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 H_1 та H_2 за безпосереднім означенням. Підрахувати частоти букв та біграм, а також значення H_1 та H_2 на довільно обраному тексті російською мовою достатньої довжини (щонайменше 1Мб), де імовірності замінити відповідними частотами. Також одержати значення H_1 та H_2 на тому ж тексті, в якому вилучено всі пробіли.

2. За допомогою програми CoolPinkProgram оцінити значення H^{10} , H^{20} , H^{30} .

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

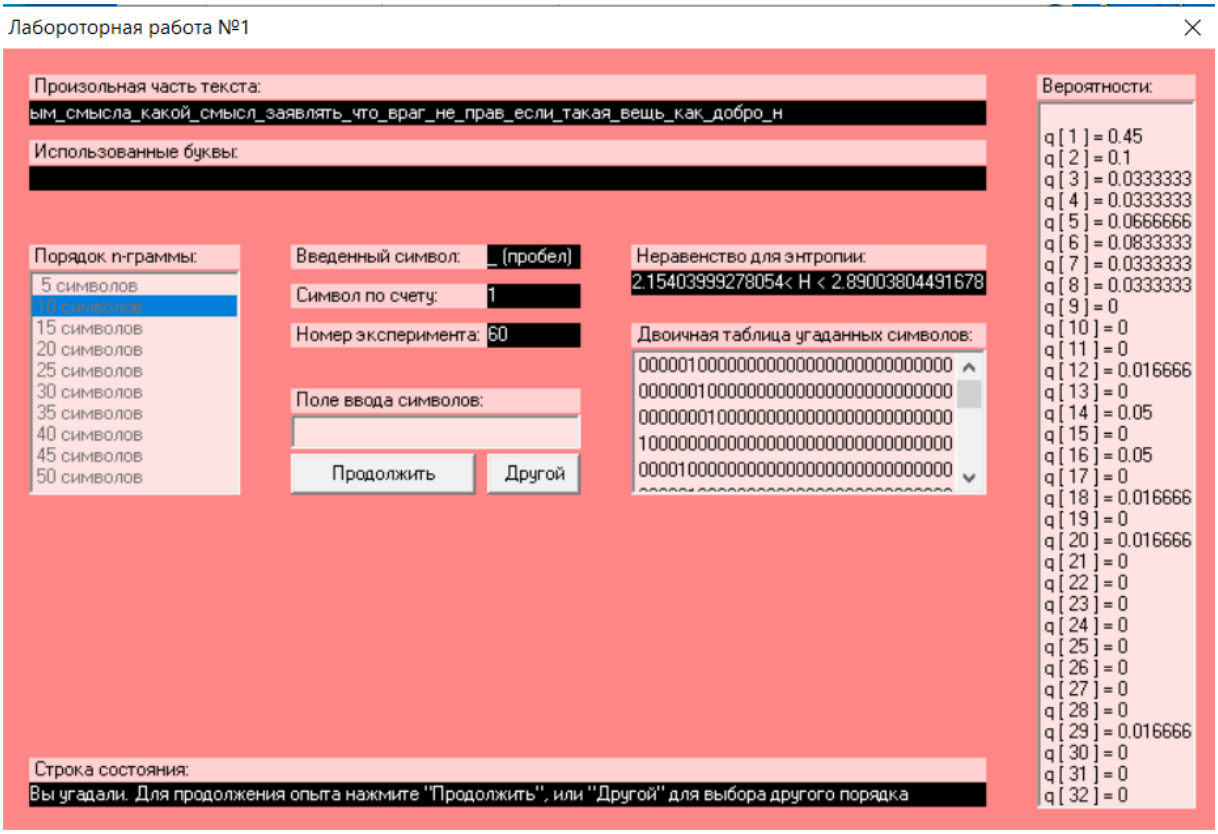
R2 for not intersected with spaces = 0.211226

H2 for not intersected without spaces = 4.145533

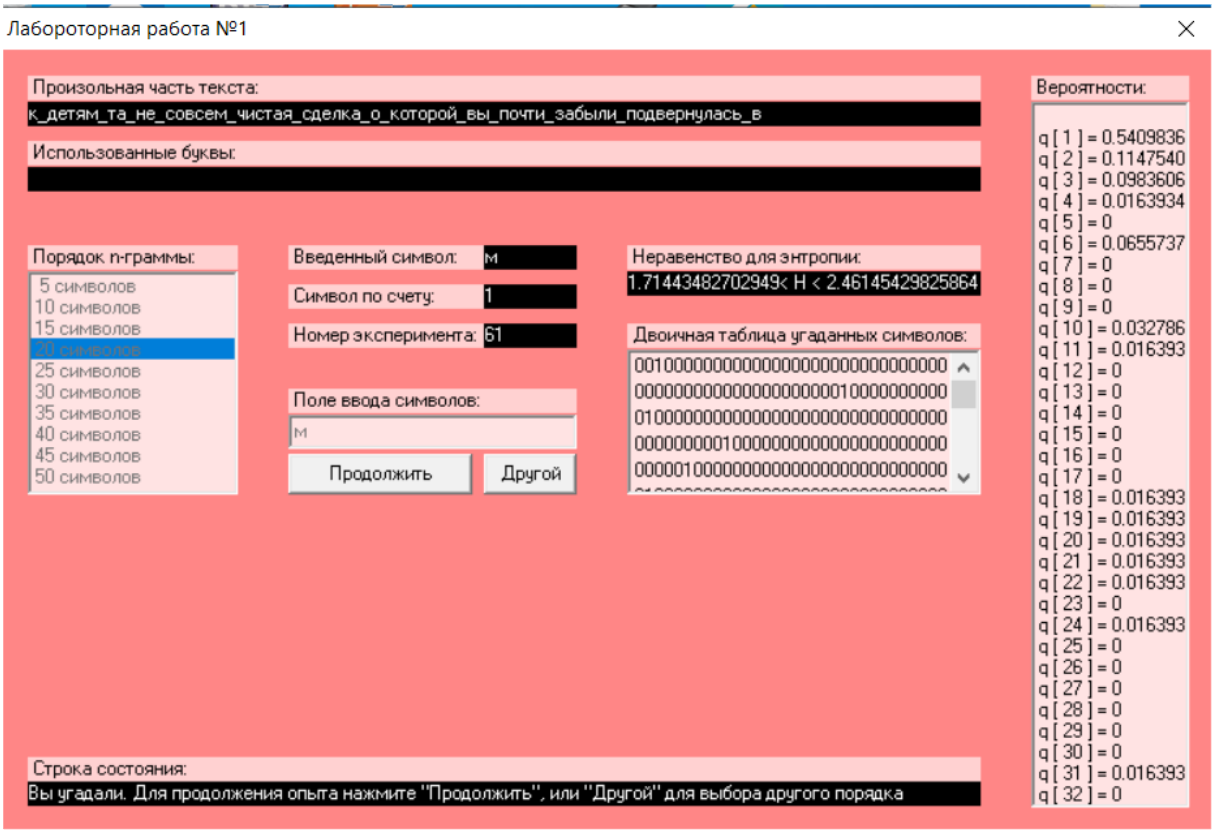
R2 for not intersected without spaces = 0.178190

