

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий фізико-технічний інститут
Кафедра інформаційної безпеки

Дисципліна «Криптографія»

Комп'ютерний практикум

Робота №1

Експериментальна оцінка ентропії на символ джерела відкритого тексту

Виконали: студенти гр. ФБ-12 Головка М. С. і Марчук І. С.

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 $(10) H$, $(20) H$, $(30) H$.
3. Використовуючи отримані значення ентропії, оцінити надлишковість російської мови в різних моделях джерела.

Хід роботи:

1. Спочатку ми зайнялися створенням текстового файлу російською мовою, який важить щонайменше 1 МБ (взяли «Мертві душі» Гоголя, тому що чому б і ні). Для того, щоб працювати з цим файлом і витягувати з нього потрібну інформацію, треба спочатку його відфільтрувати за наступними вимогами: *всі символи, окрім текстових, повинні вилучатись або замінюватись на пробіли; прописні літери – замінюватись на відповідні стрічні; послідовність пробілів (або інших розділових знаків, наприклад, символів кінця рядку) повинна трактуватись як один пробіл або вилучатись, якщо пробіл не входить до алфавіту.*

Для цього написали на Python (з яким будемо працювати і далі) такий скрипт:

```
def clean_text(text):
    text = ' '.join(text.split())

    text = ''.join(char for char in text if char.isalpha() or char.isspace() or
(char >= 'а' and char <= 'я' or char >= 'А' and char <= 'Я'))

    text = text.lower()

    text=text.replace(" ", " ")
    text=text.replace(" ", " ")
    text=text.replace(" ", " ")

    return text

input_file_path = 'D:\\uni year 3\\crypto labs\\tasks\\cp1\\test.txt'
with open(input_file_path, 'r', encoding='utf-8') as file:
    text = file.read()

output_with_spaces = clean_text(text)

output_with_spaces_path = 'D:\\uni year 3\\crypto labs\\tasks\\cp1\\output with
spaces.txt'

with open(output_with_spaces_path, 'w', encoding='utf-8') as file:
    file.write(output_with_spaces)
```

```
output_without_spaces = output_with_spaces.replace(' ', '')

output_without_spaces_path = 'D:\\\\uni year 3\\\\crypto labs\\\\tasks\\\\cp1\\\\output
without spaces.txt'

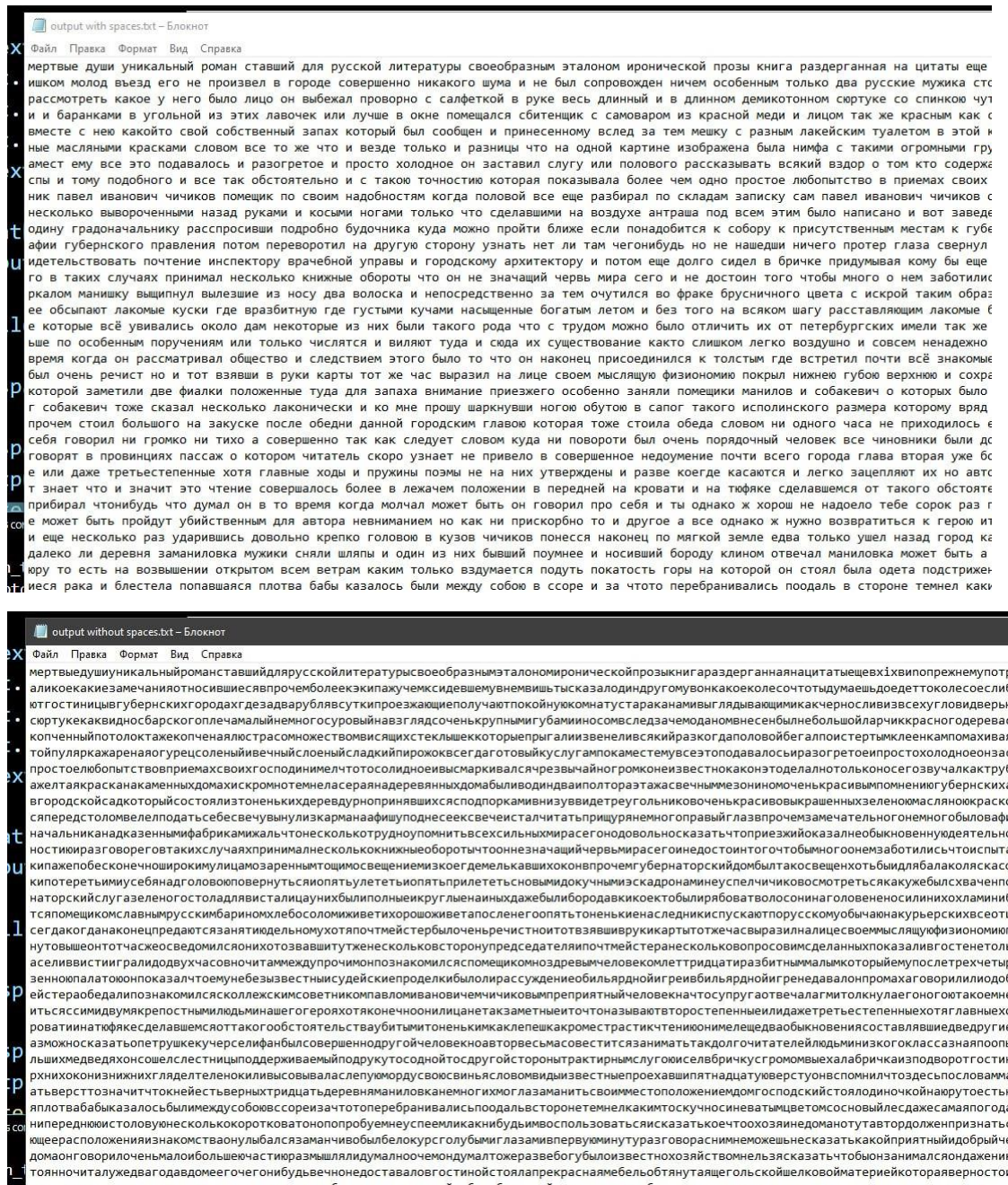
with open(output_without_spaces_path, 'w', encoding='utf-8') as file:

    file.write(output_without_spaces)

print("Formatted text with spaces has been written to 'output with spaces.txt'")

print("Formatted text without spaces has been written to 'output without
spaces.txt'")
```

