Міністерство освіти і науки України

Національний технічний університет України

"Київський політехнічний інститут імені Ігоря Сікорського"

Фізико-технічний інститут

Криптографія

Комп’ютерний практикум №1

Експериментальна оцінка на символ джерела відкритого тексту

Виконали:

Студенти 3 курсу

Загородній Я.М, Венгер П.Ю.

Перевірив:

Київ – 2024

Мета роботи: засвоєння поняття 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

Постановка задачі:

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

Хід роботи

Весь код програми поданий в lab1_cryptography.ipynb, також будуть прикріплені файли excel з табличками

```
import regex
import collections
import unicodedata
text_war_peace = open("D:\Web-devoloped\Krpt\lab1_text.txt").read()
text_war_peace = text_war_peace.lower().replace("ъ","").replace("\n"," ")
text_war_peace = ' '.join(text_war_peace.split())

#without space
string_text = ''.join(c for c in text_war_peace if unicodedata.category(c).startswith('L'))

#with space
# string_text = regex.sub(r'[^\w\s]+|[\d]+', r'',text_war_peace).strip()


word_sort = sorted(string_text)
print(string_text)
print("OK")
```

✓ 0.9s

```
левтолстойвойнаимиртомапервыйивторойвшкловскийвойнаимирльватолстогозамыселвгодупоявилосьобъявле
OK
```

```python
import regex
import collections
import unicodedata
text_war_peace = open("D:\Web-devoloped\Krpt\lab1_text.txt").read()
text_war_peace = text_war_peace.lower().replace("n","").replace("\n"," ")
text_war_peace = ' '.join(text_war_peace.split())

#without space
# string_text = ''.join(c for c in text_war_peace if unicodedata.category(c).startswith('L'))

#with space
string_text = regex.sub(r'[^\w\s]+|[\d]+', r'',text_war_peace).strip()


word_sort = sorted(string_text)
print(string_text)
print("OK")
```

✓ 0.7s

```
лев толстой война и мир тома первый и второй в шкловский война и мир льва толстого замысел в  году
OK
```

```python
#Count

list_alp = []
for i in range(0,len(word_sort)):
    list_alp.append(word_sort[i])

alpcount = dict(collections.Counter(list_alp))
print("\n")
freq = {k: alpcount[k] / len(word_sort) for k in alpcount}
print_dict_in_columns(alpcount)
print("\n")
print_dict_in_columns(freq)
```

✓ 0.5s

```
а: 2713 б: 472  с: 1465 д: 1302 е: 6306 ф: 413  г: 357
h: 560  i: 2950 j: 277  k: 64   l: 2024 м: 1637 о: 2457
п: 1044 q: 415  r: 2656 s: 2995 т: 2345 u: 2457 v: 885
w: 39   x: 271  y: 88   z: 219  а: 104486       б: 21826        в: 56730
г: 24829        д: 37700        е: 101669       ж: 12686        з: 21449        и: 81798        й: 14642
к: 42714        л: 61662        м: 37334        н: 81257        о: 140492       п: 31603        р: 55571
с: 65937        т: 71753        у: 34128        ф: 2493 х: 10631        ц: 4526 ч: 17392
ш: 11754        щ: 3582 ъ: 548  ы: 23457        ь: 24396        э: 3840 ю: 8110
я: 27868        ё: 62


а: 0.0021272825357396 01 b: 0.0003700985465790976        c: 0.0011487168871575804        d: 0.0010209074314533582        e: 0.004944579310863961 f: 0.0003238362282567104        g: 0.0002799262312049 53
h: 0.0004390999705175734        i: 0.0023131159161193598        j: 0.0002171976639881568 6        k: 5.018285377343696e-05        l: 0.0015870327505849 44 m: 0.0012835833066736923        o: 0.0019265511206458534
p: 0.0008186078021791904        q: 0.0003254044424371303        r: 0.0020825884315976337        s: 0.0023484007351788076        t: 0.0018387311265423386        u: 0.0019265511206458534        v: 0.0006939347748358079
w: 3.0580176518188 15e-05        x: 0.0002124930214468971 4        y: 6.900142393847582e-05        z: 0.0001717194527559796        а: 0.0819282134276771   б: 0.0171139213509224 24 в: 0.0444823952276106 05
г: 0.0194685949428229 12 д: 0.029560837300915 21  е: 0.0797193837545556 6  ж: 0.0009471825464034 58 з: 0.0168183129779133 73 и: 0.0641383917649937   й: 0.0114808960148541 24
к: 0.0334923502512279 1  л: 0.0483496113965260 9  м: 0.0292738541058983 7  н: 0.0637141898291901 1  о: 0.1101607733177766 4  п: 0.0247801363719051 3  р: 0.0435736151100572 7
с: 0.0517016692071736 4  т: 0.0562620360438347 2  у: 0.0267600067746852 6  ф: 0.0019547789758934 116        х: 0.0083358424760220 04 ц: 0.0035488686902902453        ч: 0.0136371905129314 95
ш: 0.0092163947383277 82 щ: 0.0028086715971320 5  ъ: 0.0004296906854350 5397       ы: 0.0183928000150548 57 ь: 0.0191290765727620 03 э: 0.0030109712264062 18 ю: 0.0063591085016027 15
я: 0.0218514963899709 56 ё: 4.861463959301706e-05
```

```python
#print count
import pandas as pd
sort_letters = sorted(alpcount, key=lambda x : alpcount[x],reverse=1)

list_freq = []
for i in range(0,len(sort_letters)):
    list_freq.append(alpcount[sort_letters[i]])

data = pd.DataFrame(index = sort_letters)
data['count'] = list_freq
print(data.head(10))
data.to_excel("liters.xlsx")
```

