КОМП’ЮТЕРНИЙ ПРАКТИКУМ №1

Гоголєва Поліна ФБ-12

**Експериментальна оцінка ентропії на символ джерела відкритого тексту**

**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.

**Хід роботи**

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

Єдиний пункт, з котрим не було проблем і складнощів! (формули налякали not gonna lie)

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.

Для початку напишемо частину коду, що буде загружати текст з файлу та фільтрувати його відносно вимог методички, цитую:  
*«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.»*

Як текстовий файл я обрала біблію, там води точно більше ніж на 1Мб.

import re

file\_path = r"C:\Users\Polya\Desktop\KPI\crypto-23-24-main\tasks\cp1\solution\bible.txt"

with open(file\_path, "r") as file:

cleaned\_text = ''

for line in file:

line = re.sub(r'[^а-яА-ЯёЁ]', ' ', line)

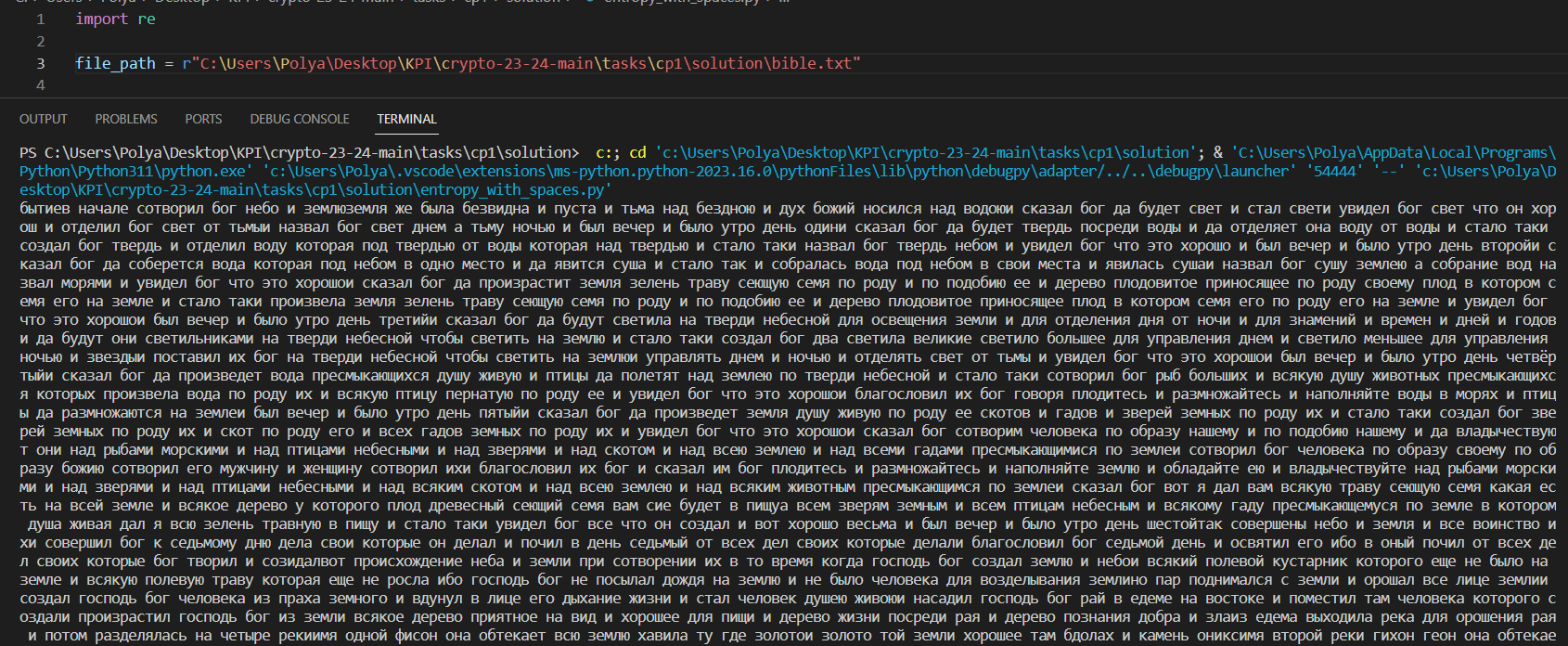
# Видалення зайвих пробілів

line = ' '.join(line.split())

cleaned\_text += line

print (cleaned\_text)

print написала просто щоб потестити чи все правильно працює, бідний пітон ледь не вмер друкуючи всю біблію, але все працює, а всі послідовності символів і пробілів замінились на один пробіл.



Тепер рахуватимемо частоти букв і біграм.