Після цього в нас вийшло два файли з текстом: один з пробілами, інший без пробілів:



Тепер треба підрахувати частоти букв, які зустрічаються в тексті. Для цього спочатку підрахуємо, скільки разів кожна буква зустрічається у тексті, а потім поділимо це число на загальну кількість букв у тексті. Ми спробували зробити це через словник у Python і під час виконання коду виявилось, що в тексті все ж таки залишилося декілька латинських літер (напевно в тексті були якісь англійські/французькі слова), тому треба було це якось виправити:

```
a: 8
b: 3
c: 6
d: 1
e: 23
f: 4
g: 2
h: 4
i: 12
l: 10
m: 9
```

Тому вирішили скористатися функцією `ord()`, яка отримує ID символу, і ми зробили так, щоб в текст, який ми форматували, записувалися тільки букви російського алфавіту, в яких ID в проміжку:

```
Type "help", "copyright":
>>> ord("a")
1072
>>> ord("я")
1103
>>>
```

Тобто в минулому коді, який вже було показано, рядки

```
text = ''.join(char for char in text if char.isalpha() or char.isspace() or
(char >= 'a' and char <= 'я' or char >= 'A' and char <= 'Я'))
```

```
text = text.lower()
```

замінили на

```
text = text.lower()
```

```
text = ''.join(char for char in text if (char.isalpha() and ord(char)>1071 and
ord(char)<1106) or char.isspace())
```

За допомогою наступного коду порахували, скільки разів зустрічається кожна буква в тексті:

```
for litera in output_without_spaces:
    if litera in letter_frequency:
        letter_frequency[litera] += 1
    else:
```

```

letter_frequency.update({litera: 1})

for letter, frequency in sorted(letter_frequency.items(), key=lambda x:x[1],
reverse=True):

    print(f"{letter}: {frequency}")

```

Текст без пробілів:

```

PS D:\uni_year_2\aks_lab>
о: 67964
е: 51416
а: 46290
и: 41319
н: 37320
т: 37010
с: 30826
в: 28236
л: 27009
р: 24990
к: 24904
д: 18840
у: 18242
м: 17904
п: 16799
ь: 12589
б: 11565
я: 11550
ч: 11480
ы: 11428
г: 10624
з: 10192
ж: 6706
й: 6320
х: 5908
ш: 5782
ю: 3633
ц: 1932
щ: 1798
э: 1337
ф: 738
ъ: 197
ё: 85
Formatted text with spaces

