Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Фізико-технічний інститут

**ЛАБОРАТОРНА РОБОТА №1**

**з дисципліни**

**«Криптографія»**

**на тему: «Експериментальна оцінка ентропії на символ джерела відкритого тексту»**

Виконали:

студенти 3 курсу ФТІ

групи ФБ-74

Постолюк Діана та Хацько Микита

Перевірили:

Чорний О.

Савчук М. М.

Завадська Л. 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 біграм в тексті, а також підрахунку H1 та H2 за безпосереднім означенням. Підрахувати частоти букв та біграм, а також значення H1 та H2 на довільно обраному тексті російською мовою достатньої довжини (щонайменше 1Мб), де імовірності замінити відповідними частотами. Також одержати значення H1 та H2 на тому ж тексті, в якому вилучено всі пробіли.

2. За допомогою програми CoolPinkProgram оцінити значення H10 , H20 , H30 .

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

**Результати виконнання програми:**

H0 = 5.044394

R0 = 0.000000

H1 with spaces = 4.382130

R1 with spaces = 0.131287

H1 without spaces = 4.464625

R1 without spaces = 0.114933

H2 for intersected with spaces = 3.978992

R2 for intersected with spaces = 0.211205

H2 for intersected without spaces = 4.145663

R2 for intersected without spaces = 0.178164

H2 for not intersected with spaces = 3.978885

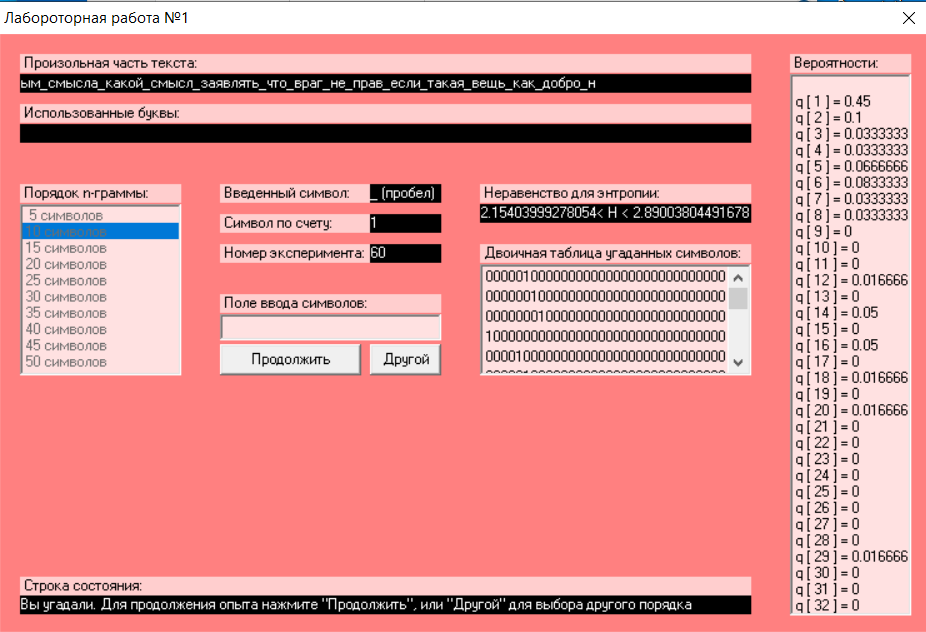
R2 for not intersected with spaces = 0.211226

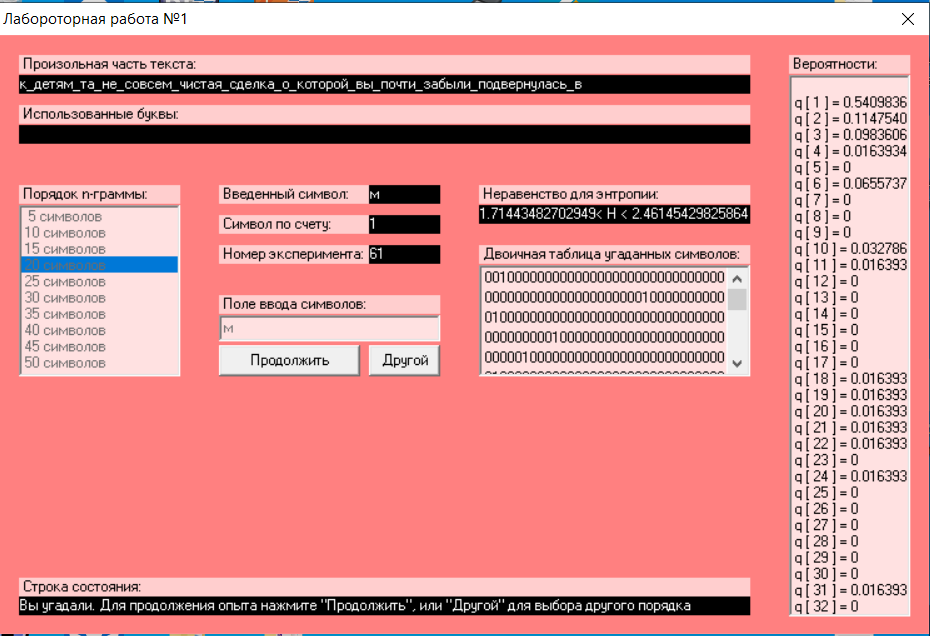
H2 for not intersected without spaces = 4.145533