Отримуємо частотний аналіз літер у Біблії (моя віруюча бабуся зараз напевно у захваті, що я цікавлюсь Біблією). Для цього створимо словник зі значенням літера:її кількість у тексті, прогоним весь алфавіт через цикл і додаватимо до countera одиничку щоразу, як зустрічатимемо літеру.

import re

file\_path = r"C:\Users\Polya\Desktop\KPI\crypto-23-24-main\tasks\cp1\solution\bible.txt"

with open(file\_path, "r") as file:

cleaned\_text = ''

for line in file:

line = re.sub(r'[^а-яА-ЯёЁ]', ' ', line)

# Видалення зайвих пробілів

line = ' '.join(line.split())

cleaned\_text += line.lower()

print (cleaned\_text)

letter\_counts={}

for letter in cleaned\_text:

if letter in letter\_counts:

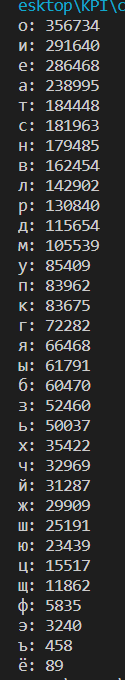
letter\_counts[letter] += 1

else:

letter\_counts[letter] = 1

sorted\_letter\_counts = dict(sorted(letter\_counts.items(), key=lambda item: item[1], reverse=True))

(тут я поняла, що одна літера upper case і та ж літера lower case вважається за дві різні літери алфавіту і спростила собі життя, переводячи весь текст у нижній регістр).

Тож так ми дізнались, що літера «о» зустрічається у Біблії 356734 разів:  


Тепер рахуватимемо частотність біграм з перетином і без, принцип як і у минулому пункті, але беремо по два символи і ітеруємо через кожні 2( у випадку без перетину) або 1(у випадку з перетином) символ.

Нюанси: При обрахунках ми від довжини тексту віднімаємо 1, оскільки для підрахунку біграм потрібно розглядати пари символів, і останній символ не може бути початком нової біграми.   
Також для підрахунту частот біграм з перетином ми витягаємо пару символів з тексту, розпочинаючи з індексу i і закінчуючи індексом i + 2. Це створює біграму з перетином, оскільки дві послідовні пари символів будуть мати одну спільну літеру (перший символ наступного біграму є останнім символом попереднього).

Вийшло щось отаке:

bigram\_with\_overlap\_counts = {}

bigram\_without\_overlap\_counts = {}

for i in range(len(cleaned\_text) - 1):

#Підрахуємо частоту біграм з перетином

bigram\_with\_overlap = cleaned\_text[i:i + 2]

if bigram\_with\_overlap in bigram\_with\_overlap\_counts:

bigram\_with\_overlap\_counts[bigram\_with\_overlap] += 1

else:

bigram\_with\_overlap\_counts[bigram\_with\_overlap] = 1

if i % 2 == 0:

#І без перетину

bigram\_without\_overlap = cleaned\_text[i:i + 2]

if bigram\_without\_overlap in bigram\_without\_overlap\_counts:

bigram\_without\_overlap\_counts[bigram\_without\_overlap] += 1

else:

bigram\_without\_overlap\_counts[bigram\_without\_overlap] = 1

sorted\_bigram\_with\_overlap\_counts = dict(sorted(bigram\_with\_overlap\_counts.items(), key=lambda item: item[1], reverse=True))

sorted\_bigram\_without\_overlap\_counts = dict(sorted(bigram\_without\_overlap\_counts.items(), key=lambda item: item[1], reverse=True))print("Частотний аналіз біграм з перетином:")

for bigram, count in sorted\_bigram\_with\_overlap\_counts.items():

print(f"{bigram}: {count}")

print("Частотний аналіз біграм без перетину:")

for bigram, count in sorted\_bigram\_without\_overlap\_counts.items():

print(f"{bigram}: {count}")

Вивід виявився величеньким і не вміщався у термінал, тому я закинула його у текстові файли, щоб була візуалізація і було з чого робити таблички частотності і прибрала прінт, приблизно отак це виглядало для кожної з функцій

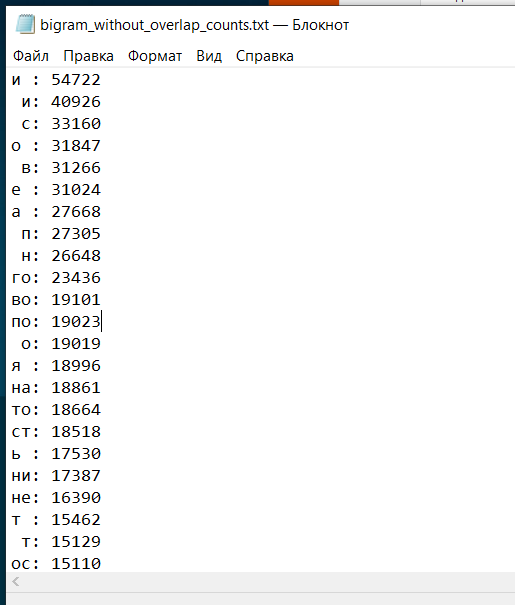
with open("letters\_count.txt", "w") as letters\_file:

letters\_file.write("Частотний аналіз літер у тексті:\n")

for letter, count in sorted\_letter\_counts.items():

letters\_file.write(f"{letter}: {count}\n")