```

І потім рахуємо частоту кожної букви, поділивши кількість зустрічань однієї букви на загальну кількість букв у тексті (для того, щоб гарно було, вивели у відсотках):

```

print("загальна "," відсоток від ")

print("кількість"," загального")

for letter, frequency in sorted(letter_frequency.items(), key=lambda x:x[1],
reverse=True):

```



```
print(f"{letter}: {frequency},      %=  
{round(int(frequency)/len(output_with_spaces)*100,3)}")
```

```
о: 67964,%= 11.272  
е: 51416,%= 8.528  
а: 46290,%= 7.677  
и: 41319,%= 6.853  
н: 37320,%= 6.19  
т: 37010,%= 6.138  
с: 30826,%= 5.113  
в: 28236,%= 4.683  
л: 27009,%= 4.48  
р: 24990,%= 4.145  
к: 24904,%= 4.13  
д: 18840,%= 3.125  
у: 18242,%= 3.026  
м: 17904,%= 2.969  
п: 16799,%= 2.786  
ь: 12589,%= 2.088  
б: 11565,%= 1.918  
я: 11550,%= 1.916  
ч: 11480,%= 1.904  
ы: 11428,%= 1.895  
г: 10624,%= 1.762  
з: 10192,%= 1.69  
ж: 6706,%= 1.112  
й: 6320,%= 1.048  
х: 5908,%= 0.98  
ш: 5782,%= 0.959  
ю: 3633,%= 0.603  
ц: 1932,%= 0.32  
щ: 1798,%= 0.298  
э: 1337,%= 0.222  
ф: 738,%= 0.122  
ъ: 197,%= 0.033  
ё: 85,%= 0.014
```

І ось з пробілом:

загальна кількість	відсоток від загального
: 115910,	%= 16.125
о: 67964,	%= 9.455
е: 51416,	%= 7.153
а: 46290,	%= 6.44
и: 41319,	%= 5.748
н: 37320,	%= 5.192
т: 37010,	%= 5.149
с: 30826,	%= 4.288
в: 28236,	%= 3.928
л: 27009,	%= 3.757
р: 24990,	%= 3.476
к: 24904,	%= 3.464
д: 18840,	%= 2.621
у: 18242,	%= 2.538
м: 17904,	%= 2.491
п: 16799,	%= 2.337
ь: 12589,	%= 1.751
б: 11565,	%= 1.609
я: 11550,	%= 1.607
ч: 11480,	%= 1.597
ы: 11428,	%= 1.59
г: 10624,	%= 1.478
з: 10192,	%= 1.418
ж: 6706,	%= 0.933
й: 6320,	%= 0.879
х: 5908,	%= 0.822
ш: 5782,	%= 0.804
ю: 3633,	%= 0.505
ц: 1932,	%= 0.269
щ: 1798,	%= 0.25
э: 1337,	%= 0.186
ф: 738,	%= 0.103
ъ: 197,	%= 0.027
ё: 85,	%= 0.012

Тепер обрахуємо значення ентропії H_1 монограм тексту. Скористаємось формулою з методичних вказівок:

$$H(x_1, x_2, \dots, x_n) = - \sum_{z_1, z_2, \dots, z_n} P(x_1 = z_1, \dots, x_n = z_n) \cdot \log_2 P(x_1 = z_1, \dots, x_n = z_n).$$

Тобто для того, щоб знайти ентропію H_1 , треба знайти суму ентропій кожної букви. Код для цього (скористалися бібліотекою `math` для обчислення логарифму):

```
import math
```

```

entropy=float(0)

for chastota in letter_frequency.values():

    entropy+=(chastota/len(WITH_SPACES_OR_NOT))*math.log2(chastota/len(WITH_SPACES_OR_NOT))

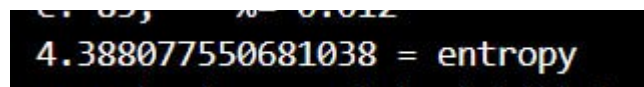
print(-entropy, "= entropy")

```

(масив WITH_SPACES_OR_NOT може бути як масивом букв з пробілом, так і без, дивлячись який файл ми відкриваємо)

Значення ентропій, які в нас вийшли:

- з пробілами:

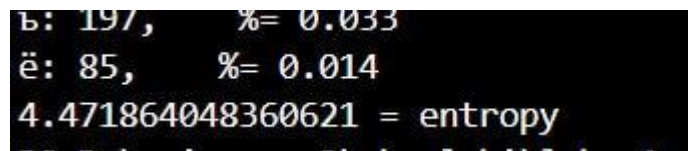


```

4.388077550681038 = entropy

```

- без пробілів:



```

4.471864048360621 = entropy

```

Значення ентропії літер з пробілами вийшло менше, ніж значення ентропії без пробілів, що звучить логічно, адже текст з пробілами (слова відокремлені) дійсно легше передбачити аніж текст без пробілів (слова написані суцільним текстом).

На цьому етапі ми також вирішили зробити обчислення ентропії і частот функціями в Python, адже нам потрібно буде ще далі рахувати частоти і ентропії, і взагалі так код виглядає набагато приємніше (показую весь код):

```

def entropy(original_text, letter_freq):

    entropy=float(0)

    for chastota in letter_freq.values():

        entropy+=(chastota/len(original_text))*math.log2(chastota/len(original_text))

    return -entropy

def find_freq(formated_text):

    letter_frequency = {}

    for litera in formated_text:

        if litera in letter_frequency:

            letter_frequency[litera] += 1

        else:

            letter_frequency.update({litera: 1})

```



```

        return letter_frequency

def print_freq(let_freq, formatted_text):
    print("загальна ", " відсоток від ")
    print("кількість", " загального")

    for letter, frequency in sorted(let_freq.items(), key=lambda x:x[1],
reverse=True):
        print(f"{letter}: {frequency},      %="
{round(int(frequency)/len(formatted_text)*100,3)}")

    return None

output_with_spaces_freq=find_freq(output_with_spaces)
output_without_spaces_freq=find_freq(output_without_spaces)
print(entropy(output_with_spaces,output_with_spaces_freq))
print(entropy(output_without_spaces,output_without_spaces_freq))
print(print_freq(output_without_spaces_freq,output_without_spaces))

```

Виводить він те ж саме, що й раніше, але працювати з ним зручніше.

Тепер знайдемо частоти біграм. Спочатку вирішили знайти частоти для пар букв, які перетинаються (тобто з кроком 1). Ось код для цього:

```

def find_freq_2(formatted_text):
    letter_frequency = {}

    for i in range(len(formatted_text) - 1):
        pair = formatted_text[i:i+2]
        if len(pair) == 2:
            if pair in letter_frequency:
                letter_frequency[pair] += 1
            else:
                letter_frequency[pair] = 1

    return letter_frequency

```

Частоти без пробілів:

загальна кількість	відсоток від загального
то: 9712,	%= 1.611
ов: 7873,	%= 1.306
ко: 7787,	%= 1.292
ст: 7659,	%= 1.27
но: 7049,	%= 1.169
по: 6932,	%= 1.15
не: 6798,	%= 1.127
на: 6364,	%= 1.056
ка: 6332,	%= 1.05
го: 5859,	%= 0.972
ен: 5822,	%= 0.966
ос: 5694,	%= 0.944
ак: 5672,	%= 0.941
ни: 5409,	%= 0.897
от: 5371,	%= 0.891
ал: 5134,	%= 0.852
ро: 5118,	%= 0.849
он: 5078,	%= 0.842
пр: 5014,	%= 0.832
во: 4938,	%= 0.819
ло: 4919,	%= 0.816
ли: 4755,	%= 0.789
ра: 4646,	%= 0.771
ел: 4617,	%= 0.766
ес: 4592,	%= 0.762
ор: 4577,	%= 0.759
ть: 4490,	%= 0.745
ет: 4360,	%= 0.723
од: 4314,	%= 0.716
ер: 4186,	%= 0.694
та: 4136,	%= 0.686
ом: 4082,	%= 0.677
ре: 4074,	%= 0.676
ол: 4043,	%= 0.671
те: 4022,	%= 0.667
ва: 3945,	%= 0.654

Частоти з пробілами:

загальна кількість	відсоток від загального
о : 15355,	%= 2.136
и : 13297,	%= 1.85
е : 13285,	%= 1.848
п: 12101,	%= 1.683
н: 11414,	%= 1.588
в: 11058,	%= 1.538
с: 10702,	%= 1.489
а : 10249,	%= 1.426
то: 9490,	%= 1.32
ко: 7610,	%= 1.059
ь : 7608,	%= 1.058
и: 7589,	%= 1.056
ст: 7489,	%= 1.042
к: 6958,	%= 0.968
по: 6928,	%= 0.964
но: 6895,	%= 0.959
не: 6761,	%= 0.941
я : 6707,	%= 0.933
о: 6557,	%= 0.912
на: 6329,	%= 0.88
ка: 6288,	%= 0.875
ов: 6187,	%= 0.861
д: 6048,	%= 0.841
т: 5924,	%= 0.824
го: 5831,	%= 0.811
м : 5522,	%= 0.768
б: 5362,	%= 0.746
ни: 5233,	%= 0.728
ч: 5099,	%= 0.709
ро: 5080,	%= 0.707
пр: 5014,	%= 0.698
ал: 4982,	%= 0.693
ак: 4969,	%= 0.691
в : 4890,	%= 0.68
й : 4865,	%= 0.677

І потім знайшли значення ентропій H_2 для цих біграм так само, як знаходили значення ентропій для букв (тобто скористалися вже існуючою функцією, яку ми раніше написали ‘entropy’):

```
print("entropy without spaces
bigram",entropy(output_without_spaces,output_without_spaces_bi_freq))
```

```
print("entropy with spaces  
bigram",entropy(output_with_spaces,output_with_spaces_bi_freq))
```

Вийшли такі ентропії:

```
пб: 1,    %= 0.0  
entropy without spaces bigram 8.271067881026687  
entropy with spaces bigram 7.932268359359433
```

Знову ентропія тексту без пробілів більша, ніж з пробілами.

І тепер роздивимось випадок, коли пари літер не перетинаються (тобто йдемо по тексту з кроком 2):

```
def find_freq_3(formatted_text):  
    letter_frequency = {}  
  
    # Iterate through the text, taking every pair of non-overlapping letters  
    for i in range(0, len(formatted_text) - 1, 2):  
        pair = formatted_text[i:i+2]  
        if len(pair) == 2:  
            if pair in letter_frequency:  
                letter_frequency[pair] += 1  
            else:  
                letter_frequency[pair] = 1  
  
    return letter_frequency
```

Кількості зустрічань біграм зменшилися майже вдвічі:

- без пробілів:

загальна кількість	відсоток від загального
то: 4800,	%= 0.796
ст: 3890,	%= 0.645
ко: 3875,	%= 0.643
ов: 3873,	%= 0.642
по: 3534,	%= 0.586
но: 3456,	%= 0.573
не: 3306,	%= 0.548
на: 3212,	%= 0.533
ка: 3181,	%= 0.528
го: 2952,	%= 0.49
ен: 2928,	%= 0.486
ос: 2832,	%= 0.47
ни: 2789,	%= 0.463
ак: 2773,	%= 0.46
от: 2669,	%= 0.443
он: 2604,	%= 0.432
ро: 2572,	%= 0.427
ал: 2559,	%= 0.424
во: 2541,	%= 0.421
пр: 2515,	%= 0.417
ло: 2483,	%= 0.412
ли: 2379,	%= 0.395
ра: 2334,	%= 0.387
ес: 2330,	%= 0.386
ор: 2303,	%= 0.382
ел: 2285,	%= 0.379
ть: 2246,	%= 0.373
ет: 2223,	%= 0.369
од: 2127,	%= 0.353
ер: 2082,	%= 0.345
та: 2052,	%= 0.34
ом: 2033,	%= 0.337
те: 2021,	%= 0.335
ре: 2020,	%= 0.335
ва: 2018,	%= 0.335

- з пробілами:

загальна кількість	відсоток від загального
о : 7601,	%= 1.057
е : 6762,	%= 0.941
и : 6586,	%= 0.916
п: 5988,	%= 0.833
н: 5710,	%= 0.794
в: 5482,	%= 0.763
с: 5378,	%= 0.748
а : 5202,	%= 0.724
то: 4772,	%= 0.664
ко: 3835,	%= 0.533
ь : 3828,	%= 0.533
и: 3821,	%= 0.532
ст: 3768,	%= 0.524
по: 3489,	%= 0.485
не: 3415,	%= 0.475
но: 3412,	%= 0.475
к: 3407,	%= 0.474
я : 3394,	%= 0.472
о: 3291,	%= 0.458
ка: 3168,	%= 0.441
на: 3124,	%= 0.435
ов: 3090,	%= 0.43
д: 3045,	%= 0.424
т: 2937,	%= 0.409
го: 2898,	%= 0.403
м : 2785,	%= 0.387
б: 2669,	%= 0.371
ни: 2639,	%= 0.367
ро: 2572,	%= 0.358
ч: 2537,	%= 0.353
пр: 2533,	%= 0.352
ал: 2507,	%= 0.349

