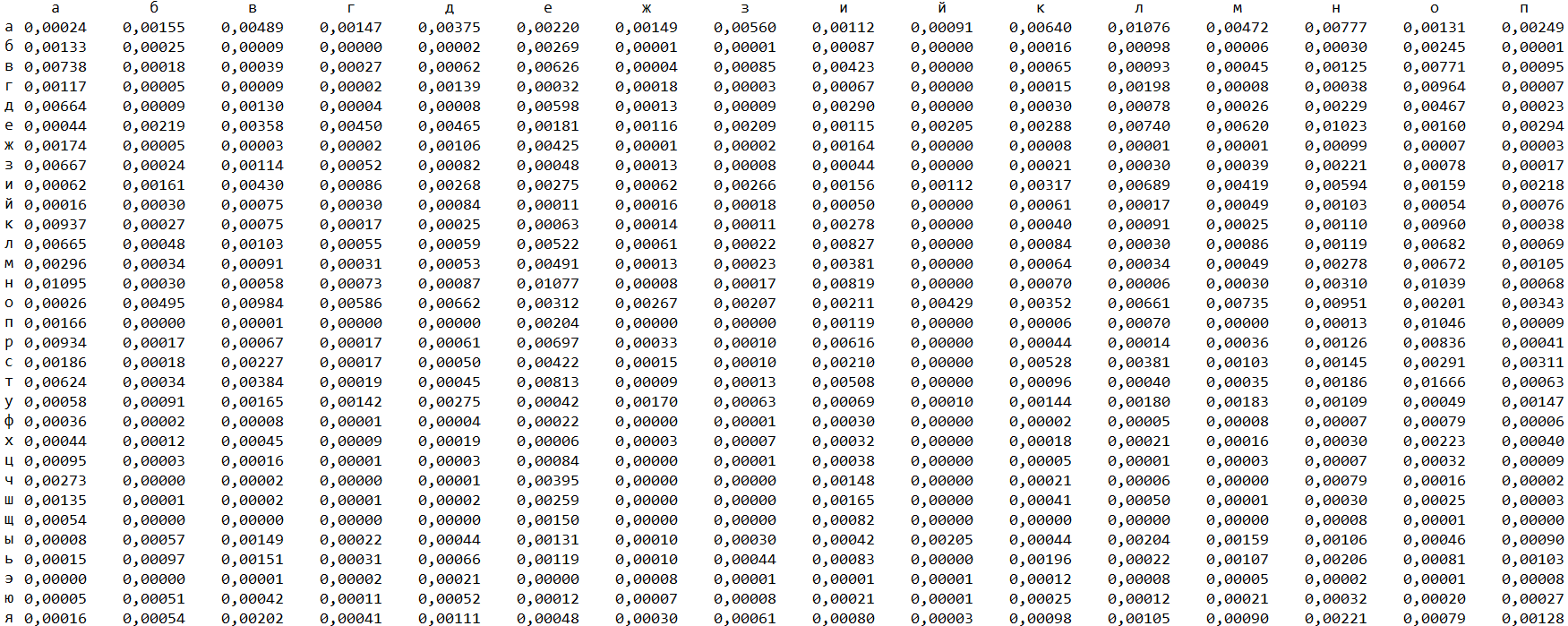
****

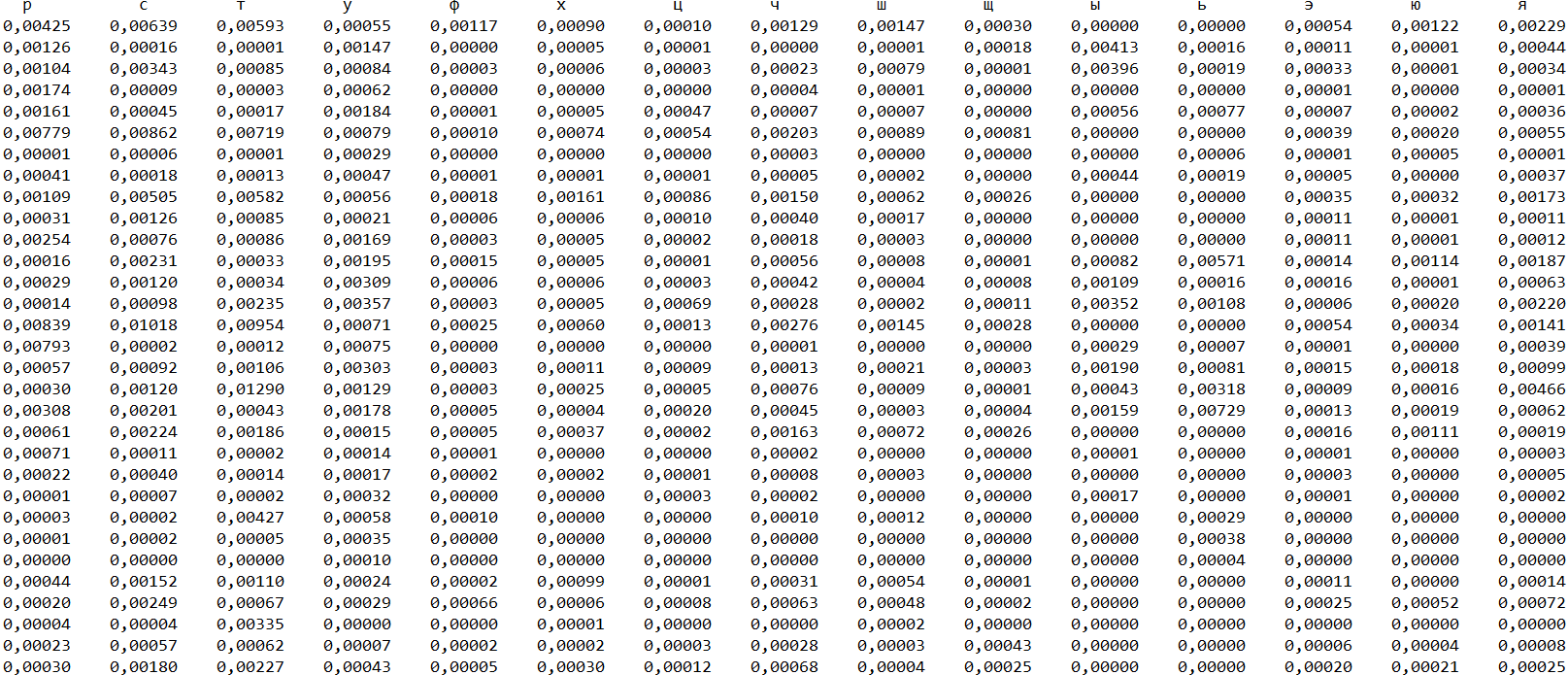
Частота біграм з пробілом і кроком 1

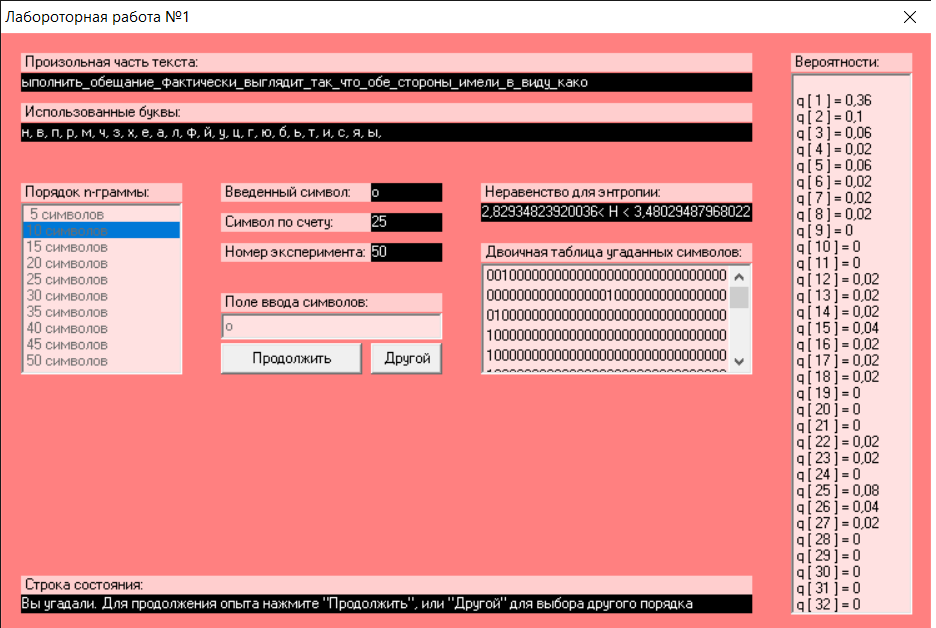