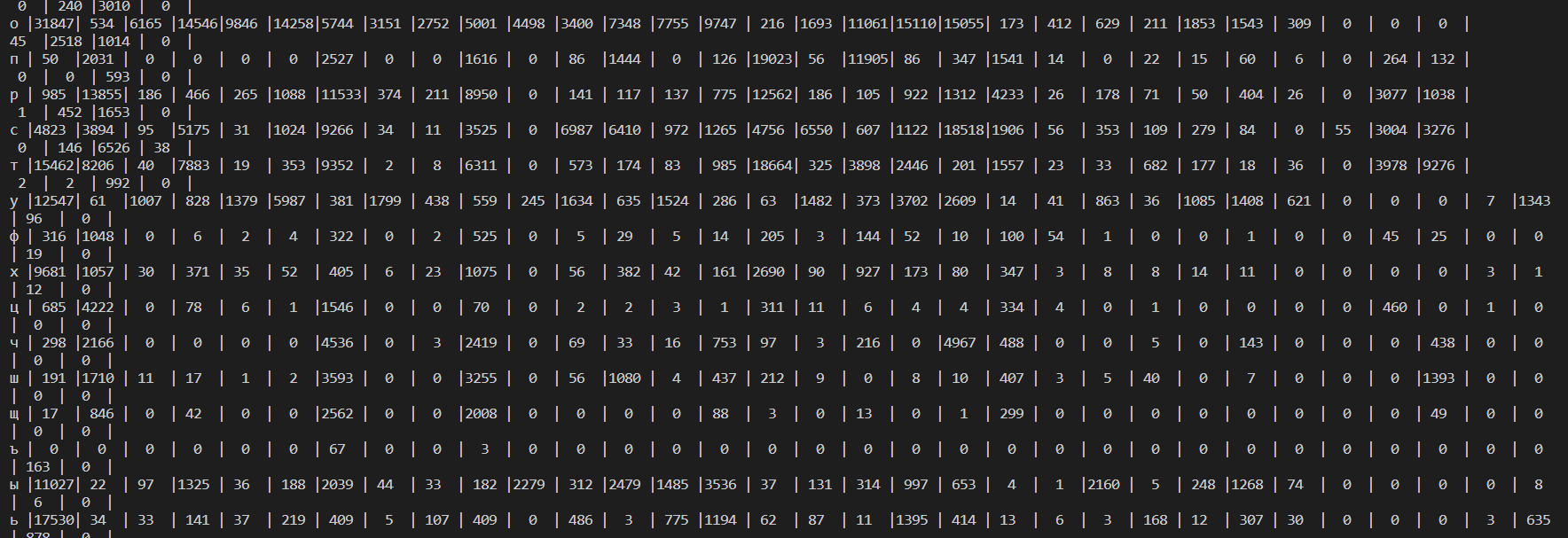
Тому тепер у мене є файлики з частотним аналізом біграм і літер. Так як ми вважаємо, що пробіл є частиною нашого алфавіту, то його частотність теж підраховується і біграми формату «пробіл-літера» теж існують.



Ну тепер треба створити таблички частот букв та біграм, для більш зручної візуалізації і та сприйняття даних.

Таблиця частот букв:

|  |  |
| --- | --- |
| Символ | Частотність |
| пробіл | 640177 |
| о | 356734 |
| и | 291640 |
| е | 286468 |
| а | 238995 |
| т | 184448 |
| с | 181963 |
| н | 179485 |
| в | 162454 |
| л | 142902 |
| р | 130840 |
| д | 115654 |
| м | 105539 |
| у | 85409 |
| п | 83962 |
| к | 83675 |
| г | 72282 |
| я | 66468 |
| ы | 61791 |
| б | 60470 |
| з | 52460 |
| ь | 50037 |
| х | 35422 |
| ч | 32969 |
| й | 31287 |
| ж | 29909 |
| ш | 25191 |
| ю | 23439 |
| ц | 15517 |
| щ | 11862 |
| ф | 5835 |
| э | 3240 |
| ъ | 458 |
| ё | 89 |

Таблицю частот біграм так просто не перепишеш, тому я модифікувала код так, щоб він сам створював та виводив матрицю. Але вивід виявився нечитабельним, бо просто не вміщався на екран

Тому я вирішила експортувати результати у файли, додавши у код отаку функцію  
  
def save\_matrix\_to\_file(matrix, file\_name):

with open(file\_name, "w") as output\_file:

alphabet = sorted(sorted\_letter\_counts.keys())

output\_file.write(f" |" + "|".join(f"{letter:^5}" for letter in alphabet) + "|\n")

for letter1 in alphabet:

row = []

for letter2 in alphabet:

bigram = letter1 + letter2

count = matrix.get(bigram, 0)

row.append(f"{count:^5}")

output\_file.write(f"{letter1} |" + "|".join(row) + "|\n")