І знову ж знаходимо ентропії H_2 тепер вже для біграм без перетину:

```
пб: 1,    %= 0.0
entropy without spaces bigram without overlap 4.635602402867155
entropy with spaces bigram without overlap 4.4658002818063105
```

Знову значення ентропії тексту без пробілів більша, ніж тексту з пробілами.

Щоб було зручніше, запишемо всі значення ентропій, які ми знайшли, у таблицку:

Произвольная часть текста:
ый_человек_знает_ег

Использованные буквы:

Порядок n-граммы:
5 символов
10 символов
15 символов
20 символов
25 символов
30 символов
35 символов
40 символов
45 символов
50 символов

Введенный символ:

Символ по счету:

Номер эксперимента: 51

Неравенство для энтропии:
 $1.51567813452112 < H < 2.29225447733486$

Двоичная таблица угаданных символов:

00000010000000000000000000000000	▲
10000000000000000000000000000000	■
00100000000000000000000000000000	
10000000000000000000000000000000	
00001000000000000000000000000000	▼

Вероятности:

q[1]	= 0.62
q[2]	= 0.08
q[3]	= 0.02
q[4]	= 0.02
q[5]	= 0.02
q[6]	= 0.04
q[7]	= 0.04
q[8]	= 0
q[9]	= 0.04
q[10]	= 0.02
q[11]	= 0.02
q[12]	= 0
q[13]	= 0
q[14]	= 0
q[15]	= 0.02
q[16]	= 0
q[17]	= 0
q[18]	= 0
q[19]	= 0
q[20]	= 0
q[21]	= 0
q[22]	= 0
q[23]	= 0.02
q[24]	= 0.02
q[25]	= 0
q[26]	= 0
q[27]	= 0
q[28]	= 0
q[29]	= 0.02
q[30]	= 0
q[31]	= 0
q[32]	= 0

Строка состояния:

Продолжить Другой

Нерівність ентропії:

Неравенство для энтропии:
 $1.51567813452112 < H < 2.29225447733486$

$$- H^{(30)}$$

Произвольная часть текста:
пояснений_но_в_данный_момент_

Использованные буквы:

Порядок n-граммы:
5 символов
10 символов
15 символов
20 символов
25 символов
30 символов
35 символов
40 символов
45 символов
50 символов

Введенный символ:

Символ по счету:

Номер эксперимента: 51

Неравенство для энтропии:
 $1.13902801008919 < H < 1.70396957073853$

Двоичная таблица угаданных символов:

01000000000000000000000000000000	▲
10000000000000000000000000000000	■
10000000000000000000000000000000	
00000000000000100000000000000000	
10000000000000000000000000000000	▼

Вероятности:

q[1]	= 0.66
q[2]	= 0.18
q[3]	= 0
q[4]	= 0
q[5]	= 0
q[6]	= 0
q[7]	= 0
q[8]	= 0.04
q[9]	= 0.02
q[10]	= 0.02
q[11]	= 0.02
q[12]	= 0.02
q[13]	= 0
q[14]	= 0
q[15]	= 0.02
q[16]	= 0
q[17]	= 0
q[18]	= 0
q[19]	= 0
q[20]	= 0
q[21]	= 0
q[22]	= 0
q[23]	= 0.02
q[24]	= 0
q[25]	= 0
q[26]	= 0
q[27]	= 0
q[28]	= 0
q[29]	= 0
q[30]	= 0
q[31]	= 0
q[32]	= 0

Строка состояния:

Продолжить Другой

Нерівність для ентропії:

$$\text{Неравенство для энтропии:} \\ 1.13902801008919 < H < 1.70396957073853$$

Також занесемо значення в таблицку:

H_n	Нерівність для ентропії
$H^{(10)}$	$2,58759931788218 < H < 3,17669456263237$
$H^{(20)}$	$1,51567813452112 < H < 2,29225447733486$
$H^{(30)}$	$1,13902801008919 < H < 1,70396957073853$