Частота біграм без пробілу з кроком 1



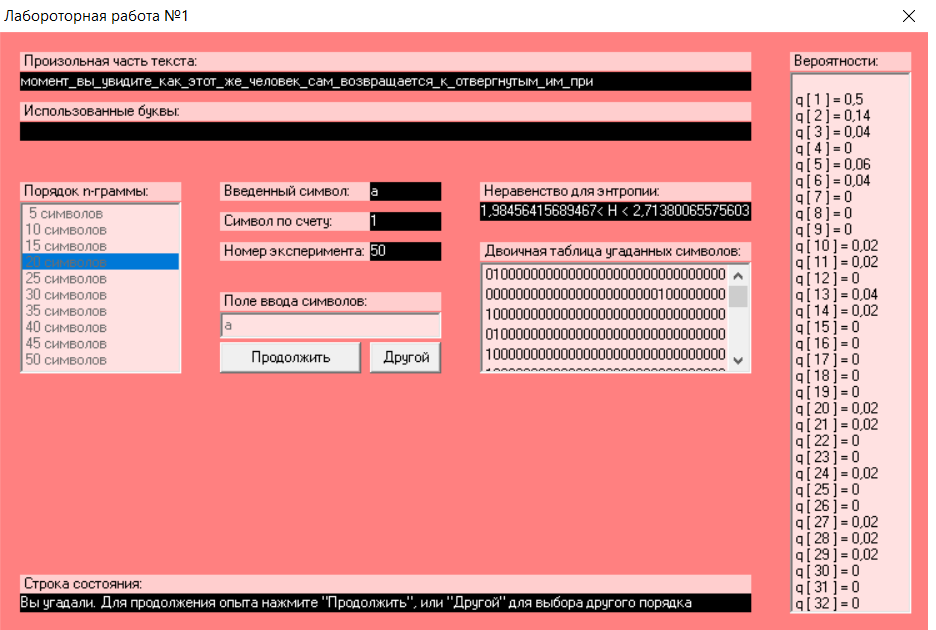


**Умовна ентропія джерела для 10 символів:**

