# КРИПТОГРАФІЯ

## КОМП’ЮТЕРНИЙ ПРАКТИКУМ №1

### Експериментальна оцінка ентропії на символ джерела відкритого тексту

#### Мета

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

Мовою програмування для створення скриптів було обрано Python 3. Тому весь процес відбувався досить швидко, зручно і з малою кількістю дописування специфічних функцій.

Більшість труднощів, які виникали під час роботи, були пов’язані з:

* відсутністю досвіду роботи з GitHub;
* недостатньою компетентністю у математичних основах криптології (поняття ентропії, умовної ентропії), що фундаментально призупиняла написання коду;
* бажанням якомога краще оптимізувати набір створених функцій;
* технічними недоліками CoolPinkProgram.exe на основній і гостьовій ОС;
* поганим розумінням умови завдання.

Першу і другу проблеми було вирішено завдяки поглибленню у відповідні теми (здебільшого шляхом перегляду навчальних відео).

Приводом до появи третьої проблеми можна вважати відносну подібність між функціями або, навпаки, занадто розбіжні дії всередині однієї функції. Розглянемо пару таких інцидентів:

* **count\_ngram()** та **find\_ngram()** зведені до однієї **ngram\_processing()**.  
  Цей випадок не зазначений в історії репозиторію, бо відбувся він через третю причину, яку ми обговоримо пізніше.   
  Так сталося, що мені потрібно було написати функцію для обрахування умовної ентропії. З’ясувалося, що наявної **entropy()** цілком достатньо, і треба лише дописати функції для підрахування “умовних” ймовірностей - **conditional\_probability()**. Суть її полягала у тому, щоб підрахувати, скільки (n-1)-грам відповідає певній n-грамі, якщо n-1 їх символів однакові. Для цього було замало (хоча, скоріше, забагато) count\_ngram() і з’явилася потреба у функції пошуку n-грам без їх підрахунку. Так була створена find\_ngram(). Проте пізніше була помічена суттєва схожість між їх циклами, що вмотивувало об’єднати ці дві функції в одну **ngram\_processing()** з додатковим параметром, котрий визначав суть обробки.  
  Врешті-решт, хоча пошук і був визнаний непотрібним, він був залишений для повноти коду.
* Відділення від **read\_file()** функції **format\_text()**.  
  Ця історія значно коротша за попередню.  
  Таке рішення було прийнято для більшої універсальності і повноти коду (типу, “а що як користувач не хоче досліджувати кастрований текст?”).

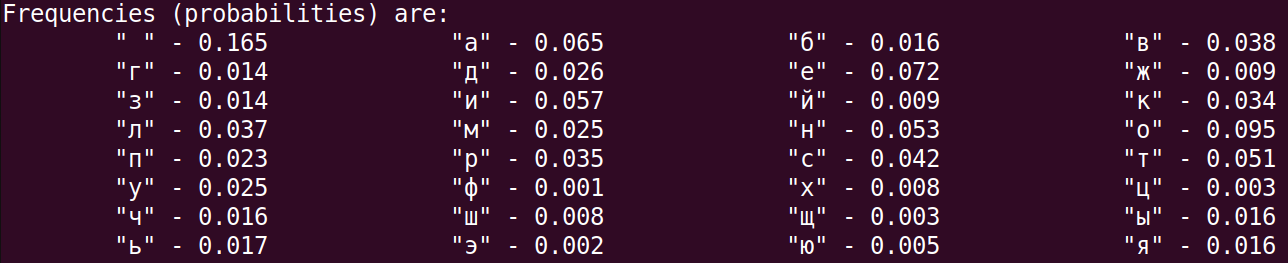
Крім того, досить нелегко було вигадати самому, як представити частоту (ймовірність) появи біграм у вигляді матриці. Довелося поблукати мережею, і після недовгих пошуків таки-знайти [зрозуміле і естетичне рішення](https://www.educba.com/python-print-table/).

Третя проблема гальмувала увесь процес виконання лабораторної, адже спотворювало текст CoolPinkProgram.exe. З’ясувалося, що вона виникла через повну відсутність росязьіка у моїх системах. Була вирішена за пару хвилин.