✓ 7.5s

```
   count
о  140492
а  104486
е  101669
и   81798
н   81257
т   71753
с   65937
л   61662
в   56730
р   55571
```

```python
#print freq
frequency= sorted(freq, key=lambda x : alpcount[x],reverse=1)

relative_frequencies = []
for letter in frequency:
    relative_frequencies.append(freq[letter])

# Створюємо DataFrame для відносних частот
frequency_data = pd.DataFrame(index=frequency)
frequency_data['relative_frequency'] = relative_frequencies

# Виводимо перші кілька рядків DataFrame
print(frequency_data.head(10))

# Експортуємо дані в Excel
frequency_data.to_excel("relative_frequencies.xlsx")
```

✓ 0.1s

	relative_frequency
о	0.110161
а	0.081928
е	0.079719
и	0.064138
н	0.063714
т	0.056262
с	0.051702
л	0.048350
в	0.044482
р	0.043574

```python
from collections import Counter
import math

# Задаємо текст і алфавіт
input_text = string_text
alphabet = ['а', 'б', 'в', 'г', 'д', 'е', 'ё', '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']

# Створюємо біграми з фільтрацією
bigrams = []
for i in range(len(input_text) - 1):
    if input_text[i] in alphabet and input_text[i + 1] in alphabet:
        bigrams.append(input_text[i] + input_text[i + 1])

# Створюємо "двійні" біграми з фільтрацією
double_bigrams = []
for i in range(0, len(input_text) - 2, 2):
    if input_text[i] in alphabet and input_text[i + 1] in alphabet:
        double_bigrams.append(input_text[i] + input_text[i + 1])



# Підрахунок частот біграм
bigram_counts = dict(Counter(bigrams))
bigram_frequencies = {bigram: count / len(bigrams) for bigram, count in bigram_counts.items()}

# Підрахунок частот "двійних" біграм
double_bigram_counts = dict(Counter(double_bigrams))
double_bigram_frequencies = {bigram: count / len(double_bigrams) for bigram, count in double_bigram_counts.items()}

# Підрахунок частот монограм
monogram_counts = dict(Counter(input_text))
monogram_frequencies = {letter: count / len(input_text) for letter, count in monogram_counts.items() if letter in alphabet}


# Обчислення ентропії H1 для монограм
monogram_entropy_contributions = []
for frequency in monogram_frequencies.values():
    if frequency > 0:  # Переконайтеся, що частота не дорівнює нулю
        monogram_entropy_contributions.append(frequency * math.log(frequency, 2))

# Обчислюємо загальну ентропію H1 для монограм
H1_monogram = -sum(monogram_entropy_contributions)
print("Entropy H1 ", H1_monogram)
print(monogram_entropy_contributions)
```

```python
# Кількість символів в алфавіті
N = len(alphabet)

# Обчислення надлишковості для монограм
R_monogram = 1 - (H1_monogram / math.log2(N))
print("Redundancy:", R_monogram)
```