Надлишковість російської мови 10 символів

0,3039412<=R<= 0,4341304‬

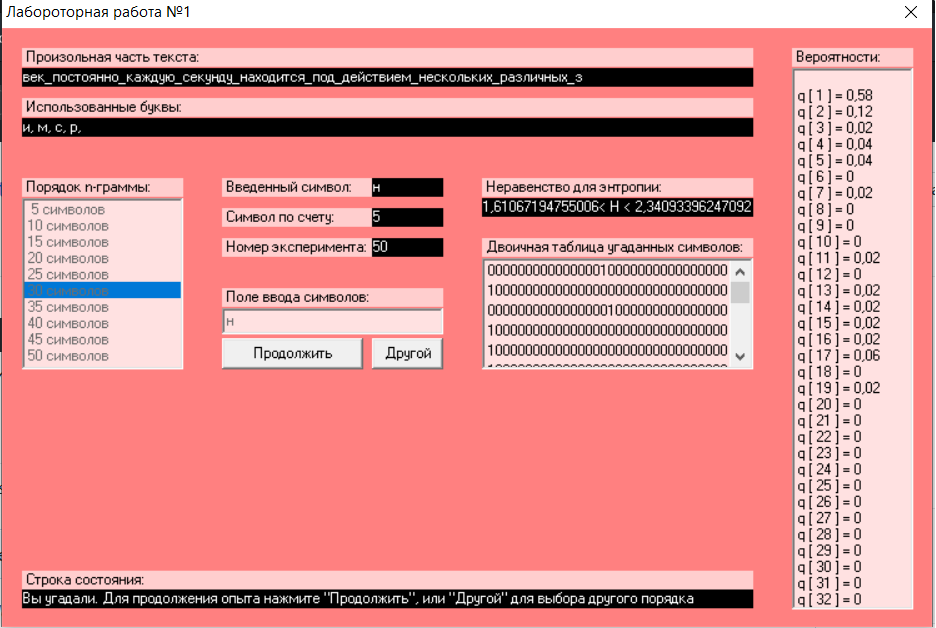
**Умовна ентропія джерела для 20 символів:**