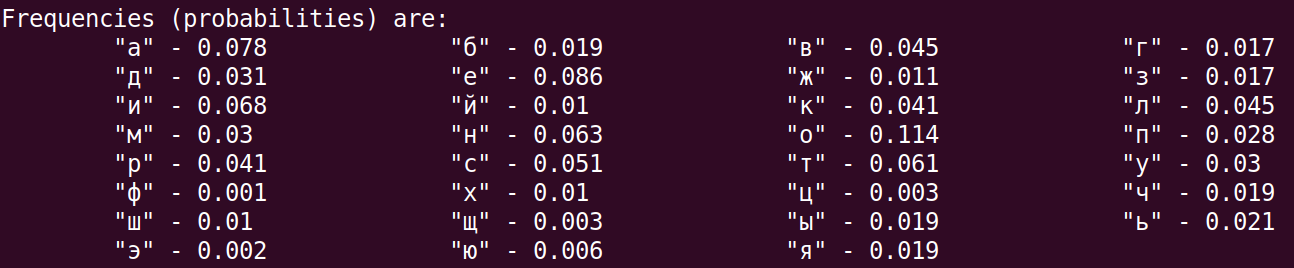
Остання проблема має генетично-середовищне походження, тому не будемо на ній зупинятися (дякую за терпіння і відповіді на питання).

Розглянемо, отримані у ході виконання лабораторної роботи, дані. Варто зазначити, що обрахунки відбувалися для текстового файлу large.txt дуже значного обсягу (“Мертві душі” разом з “Війною та миром” фрагментами “Злочина і кари”).

1. Частота літер (міститься у прикладеному файлі data.txt).

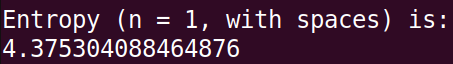


Знімок 1. Частоти літер у тексті з пробілами.

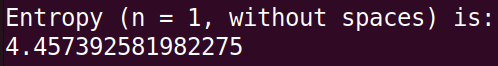


Знімок 2. Частоти літер у тексті без пробілів.

1. Частота біграм (міститься у прикладеному файлі data.txt).
2. Ентропія та (міститься у прикладеному файлі data.txt).



Знімок 3. для тексту з пробілами.



Знімок 4. для тексту без пробілів.

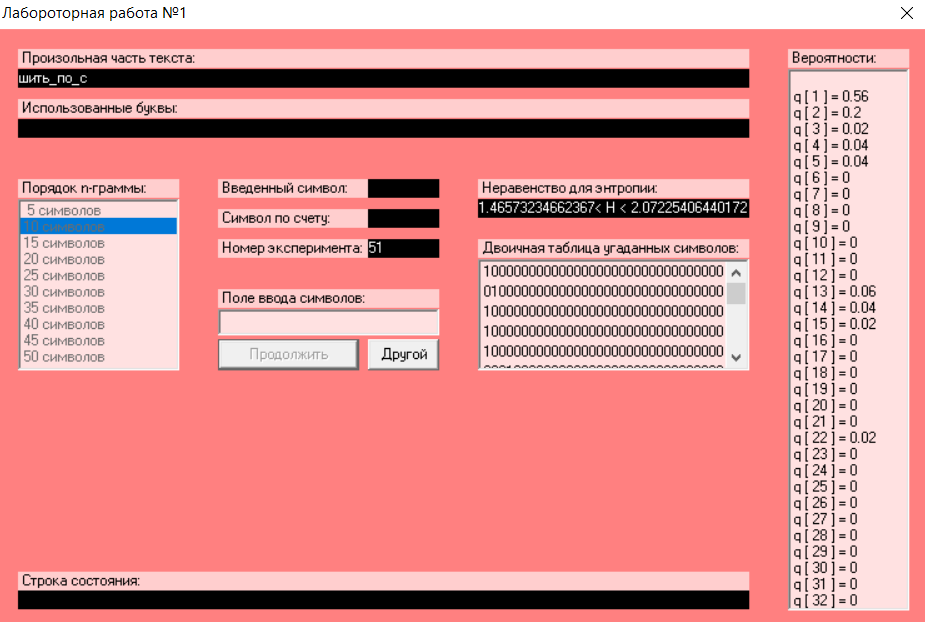


Знімок 5. для тексту з пробілами.