Таблиця 2. Результати роботи з CoolPinkProgram

Можна побачити, що зі збільшенням кількості символів ентропія зменшується, адже чим більше кількість відомих символів (інформації), тим легше відгадати наступний символ. (складніше за все було відгадувати букву, яка йшла після пробілу, тому що не завжди легко відгадати наступне слово)

3. Нарешті за знайденими значеннями ентропії обрахуємо надлишковість російської мови в різних моделях джерела. Для цього використаємо формулу з методичних вказівок:

$$R = 1 - \frac{H_{\infty}}{H_0},$$

де H_{∞} - значення ентропії, які ми порахували для різних випадків і n-грам, а

$H_0 = \log_2 m$, де m – це кількість букв в алфавіті (або кількість пар букв, якщо це біграми).

Код для обрахування значень надлишковості:

```
def nadlishkovist_without_spaces(entropy):
    return 1-(entropy/math.log2(33))

def nadlishkovist_with_spaces(entropy):
    return 1-(entropy/math.log2(34))

def nadlishkovist_without_spaces_bigram(entropy):
    return 1-(entropy/math.log2(1089))

def nadlishkovist_with_spaces_bigram(entropy):
    return 1-(entropy/math.log2(1156))
```

В нас вийшли такі значення надлишковості:

```

entropy with spaces 4.388077550681038
R= 0.1374723142739288
*****
entropy without spaces 4.471864048360621
R= 0.11349828293564157
*****
entropy without spaces bigram 8.271067881026687
R= 0.18017231749542584
*****
entropy with spaces bigram 7.932268359359433
R= 0.2204101919877679
*****
entropy without spaces bigram without overlap 4.635602402867155
R= 0.5405194069712468
*****
entropy with spaces bigram without overlap 4.4658002818063105
R= 0.5610975036911762

```

Також занесемо дані в таблицю:

H_n	Текст з пробілами	Текст без пробілів
H_1	0,1374723142739288	0,11349828293564157
H_2 (з перетинами)	0,2204101919877679	0,18017231749542584
H_2 (без перетинів)	0,5610975036911762	0,5405194069712468

Таблиця 3. Значення надлишковості R для монограм і біграм

Висновки: у ході комп'ютерного практику було обраховано значення ентропії для монограм і біграм певного тексту, написаного російською мовою, теоретичним шляхом. Було також обраховано значення ентропії $H^{(10)}$, $H^{(20)}$, $H^{(30)}$ експериментальним шляхом. За даними таблиці 1 можна побачити, що значення ентропій в тексті з пробілами менше, ніж в тексті без пробілів, що вказує на те, що текст з пробілами передбачити легше, ніж без пробілів. За даними таблиці 3 виявилося, що зі збільшенням кількості символів зменшується ентропія, тому що маючи більше інформації, простіше передбачити інформацію, яка буде йти далі. Наприкінці лабораторної роботи також було обчислено і занесено у таблицю 3 значення надлишковості російської мови R для монограм і біграм, що дозволяє нам визначити, наскільки можна ущільнити текст деякою схемою кодування без втрати його змісту. Таким чином ми навчилися визначати складність передбачування тексту за певною кількістю букв.

UPD: Виправили обчислення ентропії і надлишковості (треба було брати кількість монограм і біграм тільки конкретного джерела, а не в принципі всіх можливих біграм і монограм). Для цього створили три функції для ентропії (entropy, entropy_bigram_with_overlap, entropy_bigram_without_overlap) і надлишковості (nadlishkovist_monogram, nadlishkovist_bigram_with_overlap, nadlishkovist_bigram_without_overlap).

В результаті вийшло так:

```
PS D:\uni_year_2\aks_laba1\laba_1> & C:/Users/IgorM/AppData/Local/Microsoft/WindowsA
entropy with spaces 4.388077550681038
R= 0.774453558950362
*****

entropy without spaces 4.471864048360621
R= 0.7671102845920259
*****

entropy without spaces bigram 8.271079206307245
R= 0.5692513230097132
*****

entropy with spaces bigram 7.93227738717684
R= 0.5922822470678346
*****

entropy without spaces bigram without overlap 8.271206464293378
R= 0.5455790084465703
*****

entropy with spaces bigram without overlap 7.931601954737005
R= 0.5702267333032143
```