Надлишковість російської мови 20 символів

0,45724<=R<= ‬0,6030872

**Умовна ентропія джерела для 30 символів:**



Надлишковість російської мови 30 символів

0,5318134‬<=R<= 0,6778658

**Код**

import java.io.FileReader;

import java.io.IOException;

import java.text.DecimalFormat;

import java.util.\*;

public class Main {

private final static int CAPACITY = 32;

private static Map<Character, Integer> alphabet = new TreeMap<>();

private static Map<Character, Integer> alphabetWithoutSpace = new TreeMap<>();

private static Map<String, Integer> alphabetForBigramm = new TreeMap<>();

private static Map<String, Integer> alphabetForBigrammWithouSpace = new TreeMap<>();

private static Map<String, Integer> alphabetForBigrammSingleStep= new TreeMap<>();

private static Map<String, Integer> alphabetForBigrammWithouSpaceSingleStep = new TreeMap<>();

private static StringBuffer getFileContent(){

StringBuffer fileData = new StringBuffer();

try(FileReader reader = new FileReader("graf-monte-kristo.txt")){

int c;

while((c=reader.read())!=-1){

if(((c >= 1072) && (c<=1097))||((c >= 1099)&&(c <= 1103))||(c == 32)||((c >= 1040) && (c<=1065))||((c >= 1067)&&(c <= 1071))) {

if (((c >= 1040) && (c<=1065))||((c >= 1067)&&(c <= 1071)))

c += 32;

if (c == ' ') {

c = '0';

if (fileData.charAt(fileData.length() - 1) == '0')

continue;

}

fileData.append((char) c);

}

}

}catch(IOException ex){

System.out.println(ex.getMessage());

}

return fileData;

}

private static void initAlphabet(StringBuffer fileData, Map<Character, Integer> alphabet){

for (int i=0; i<fileData.length(); i++){

char symbol = fileData.charAt(i);

int temp = alphabet.getOrDefault(symbol, 0);

temp++;

alphabet.put(symbol, temp);

}

}

private static void initAlphabetForBigram(StringBuffer fileData, Map<String, Integer> alphabet, int step){

for (int i=0; i<fileData.length()-3; i+=step){

String bigram = fileData.substring(i, i+2);

int temp = alphabet.getOrDefault(bigram, 0);

temp++;

alphabet.put(bigram, temp);

}

}

private static void deleteSpaceFromBuffer(StringBuffer fileData, StringBuffer fileDataWithouSpace){

for (int i=0; i<fileData.length(); i++){

if (fileData.charAt(i) != '0')

fileDataWithouSpace.append(fileData.charAt(i));

}

}

private static void printMap(String desc, Map map){

System.out.println();

System.out.println(desc);

System.out.println(map);

}

private static void printAlphabetMap(String desc, Map<Character, Integer> map, int total){

System.out.println();

System.out.println(desc);

Map<Character, String> mapFrequency = new TreeMap<>();

for (Map.Entry<Character, Integer> entry : map.entrySet())

mapFrequency.put(entry.getKey(), new DecimalFormat("#0.00000").format((double) entry.getValue()/total));

System.out.println(mapFrequency);

List list = new ArrayList(mapFrequency.entrySet());

list.sort((Comparator<Map.Entry<Character, String>>) (a, b) -> b.getValue().compareTo(a.getValue()));

System.out.println("letter frquency sorted by value: " + list);

}

private static void initArray(String[][] array, String[] alpha, Map<String, Integer> alphabet){

for(int column = 1; column<array.length; column++){

array[0][column] = alpha[column-1];

}

for(int row = 1; row <array.length; row++){

array[row][0] = alpha[row-1];

}

array[0][0] = " ";

for (int row =1; row<array.length; row++){

for (int column =1; column<array.length; column++){

String key = array[row][0] + array[0][column];

String result;

if (alphabet.get(key) != null)

result = Integer.toString(alphabet.get(key));

else

result = Integer.toString(0);

array[row][column] = result;

}

}

}

private static void showArray(String[][] array, int total){

System.out.println();

System.out.print(" ");

for (int i = 0; i < array.length; i++) {

for (int j = 0; j <array.length; j++) {

if (i >=1 && j>=1) {

String formattedDouble = new DecimalFormat("#0.00").format(0.1321231);

System.out.print(new DecimalFormat("#0.00000").format((double) Integer.parseInt(array[i][j]) / total) + " ");