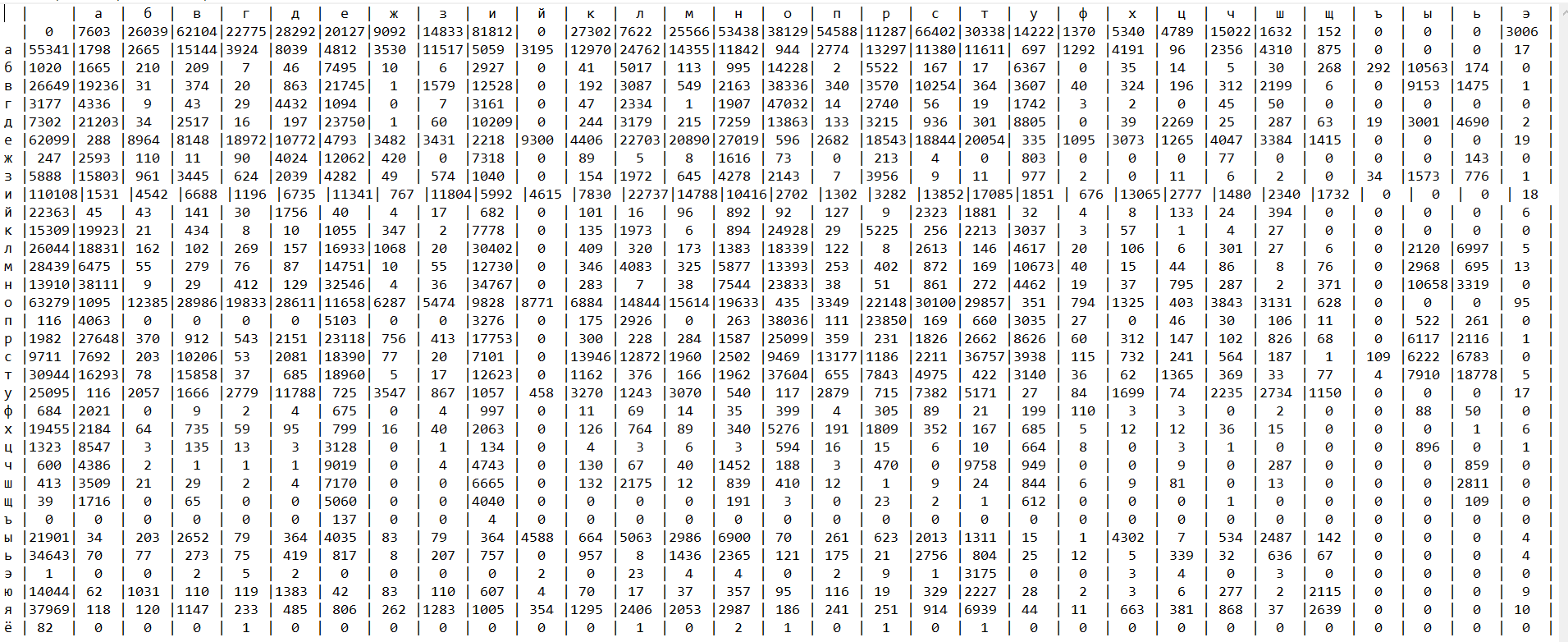
# Зберігаємо матрицю біграм з перетином

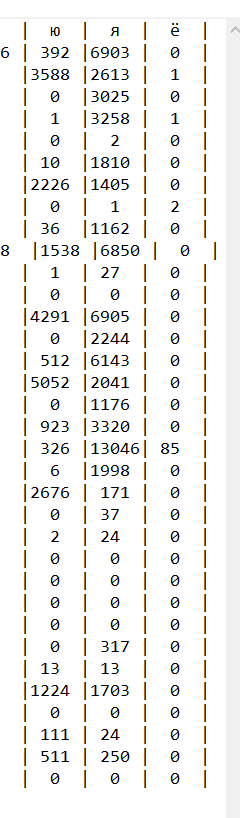
save\_matrix\_to\_file(bigram\_with\_overlap\_counts, "bigram\_with\_overlap\_matrix.txt")