Скріншоти результатів рожевої программи:

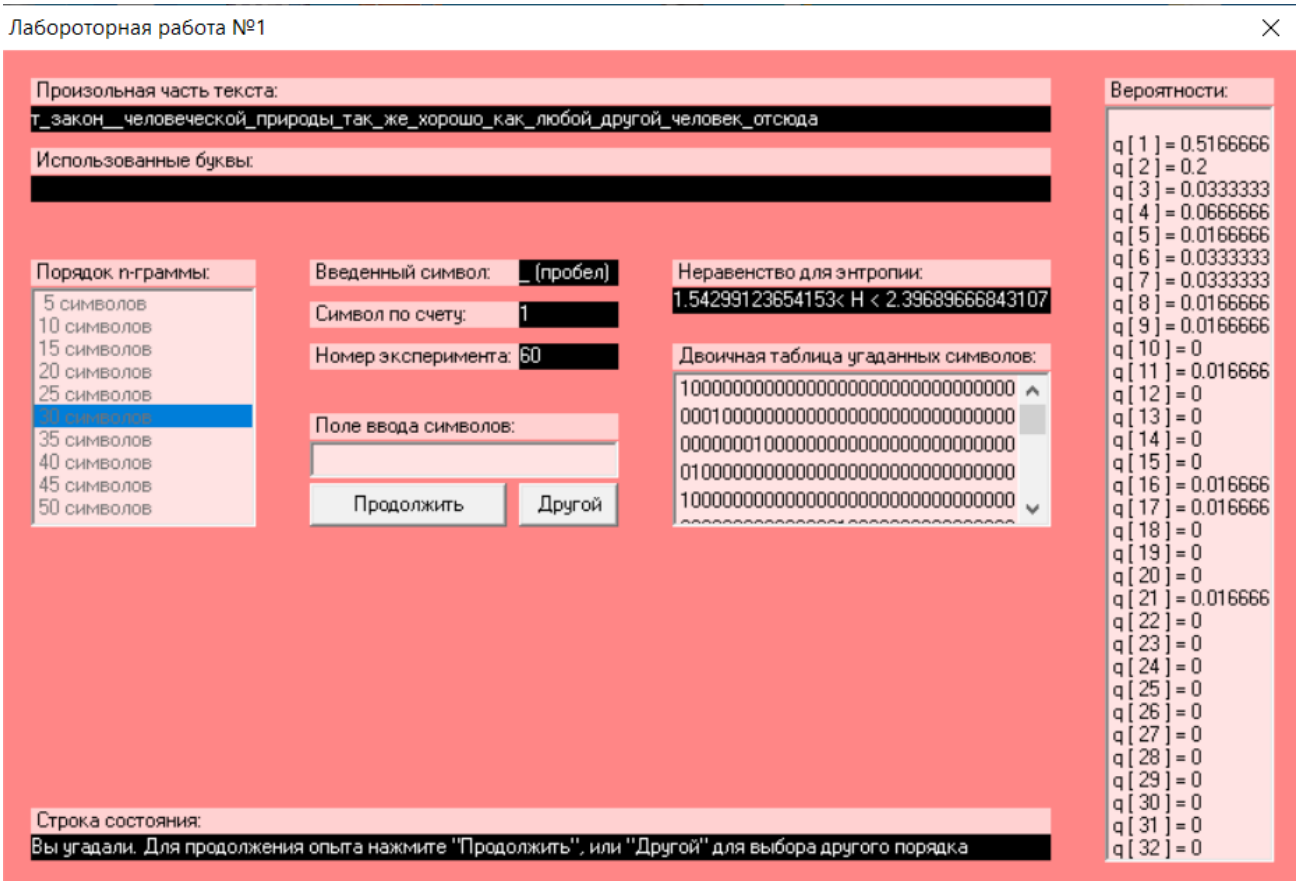
10-грамми:



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 ru
s_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 ({}) Cou
nt: {}\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=cm
dCountSpace)

    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 no
t 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.l
ower()!="true" else True,
         cmdSpecCharAsSpace=False if args.specCharAsSpace.lower()!="true" else True, cmdOutpu
t=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.