✓ 4.1s

```
Entropy H1  4.380281723653518
[-0.2113048158131395, -0.2908901027374183, -0.19975360995937094, -0.23358278149913916, -0.35056655703894185, -0.2209545971883068, -0.07399002297865534, -0.25308814757323445, -0.29571954512145887, -0.254159153281074,
Redundancy: 0.13165355045441962
```

```python
import numpy as np

# Функція для обчислення ентропії
def calculate_entropy(bigram, freq_bigram):
    entropy_contributions = []
    for frequency in freq_bigram.values():
        entropy_contributions.append(frequency * math.log(frequency, 2) if frequency > 0 else 0)
    return -sum(entropy_contributions) / 2

# Обчислення ентропії для біграм і двійних біграм
H_bigrams = calculate_entropy(bigrams, bigram_frequencies)
H_double_bigrams = calculate_entropy(double_bigrams, double_bigram_frequencies)

print("Entropy of bigrams (H2):", H_bigrams)
print("Entropy of double bigrams (H22):", H_double_bigrams)

# Обчислення надлишковості
R_bigrams = 1 - (H_bigrams / math.log2(N))
R_double_bigrams = 1 - (H_double_bigrams / math.log2(N))

# Виведення надлишковості
print("Redundancy R (bigrams):", R_bigrams)
print("Redundancy R (double bigrams):", R_double_bigrams)

# Створення DataFrame з алфавіту
frequency_df = pd.DataFrame(index=alphabet, columns=alphabet)

# Створення всіх можливих біграм з алфавіту
bigrams_list = [i + j for i in alphabet for j in alphabet]

# Заповнення DataFrame біграмами
for idx, letter in enumerate(alphabet):
    frequency_df[letter] = bigrams_list[idx * len(alphabet):(idx + 1) * len(alphabet)]

# Транспонування DataFrame для правильного формату
frequency_df = frequency_df.T

# Заповнення DataFrame частотами біграм
for bigram in bigram_frequencies.keys():
    row_idx, col_idx = np.where(frequency_df == bigram)
    frequency_df.iloc[row_idx, col_idx] = bigram_frequencies[bigram]
```

```python
    # Встановлення 0 для біграм, які не мають частоти
    for bigram in bigrams_list:
        row_idx, col_idx = np.where(frequency_df == bigram)
        frequency_df.iloc[row_idx, col_idx] = 0

    # Вивід фінального DataFrame (необов'язково)
    print(frequency_df)
```