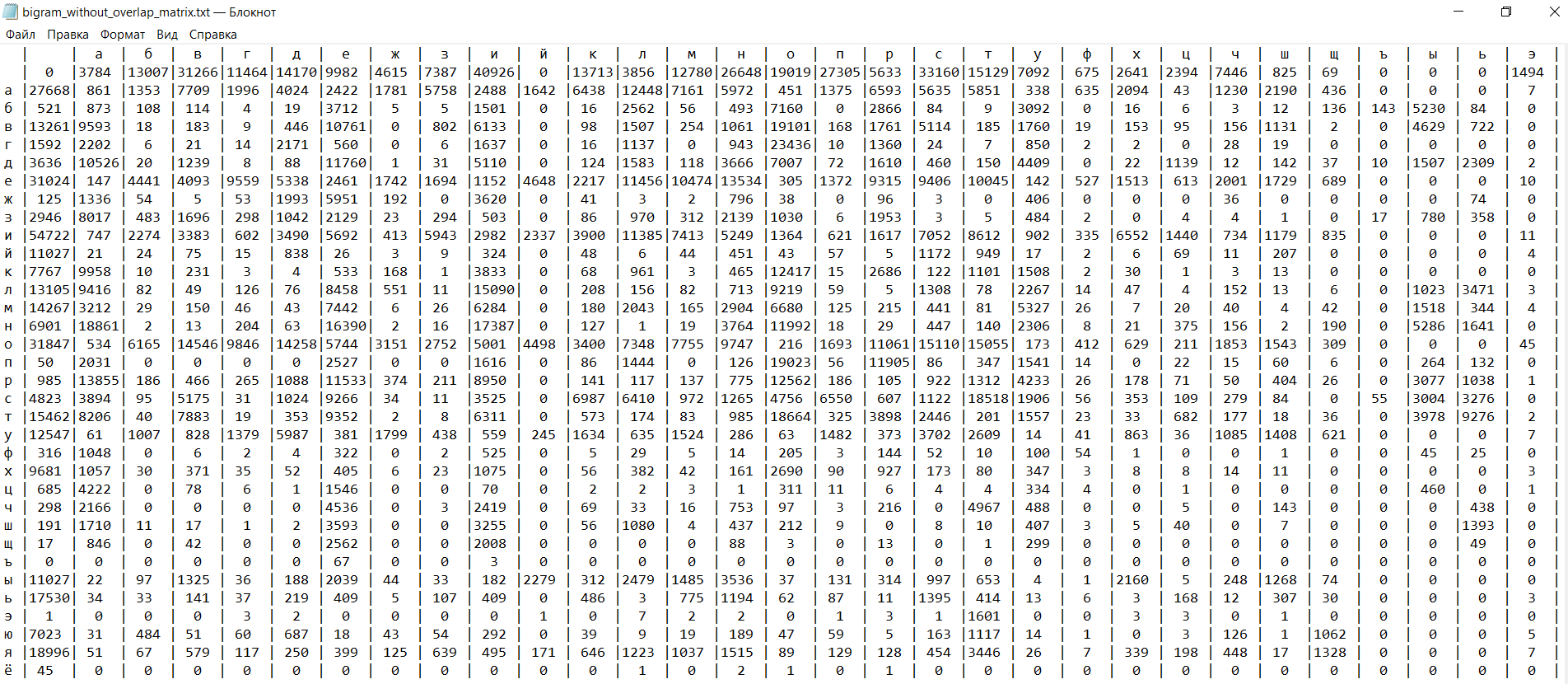
# Зберігаємо матрицю біграм без перетину

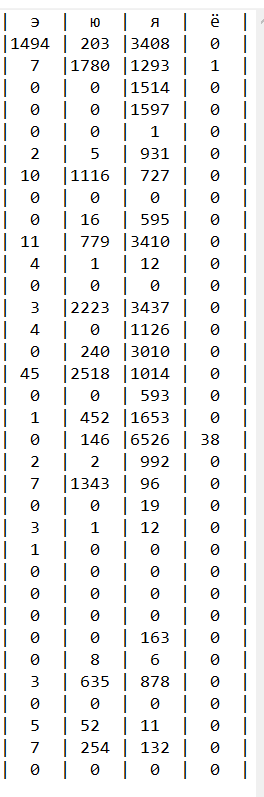
save\_matrix\_to\_file(bigram\_without\_overlap\_counts, "bigram\_without\_overlap\_matrix.txt")

Запустили і отримали такі результати для біграм з перетином:



(на один екран все одно не вмістилось вибачте)

І для біграм без перетину:  




Таблиці частотності отримали, тож видаляємо функцію для друку матриць, щоб на це не витрачався час при кожному запуску.

Тепер нам треба знайти за означенням H1 та H2, тобто ентропію для окремих літер і для біграм у тексті, але у формулі імовірності замінити відповідними частотами, що ми тільки що знайшли.

Ентропія тексту обчислюється як сума ентропій для кожної літери (H1) або біграми (H2) окремо. Тобто ми обчислюємо ентропію для кожного можливого символу або біграми в тексті і потім сумуємо ці значення. Це дає загальну ентропію тексту і вказує на ступінь невизначеності або «непередбачуваності» тексту.

Загальні формули для ентропії:

***H*1(*X*)=−∑*i*=1*n*​*P*(*xi*​)⋅log2​(*P*(*xi*​))**

***H*2(*X*)=−∑*i*=1*n*​*P*(*bigrami*​)⋅log2​(*P*(*bigrami*​))**

Ну а у нашому випадку, ми замість P(xi) i P(bigram) ставитимо обраховане значення частоти для цієї літери або біграми.

Пройшла ніч і я зрозуміла, щоб не порушувати логіку формули ентропії ми можем розрахувати ці ймовірності, просто поділивши частотність літер/біграм на загальну кількість літер/біграм. Бо просто замінивши значення P на частотність навряд вийде розрахувати ентропію за означенням, бо ми викинемо логічну частину формули. Тому мій розрахунок виглядає так:

# Підрахунок ентропії для літер (Н1)

entropy\_letter = 0.0

total\_letters = len(cleaned\_text)

for count in sorted\_letter\_counts.values():

probability = count / total\_letters

entropy\_letter -= probability \* math.log2(probability)

print("H1 у тексті з пробілами:", entropy\_letter)