}

else

System.out.print(array[i][j] + " ");

}

System.out.println();

}

}

private static int countAmountBigram(StringBuffer fileData, int step){

int result = 0;

for (int i=0; i<fileData.length()-3; i+=step){

result++;

}

return result;

}

public static void main(String[] args) {

StringBuffer fileData;

fileData = getFileContent();

initAlphabet(fileData, alphabet);

int total = fileData.length();

System.out.println("total: " + total);

printMap("Alphabet:", alphabet);

printAlphabetMap("Alphabet frquency:", alphabet, total);

initAlphabetForBigram(fileData, alphabetForBigramm, 2);

initAlphabetForBigram(fileData, alphabetForBigrammSingleStep, 1);

printMap("Bigram: ", alphabetForBigramm);

StringBuffer fileDataWithouSpace = new StringBuffer();

deleteSpaceFromBuffer(fileData, fileDataWithouSpace);

initAlphabet(fileDataWithouSpace, alphabetWithoutSpace);

int totalWithoutSpace = fileDataWithouSpace.length();

System.out.println("total without spaces: " + totalWithoutSpace);

int totalForBigramDoubleStep = countAmountBigram(fileData, 2);

System.out.println("total for bigram double step: " + totalForBigramDoubleStep);

int totalForBigramSingleStep = countAmountBigram(fileData, 1);

System.out.println(" total for bigram single step: " + totalForBigramSingleStep);

int totalForBigramWithoutSpacesDoubleStep = countAmountBigram(fileDataWithouSpace,2);

System.out.println("total for bigram without spaces double step: " + totalForBigramWithoutSpacesDoubleStep);

int totalForBigramWithoutSpacesSingleStep = countAmountBigram(fileDataWithouSpace, 1);

System.out.println("total for bigram without spaces single step: " + totalForBigramWithoutSpacesSingleStep);

printMap("Alphabet without space:", alphabetWithoutSpace);

printAlphabetMap("Alphabet without space frequency:", alphabetWithoutSpace, totalWithoutSpace);

initAlphabetForBigram(fileDataWithouSpace, alphabetForBigrammWithouSpace, 2);

initAlphabetForBigram(fileDataWithouSpace, alphabetForBigrammWithouSpaceSingleStep, 1);

printMap("Bigram without space:", alphabetForBigrammWithouSpace);

String[][] array = new String[33][33];

String[] alpha ={"0","а","б","в","г","д","е","ж","з","и","й","к","л","м","н","о","п","р","с","т","у","ф","х","ц","ч","ш","щ","ы","ь","э","ю","я"};

initArray(array, alpha, alphabetForBigramm);

String[][] arrayWithoutSpaces = new String[32][32];

String[] alphWithoutSpaces ={"а","б","в","г","д","е","ж","з","и","й","к","л","м","н","о","п","р","с","т","у","ф","х","ц","ч","ш","щ","ы","ь","э","ю","я"};

initArray(arrayWithoutSpaces, alphWithoutSpaces, alphabetForBigrammWithouSpace);

System.out.println();

System.out.println("Array for bigram with spaces for step = 2 : ");

showArray(array, totalForBigramDoubleStep);

System.out.println();

System.out.println("Array for bigram without spaces for step = 2: ");

showArray(arrayWithoutSpaces, totalForBigramWithoutSpacesDoubleStep);

System.out.println();

String[][] arraySingleStep = new String[33][33];

String[][] arrayWithoutSpacesSingleStep = new String[32][32];

initArray(arraySingleStep, alpha, alphabetForBigrammSingleStep);

initArray(arrayWithoutSpacesSingleStep, alphWithoutSpaces, alphabetForBigrammWithouSpaceSingleStep);

System.out.println();

System.out.println("Array for bigram with spaces for step =1 : ");

showArray(arraySingleStep, totalForBigramSingleStep);

System.out.println();

System.out.println("Array for bigram without spaces for step =1: ");

showArray(arrayWithoutSpacesSingleStep, totalForBigramWithoutSpacesSingleStep);