R2 for not intersected without spaces = 0.178190

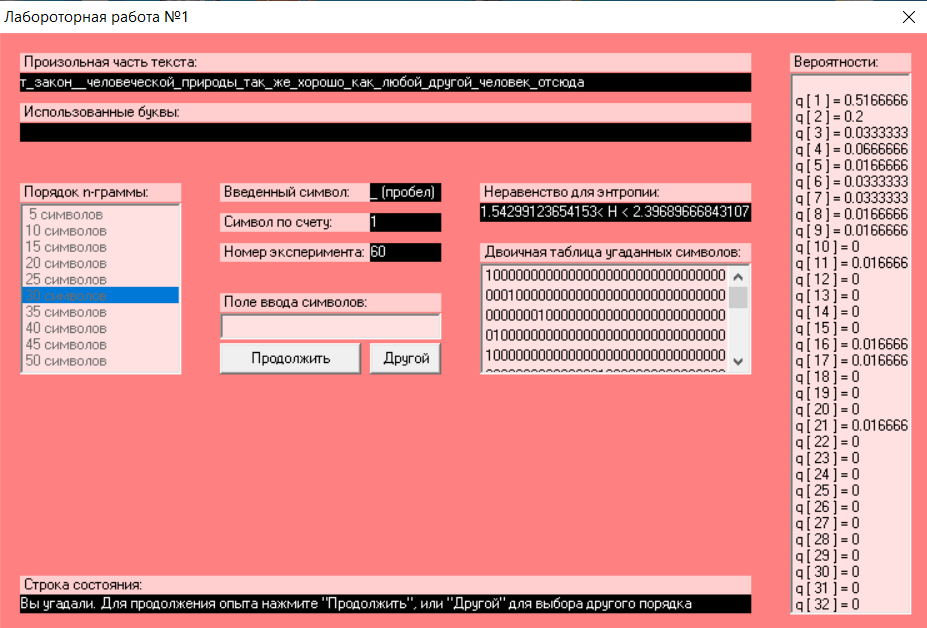
**Скріншоти результатів рожевої программи:**

**10-грамми:**

****

**20-грамми:**

**30-грамми:**

****

**Налишковість джерела:**

**10-грамми: 0.572 < R < 0.427**

**20-грамми: 0.512 < R < 0.659**

**30-грамми: 0.526 < R < 0.694**

**Код програми**

import sys, argparse, math, re

from operator import itemgetter      #  For dict reversing,

from itertools import groupby        #  without losing n-gramms

\_\_count=0

\_\_num=0

\_\_unique=0

\_\_entropy=0

rus\_dict=['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', 'ъ',

          'ы', 'ь', 'э', 'ю', 'я', ' ']

def polygramms\_freq(filename, fileEncoding, num, step=1, specCharAsSpace=True, countSpace=True):

    global \_\_num, \_\_count, \_\_unique

    res\_dict={}

    polygramm\_count=0

    polygramm\_str=''

    lastIsSpace=False

    with open(filename, encoding=fileEncoding) as file:

        for line in file:

            for i in range(len(line)):

                if ( (line[i].isalpha()==False and line[i]!=' ') or (line[i]==' ' and countSpace==False ) or (line[i] not in rus\_dict)):

                    if specCharAsSpace==False or countSpace==False:

                        continue

                    polygramm\_str+=' '

                else:

                    polygramm\_str+=line[i].lower()

                polygramm\_str=re.sub(' +', ' ', polygramm\_str)

                if (len(polygramm\_str)==1 and polygramm\_str==' '):

                    if lastIsSpace==True:

                        polygramm\_str=''

                    else:

                        lastIsSpace=True

                else:

                    lastIsSpace=False

                if len(polygramm\_str)==num:

                    try:

                        res\_dict[polygramm\_str]+=1

                    except:

                        \_\_unique+=1

                        res\_dict[polygramm\_str]=1

                    polygramm\_str=polygramm\_str[step:]

                    polygramm\_count+=1

    \_\_num=num

    \_\_count=polygramm\_count

    for key in res\_dict.keys():

        res\_dict[key]=((float)(res\_dict[key]))/((float)(polygramm\_count))

    return res\_dict

def print\_inv\_dict(\_dict):

    string="=========================================================\n" + "Polygramm ({}) Count: {}\n".format(\_\_num, \_\_count)

    print(string)

    for a in sorted(\_dict, reverse=True):

        string = str(a)+":\t["

        for elem in sorted(\_dict[a]):

            string+=" '{}' ,".format(elem)

        print(string[:-1]+"],")

def write\_inv\_dict(\_dict, file):

    with open(file, 'a') as out:

        string="=========================================================\n" + "Polygramm ({}) Count: {}\n".format(\_\_num, \_\_count)

        out.write(string)

        for a in sorted(\_dict, reverse=True):

            string = str(a)+":\t["

            for elem in sorted(\_dict[a]):

                string+=" '{}' ,".format(elem)

            out.write(string[:-1]+"],\n")

def find\_entropy(freq\_dict):

    res=0

    for elem in freq\_dict:

        res+=(elem\*math.log(elem,2))\*len(freq\_dict[elem])

    res\*=(-1/\_\_num)

    print(\_\_num)

    return res

def find\_redundancy(alphabet\_len):

    return 1-(\_\_entropy/(math.log(alphabet\_len,2)))

def main(cmdFilename="TEXT.txt", cmdFileEncoding="cp866", cmdNum=1, cmdNumStep=1,

         cmdSpecCharAsSpace=True, cmdCountSpace=True, cmdOutput="Out"):

    global \_\_entropy

    print(cmdSpecCharAsSpace, cmdCountSpace)

    snd = itemgetter(1)

    res=polygramms\_freq(filename=cmdFilename, fileEncoding=cmdFileEncoding,

                        num=cmdNum, step=cmdNumStep, specCharAsSpace=cmdSpecCharAsSpace, countSpace=cmdCountSpace)

    inv\_map = {number: [char for char,\_ in v]

                 for number, v in groupby(sorted(res.items(), key=snd), snd)}

    print\_inv\_dict(inv\_map)

    write\_inv\_dict(inv\_map, cmdOutput)

    \_\_entropy=find\_entropy(inv\_map)

    #\_\_entropy=find\_entropy(res)

    redundancy= find\_redundancy(len(res) if cmdNum==1 else len(polygramms\_freq(filename=cmdFilename, fileEncoding=cmdFileEncoding,

                        num=1, step=1, specCharAsSpace=cmdSpecCharAsSpace, countSpace=cmdCountSpace)))

    print("H\_{}: {}".format(cmdNum, \_\_entropy))

    print("R by H\_{}: {}".format(cmdNum, redundancy))

    data={}

    data["inputFile"]=cmdFilename

    data["n\_gram"]=cmdNum

    data["n\_gramStep"]=cmdNumStep

    data["specCharAsSpace"]=cmdSpecCharAsSpace

    data["countSpace"]=cmdCountSpace

    data["entropy"]=\_\_entropy

    data["redundancy"]=redundancy

    data["result"]=inv\_map

if \_\_name\_\_ == "\_\_main\_\_":

    #try:

    parser = argparse.ArgumentParser()

    parser.add\_argument("input", help="name of input file", type=str)

    parser.add\_argument("inputEncoding", help="encoding of input file", type=str)

    parser.add\_argument("n\_gram", help="number for n-gram", type=int)

    parser.add\_argument("n\_gramStep", help="number of step for n-gram", type=int)

    parser.add\_argument("countSpace", help="True for counting spaces, False - to not count them in n-grams", type=str)

    parser.add\_argument("specCharAsSpace", help="True for counting non-alpha chars as space, False - to not count them at all", type=str)

    parser.add\_argument("output", help="name of output file", type=str)

    args = parser.parse\_args()

    main(cmdFilename=args.input, cmdFileEncoding=args.inputEncoding,

            cmdNum=args.n\_gram, cmdNumStep=args.n\_gramStep,  cmdCountSpace= False if args.countSpace.lower()!="true" else True,

            cmdSpecCharAsSpace=False if args.specCharAsSpace.lower()!="true" else True, cmdOutput=args.output)

**Висновки:**

Під час данного комп’ютерного практикуму, ми навчились визначати ентропію на символ джерела та його надлишковості. Порівняли різні моделі джерел відркритого тексту для наближенного визначення ентропії та набули практичних навичок оцінки ентропії на символ джерела.