# Ентропія для біграм з перетином (H2)

entropy\_bigram\_with\_overlap = 0.0

total\_bigrams\_with\_overlap = len(cleaned\_text) - 1

for count in sorted\_bigram\_with\_overlap\_counts.values():

probability = count / total\_bigrams\_with\_overlap

entropy\_bigram\_with\_overlap -=(probability \* math.log2(probability))/2

print("H2 з перетином у тексті з пробілами:", entropy\_bigram\_with\_overlap)

# Ентропія для біграм без перетину (H2)

entropy\_bigram\_without\_overlap = 0.0

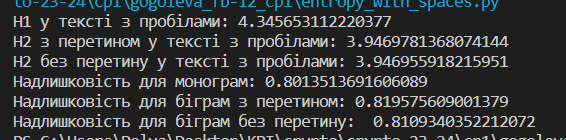
total\_bigrams\_without\_overlap = len(cleaned\_text) // 2

for count in sorted\_bigram\_without\_overlap\_counts.values():

probability = count / total\_bigrams\_without\_overlap

entropy\_bigram\_without\_overlap -= (probability \* math.log2(probability))/2

print("H2 без перетину у тексті з пробілами:", entropy\_bigram\_without\_overlap)

Запустимо і збережемо значення:  


Тепер виконаємо ті ж самі розрахунки, але попередньо вилучивши всі пробіли з тексту. Для цього створю окремий файл.

Код виглядатиме так само, просто з трошки зміненим кодом фільтрації тексту  
with open(file\_path, «r») as file:

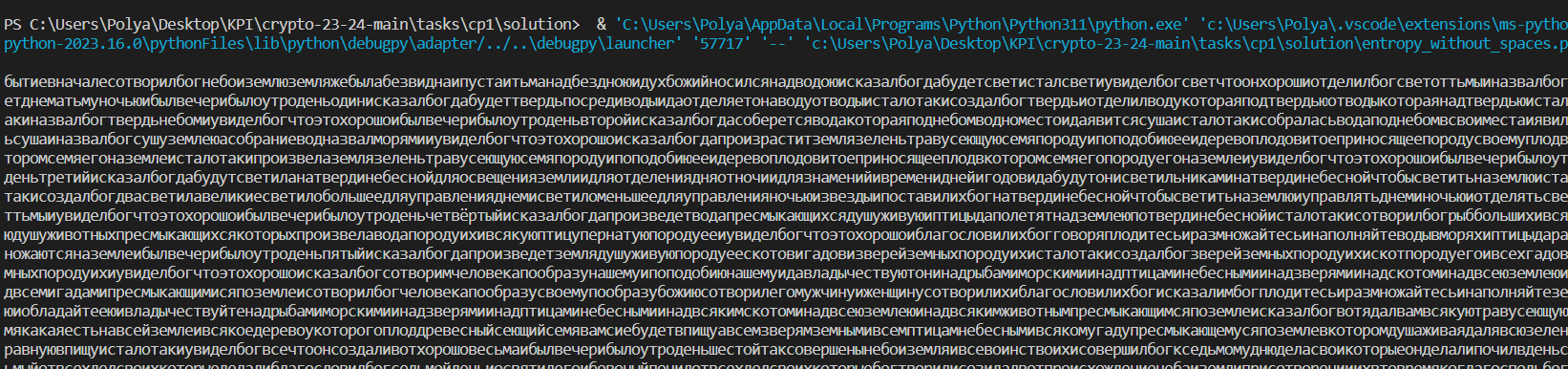
cleaned\_text = ‘’

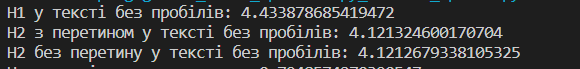
for line in file:

line = re.sub(r’[^а-яА-ЯёЁ]’, ‘’, line) # Вилучення всіх символів окрім літер

cleaned\_text += line.lower()

print (cleaned\_text)

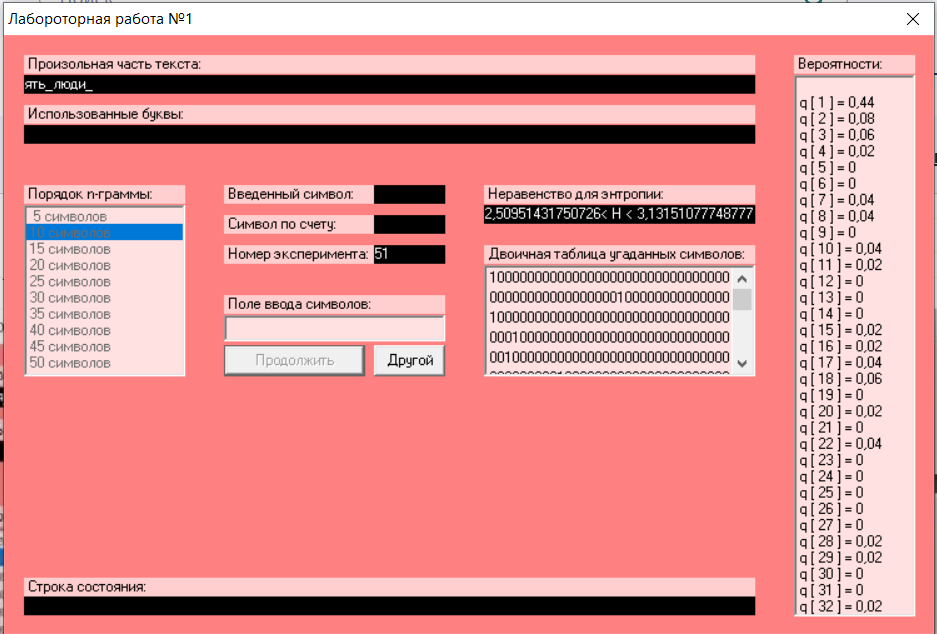
Тепер наш текст виглядатиме отак  


А значення ентропії:  


1. За допомогою програми CoolPinkProgram оцінити значення (10)H , (20)H, (30)H .