System.out.println();

double entropyAlphabet = calculateAndShowEntropyAlphabet(alphabet, total, "entropyAlphabet: ");

double entropyAlphabetWithoutSpace = calculateAndShowEntropyAlphabet(alphabetWithoutSpace, totalWithoutSpace, "entropyAlphabetWithoutSpace: ");

double entropyAlphabetForBigram = calculateAndShowEntropyAlphabetBigram(alphabetForBigramm, totalForBigramDoubleStep, "entropyAlphabetForBigram: ");

double entropyAlphabetForBigramWithoutSpace = calculateAndShowEntropyAlphabetBigram(alphabetForBigrammWithouSpace, totalForBigramWithoutSpacesDoubleStep, "entropyAlphabetForBigramWithoutSpace: ");

double entropyAlphabetForBigramSingleStep = calculateAndShowEntropyAlphabetBigram(alphabetForBigrammSingleStep, totalForBigramSingleStep, "entropyAlphabetForBigramSingleStep: ");

double entropyAlphabetForBigramWithoutSpaceSingleStep = calculateAndShowEntropyAlphabetBigram(alphabetForBigrammWithouSpaceSingleStep, totalForBigramWithoutSpacesSingleStep, "entropyAlphabetForBigramWithoutSpaceSingleStep: ");

System.out.println("--------------------Task\_3----------------------------");

System.out.println();

double entropyIdeal= Math.log(total)/Math.log(2);

printDouble("entropyIdeal: ", entropyIdeal);

double entropyIdealWithoutSpaces = Math.log(totalWithoutSpace)/Math.log(2);

printDouble("entropyIdealWithoutSpaces: ", entropyIdealWithoutSpaces);

double rForAlphabet = 1 - (entropyAlphabet/entropyIdeal);

printDouble("R for alphabet for text with spaces:", rForAlphabet);

double rForAlphabetWithoutSpaces = 1 - (entropyAlphabetWithoutSpace/entropyIdealWithoutSpaces);

printDouble("R for alphabet for text without spaces:", rForAlphabetWithoutSpaces);

double rForAlphabetForBigram = 1 - (entropyAlphabetForBigram/entropyIdeal);

printDouble("R for alphabet for bigram for text with spaces:", rForAlphabetForBigram);

double rForAlphabetForBigramWithoutSpaces = 1 - (entropyAlphabetForBigramWithoutSpace/entropyIdealWithoutSpaces);

printDouble("R for alphabet for bigram for text without spaces:", rForAlphabetForBigramWithoutSpaces);

double rForAlphabetForBigramSingleStep = 1 - (entropyAlphabetForBigramSingleStep/entropyIdeal);

printDouble("R for alphabet for bigram single step for text with spaces:", rForAlphabetForBigramSingleStep);

double rForAlphabetForBigramWithoutSpacesSingleStep = 1 - (entropyAlphabetForBigramWithoutSpaceSingleStep/entropyIdealWithoutSpaces);

printDouble("R for alphabet for bigram single step for text without spaces:", rForAlphabetForBigramWithoutSpacesSingleStep);

}

private static void printDouble(String desc, double statement){

System.out.println(desc + statement);

System.out.println();

}

private static double calculateAndShowEntropyAlphabet(Map<Character, Integer> alphabet, int total, String desc){

double entropyAlphabet = 0;

for (char c : alphabet.keySet()){

double probability = (double)alphabet.get(c)/total;

entropyAlphabet += probability\*(Math.log(probability)/Math.log(2));

}

entropyAlphabet = Math.abs(entropyAlphabet);

System.out.println(desc + entropyAlphabet);

System.out.println();

return entropyAlphabet;

}

private static double calculateAndShowEntropyAlphabetBigram(Map<String, Integer> alphabet, int total, String desc){

double entropyAlphabet = 0;

for (String s : alphabet.keySet()){

double probability = (double)alphabet.get(s)/total;

entropyAlphabet += probability\*(Math.log(probability)/Math.log(2));

}

entropyAlphabet = Math.abs(entropyAlphabet);

entropyAlphabet /= 2;

System.out.println(desc + entropyAlphabet);

System.out.println();

return entropyAlphabet;

}

}

Проблеми:

Головною проблемою було виведення кількості біграм в тексті у вигляді таблиці, оскільки для цього було використано масив величиною +1 від літер алфавіту для того, щоб створити шапку таблиці. Було проблематично співставити ключ значення біграми з відповідною ячейкою масиву.

Висновок:

Засвоїли поняття ентропії на символ джерела та його надлишковості, вивчили та порівняли різні моделі джерела відкритого тексту для наближеного визначення ентропії, набули практичних навичок щодо оцінки ентропії на символ джерела.