✓ 0.5s

```
Entropy of bigrams (H2): 4.14798745486047
Entropy of double bigrams (H22): 4.148029783764964
Redundancy R (bigrams): 0.17770353451525878
Redundancy R (double bigrams): 0.1776951432390237
          а         б         в         г         д         е         ё
а  0.000337  0.001553  0.005636  0.001614  0.003114  0.001772         0
б  0.001273  0.000025  0.000103  0.000011  0.000027  0.002457         0
в  0.006747  0.000258   0.00048  0.000441  0.000769  0.005354         0
г   0.00116  0.000038  0.000128  0.000011  0.001181  0.000654         0
д  0.004996  0.000088  0.001093  0.000064  0.000077  0.005291  0.000001
е  0.000218  0.002334  0.003611  0.004567  0.003916  0.002243         0
ё  0.000002  0.000003  0.000006  0.000002  0.000002  0.000002         0
ж  0.001473  0.000081  0.000027  0.000026  0.000832  0.004267         0
з  0.006234  0.000207  0.000981   0.00055  0.000959  0.000302         0
и  0.000355  0.001397  0.005116  0.001258  0.002888  0.003424         0
й  0.000208  0.000408  0.000792  0.000364  0.000785  0.000126         0
к   0.00858  0.000518  0.000627  0.000176  0.000261  0.000636  0.000002
л  0.008445   0.00034  0.000823  0.000409  0.000624  0.004992  0.000004
м  0.003523  0.000338  0.000815  0.000376  0.000402  0.003456         0
н  0.012856  0.000352  0.000531   0.00031  0.001429  0.010714         0
о  0.000217  0.005194  0.011612  0.006031  0.006358  0.003333  0.000001
п  0.001264  0.000001  0.000001  0.000015  0.000002   0.00292         0
р  0.009785  0.000261  0.000512  0.000409  0.000505  0.007073         0
с  0.001787  0.000207  0.002371  0.000166  0.000549  0.004165  0.000039
т  0.006795  0.000283  0.003355  0.000143  0.000411  0.005487  0.000004
...
ю         0         0         0   0.00005  0.000046  0.000045
я         0         0         0   0.00018  0.000132  0.000293
```

```python
# Створення DataFrame з алфавіту
frequency_df = pd.DataFrame(index=alphabet, columns=alphabet)

# Генерація всіх можливих біграм з алфавіту
bigrams_list = [letter1 + letter2 for letter1 in alphabet for letter2 in alphabet]

# Заповнення DataFrame біграмами
for idx, letter in enumerate(alphabet):
    frequency_df[letter] = bigrams_list[idx * len(alphabet):(idx + 1) * len(alphabet)]

# Транспонування DataFrame для правильного формату
frequency_df = frequency_df.T

# Заповнення DataFrame частотами двійних біграм
for bigram in double_bigram_frequencies.keys():
    row_idx, col_idx = np.where(frequency_df == bigram)
    frequency_df.iloc[row_idx, col_idx] = double_bigram_frequencies[bigram]

# Встановлення 0 для біграм, які не мають частоти
for bigram in bigrams_list:
    row_idx, col_idx = np.where(frequency_df == bigram)
    frequency_df.iloc[row_idx, col_idx] = 0

# Вивід фінального DataFrame
print(frequency_df.head())

# Збереження DataFrame у файл Excel
frequency_df.to_excel("double_bigram_frequencies.xlsx")
```

✓ 0.7s

```
         а         б         в         г         д         е  ё         ж  \
а  0.00032  0.001537  0.005553  0.001611  0.002976  0.001749  0  0.001443
б  0.001226   0.00002  0.000125  0.000008  0.000028  0.002376  0  0.000008
в  0.006535  0.000274  0.000464  0.000428  0.000756  0.005116  0  0.000053
г  0.001096  0.000036  0.000116  0.000011  0.001104  0.000632  0  0.000005
д  0.004786  0.000083  0.001059  0.000075  0.000066  0.005152  0  0.000024

         з         и  ...         ц         ч         ш         щ         ъ  \
а  0.004863  0.001559  ...  0.000136  0.001331  0.001676  0.000265         0
б  0.000009  0.000889  ...  0.000006   0.00002  0.000008  0.000252   0.00011
в  0.000665  0.003762  ...  0.000058  0.000238  0.001247  0.000009  0.000028
г  0.000045  0.000941  ...  0.000003  0.000035  0.000009         0         0
д  0.000052  0.002937  ...  0.000202  0.000075  0.000127  0.000002   0.00021

         ы         ь         э         ю         я
а         0         0  0.000345  0.000949  0.003096
```

Результат

Таблиця частот по файлам:

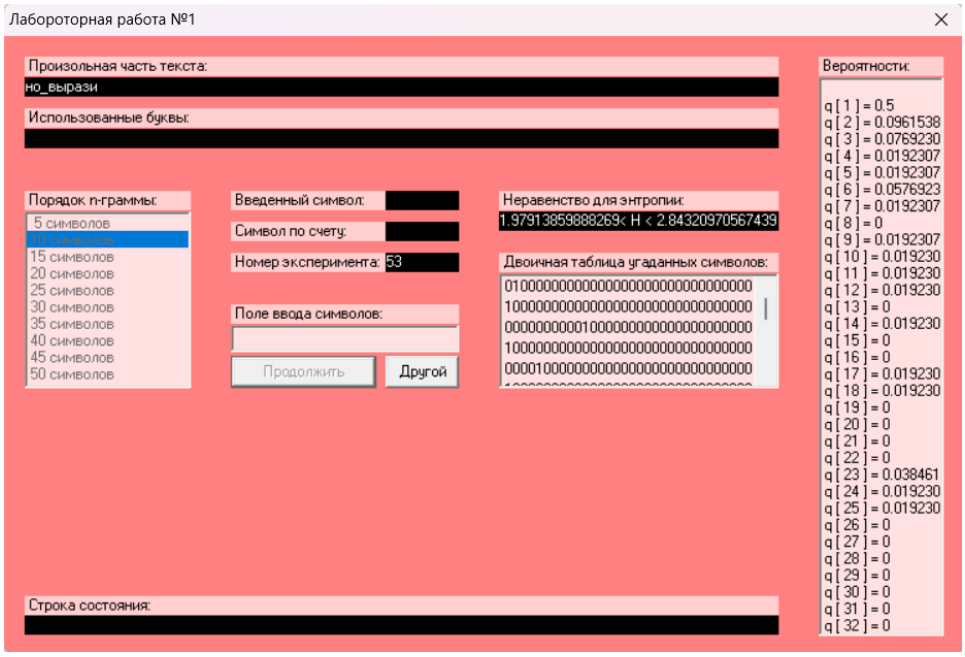
	Текст без пробілів	Текст з пробілами
Букви	liters.xlsx	liters_space.xlsx
Біграма	relative_frequencies.xlsx	relative_frequencies_space.xlsx
Біграма з кроком 2	double_bigram_frequencies.xlsx	double_bigram_frequencies_space.xlsx

Ентропія

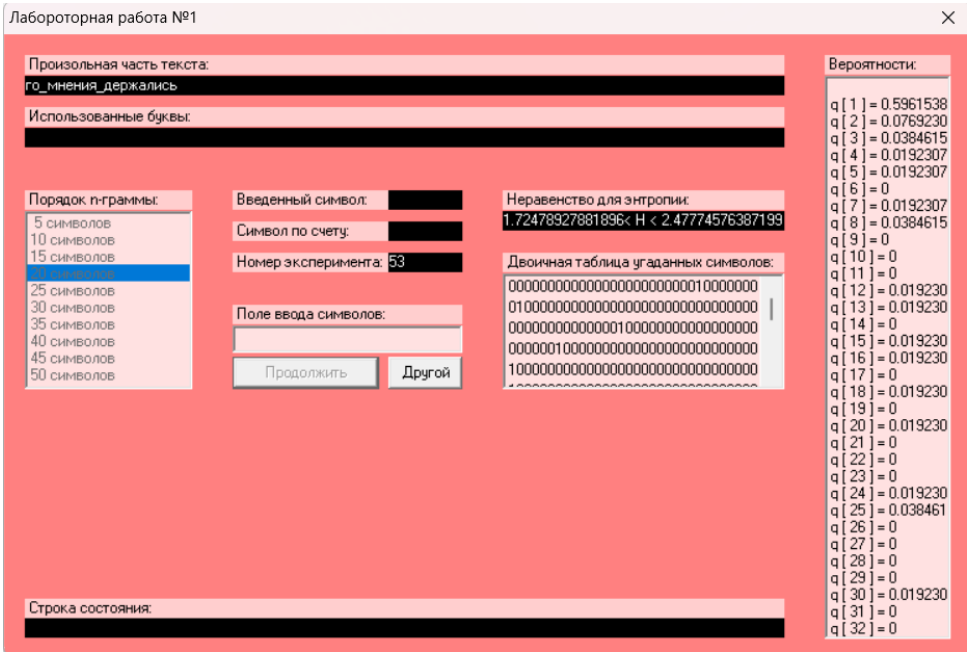
	Текст без пробілів	Текст з пробілами
H1	4.380281723653518	3.8548255361596637
R1	0.13165355045441962	0.23581991316532558
H2	4.14798745486047	3.9667620592447643
R2	0.17770353451525878	0.21362963214514696
H2 (з кроком 2)	4.148029783764964	3.966857027999631
R2 (з кроком 2)	0.1776951432390237	0.2136108055521767

Пункт 2. За допомогою програми *CoolPinkProgram* оцінити значення $H^{(10)}$, $H^{(20)}$, $H^{(30)}$.