* Оцінюємо значення для 10 символів:

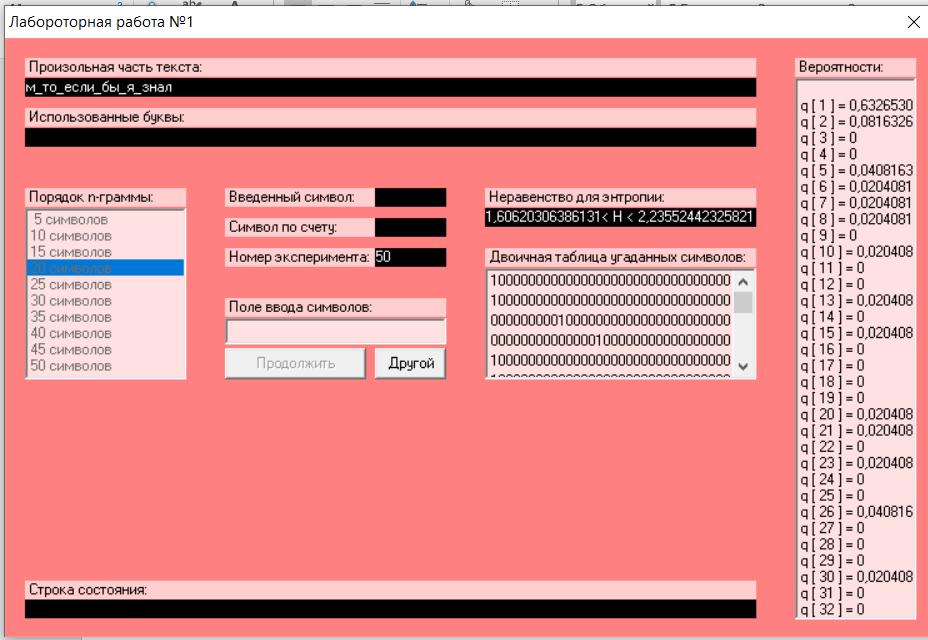
Нерівність для ентропії:  

* Для 20 символів:

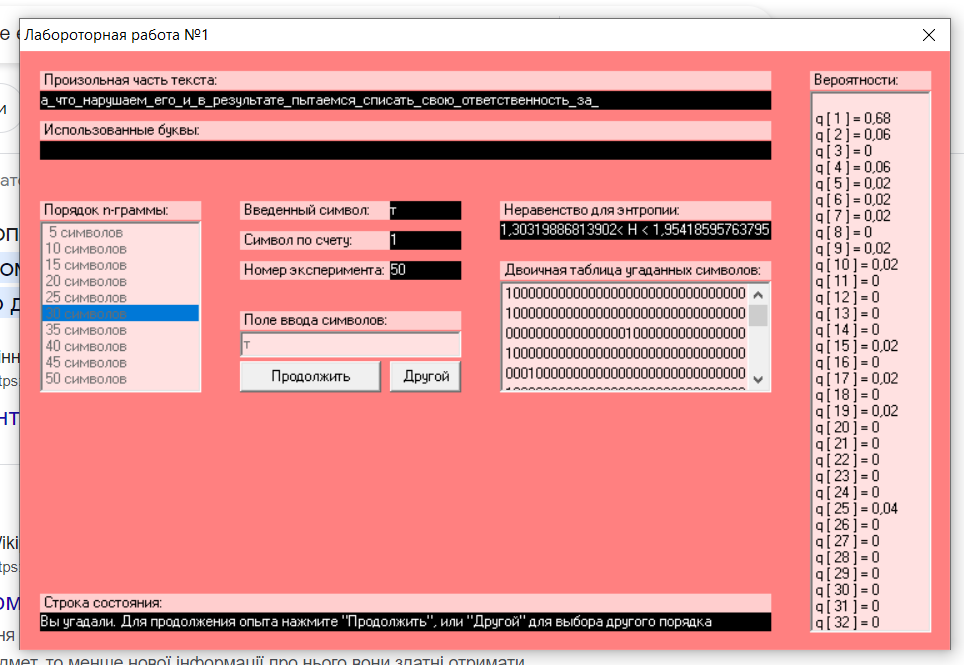
Нерівність для ентропії:





* Для 30 символів:

Нерівність для ентропії:  

За моїми спостереженнями, чим більше символів ми бачимо – тим менше загальне значення ентропії = менша «хаотичність» і невизначеність тексту.