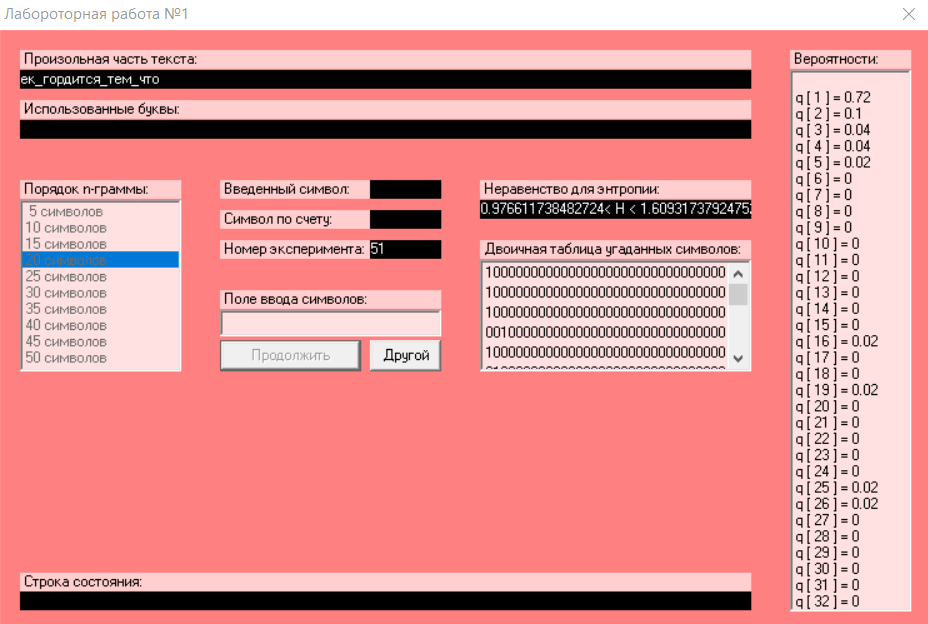


Знімок 6. для тексту без пробілів.

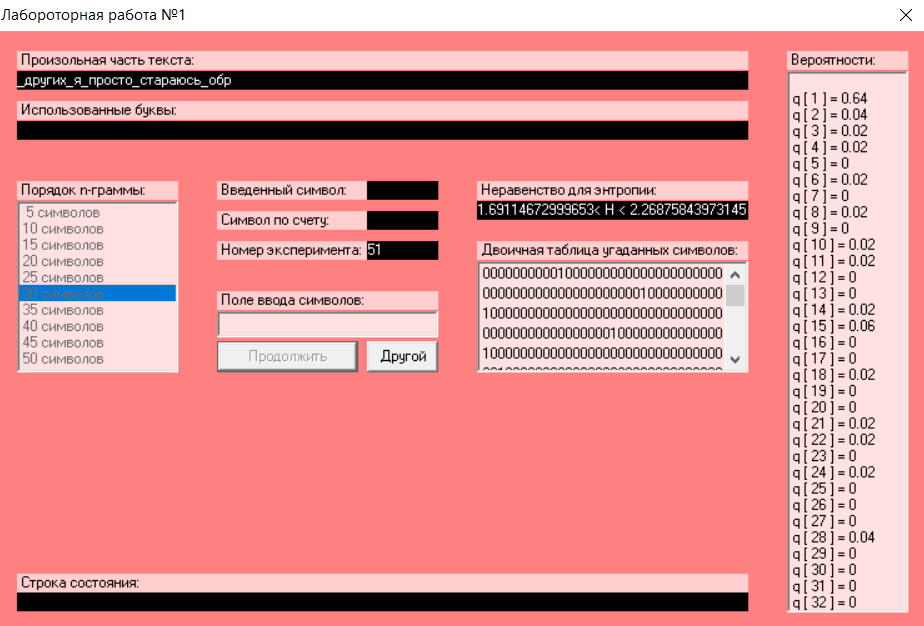
1. Оцінки для , та .



Знімок 7. Отримане значення .



Знімок 8. Отримане значення .



Знімок 9. Отримане значення .

Варто зазначити, що отримані значення значно нижчі за інші, бо рядки для цього експеримента були найпередбачуванішими, а саме ті, що:

* закінчуються на кінці слова (наступний символ, вочевидь, пробіл);
* закінчувались на середині слова;
* вгадувались за сенсом.

У інших експериментах досить часто траплялися непередбачувані рядки, як-от такі, що:

* закінчувались на пробіл;
* закінчувались на першій букві наступного слова.

1. Оцінка надлишковості R російської мови.

Здійснена за формулою:

Значення надлишковості для тексту з пробілами:

Значення надлишковості для тексту без пробілів:

Можна встановити закономірність, що надлишковість росязьіка зростає зі збільшенням розмірів n-грами і алфавіту.

#### Висновок

У ході виконання лабораторної роботи було обчислено частоти та ентропію літер і біграм у текстах російською мовою з пробілами та без них. Крім того, було висунуте припущення щодо надлишковості росязьіка при великих значеннях n - при збільшенні значення n та розмірів алфавіту кількість (корисної) падає.