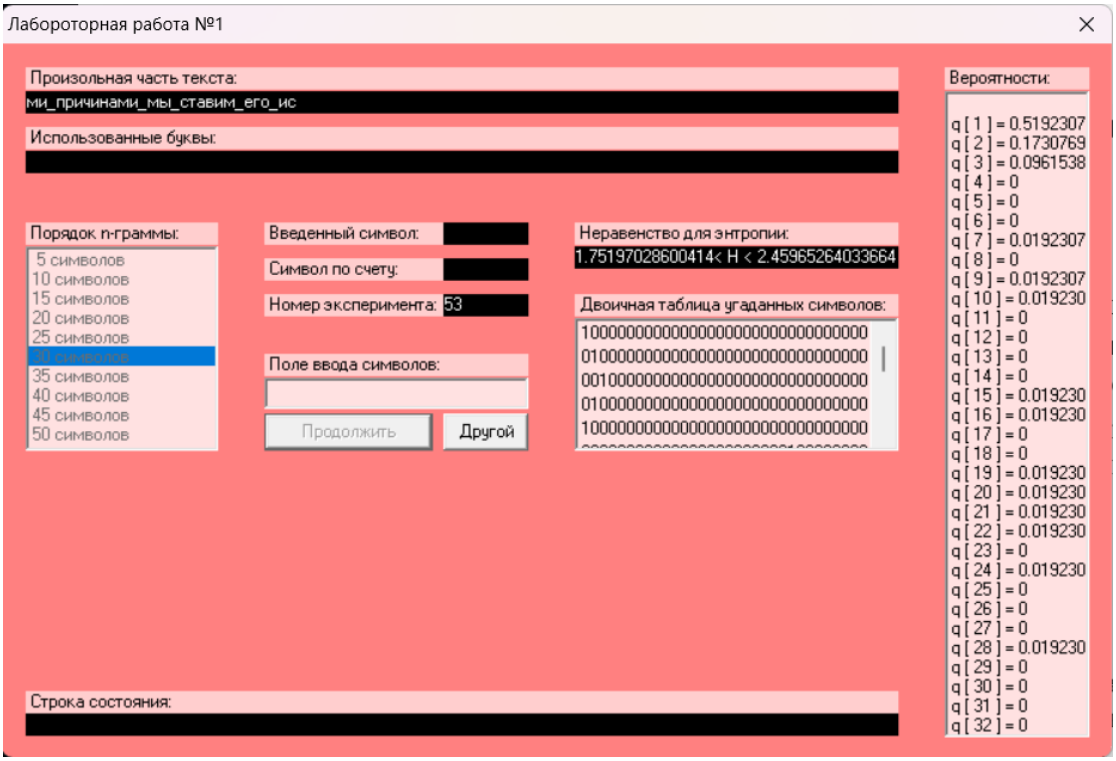
- Для $H^{(10)}$:



- Для $H^{(20)}$:



- Для $H^{(30)}$:



	Ентропія	Надлишковість
$H^{(10)}$	1.97914 < H < 2.84321	0.43136 < H < 0.60417
$H^{(20)}$	1.72479 < H < 2.47775	0.50445 < H < 0.65504
$H^{(30)}$	1.75197 < H < 2.45965	0.50807 < R < 0.64961

Висновок

Ознайомилися з поняттями ентропії на символ джерела та його надлишковості, навчилися визначати частоти літер та біграм на довільному тексті.

З отриманих значень можемо визначити, що ентропія H1 з пробілами більша за H1 без пробілів. Ентропія та надлишковість перехресних та неперехресних біграм без пробілів (або з пробілами) (H2 та H2(з кроком 2)) майже не відрізняються.