Давайте тепер розрахуємо ще й межі надлишковості для н-грам різної довжини, використовуючи отримані значення ентропії. Напишемо для цього такий код:

import math

h\_down\_10symbols=2.50951431750726

h\_up\_10symbols=3.13151077748777

h\_down\_20symbols=1.60620306386131

h\_up\_20symbols=2.23552442325821

h\_down\_30symbols=1.30319886813902

h\_up\_30symbols=1.95418595763795

r\_down\_for\_10symbols = 1 - (h\_down\_10symbols/math.log2(34))

r\_up\_for\_10symbols = 1 - (h\_up\_10symbols/math.log2(34))

r\_down\_for\_20symbols = 1 - (h\_down\_20symbols/math.log2(34))

r\_up\_for\_20symbols = 1 - (h\_up\_20symbols/math.log2(34))

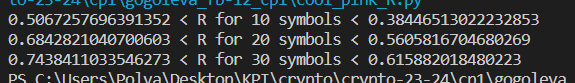
r\_down\_for\_30symbols = 1 - (h\_down\_30symbols/math.log2(34))

r\_up\_for\_30symbols= 1 - (h\_up\_30symbols/math.log2(34))

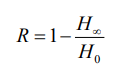
print(r\_down\_for\_10symbols, "< R for 10 symbols <", r\_up\_for\_10symbols)

print(r\_down\_for\_20symbols,"< R for 20 symbols <", r\_up\_for\_20symbols)

print(r\_down\_for\_30symbols,"< R for 30 symbols <", r\_up\_for\_30symbols)

Отримали результати:  


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

Надлишковість джерела відкритого тексту (мови) розраховується за формулою:  
, де ми прийматимемо за H(inf) наші розраховані значення, а H0 це стале значення, котре = кількості символів у алфавіті, у випадку тексту з пробілами це 33 літери + пробіл =34.

r\_for\_monograms = 1 - (entropy\_letter/math.log2(34))

r\_for\_bigram\_with\_overlap = 1 - (entropy\_bigram\_with\_overlap/math.log2(34))

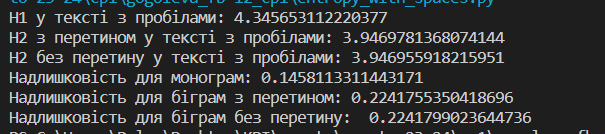
r\_for\_bigram\_without\_overlap = 1 - (entropy\_bigram\_without\_overlap/math.log2(34))

print("Надлишковість для монограм:",r\_for\_monograms)

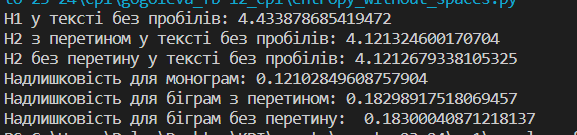
print("Надлишковість для біграм з перетином:",r\_for\_bigram\_with\_overlap)

print("Надлишковість для біграм без перетину: ",r\_for\_bigram\_without\_overlap)

Результат:



Тепер розрахуємо це значення для джерела без пробілів. (Тут буде логарифм 33 по основі 2)



**Висновки**

Під час виконання цієї лабораторної роботи я не тільки покращила свої навички у програмуванні, а й зрозуміла, чому значення ентропії і надлишковості є важливими у кібербзпеці, а не нас просто вирішили задовбати вищою математикою.

Як ми визначили, ентропія - це міра невизначеності чи непередбачуваності. У контексті тексту ентропія вимірює, наскільки випадковим є розподіл символів чи біграм у тексті (у нашому випадку, можна так то рахувати триграми і т.д.) . Чим більше ентропія, тим більше невизначеність та непередбачуваність.

Різниця ентропій для тексту з пробілами та без пробілів може бути пояснена тим, що пробіли - це також символи, котрі вважаються частиною алфавіту, і вони додаються до загальної кількості символів у тексті. Якщо текст має багато пробілів, то розподіл символів буде менш випадковим, оскільки пробіли будуть домінувати серед інших символів. Таким чином, ентропія тексту з пробілами буде нижчою порівняно з текстом без пробілів.

Надлишковість - це міра того, наскільки розподіл символів чи біграм у тексті відрізняється від рівномірного розподілу. Висока надлишковість вказує на невизначеність мови та може бути використана для криптографічних цілей, так як важко передбачити наступний символ. Висока надлишковість джерела мови вказує на те, що мова має багато невизначеності та непередбачуваних залежностей між символами чи біграмами. Це означає, що для побудови криптосистеми буде складно передбачити, які символи чи біграми випадуть наступними, що робить шифрування більш надійним перед атакою.

В криптографії, ентропія та надлишковість важливі, оскільки вони впливають на стійкість криптосистем. Висока ентропія тексту робить його складнішим для аналізу і підірвання. Надлишковість джерела мови може використовуватися для побудови надійних шифрів, де важко передбачити, який символ або біграм вийде наступним.

Тож ми розібрались з базовими поняттями та принципами роботи криптографії та надійності шифрування, було весело 😊