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 No 1

Виконав : студент групи ФБ-24 Луняка Артем

Київ – 2024

**Тема:**

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.

**Варіант 10**

***Завдання до виконання***

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 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.

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.

**1. Написання програм для виконання роботи.**

В якості мови програмування виберемо Python. Напишемо декілька службових модулів та основну програму для виконання лабораторної роботи. У службових модулях опишемо декілька класів та функцій, які ми зможемо використовувати і надалі для схожих задач.

**2. Модуль alphabet.**

Модуль alphabet призначено для дій над алфавітом: отримання номерів за літерами, отримання літер за номерами тощо. Основна функціональність зібрана у класі Alphabet.

UKR\_CAPS = "АБВГҐДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ"  
RUSSIAN\_CAPS = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"  
LATIN\_CAPS = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"  
  
class Alphabet:  
  
 def \_\_init\_\_(self, alphabet\_string=""):  
 self.letters\_dict = {c: i for i, c in enumerate(alphabet\_string)}  
 self.numbers\_dict = dict(zip(self.letters\_dict.values(), self.letters\_dict.keys()))  
  
 self.\_m = len(alphabet\_string)  
  
 @property  
 def m(self):  
 return self.\_m  
  
 def get\_number(self, letter):  
 return self.letters\_dict.get(letter)  
  
 def get\_letter(self, number):  
 return self.numbers\_dict.get(number)  
  
 def get\_numbers\_list(self, letters):  
 return [self.get\_number(c) for c in letters]  
  
 def get\_letters\_list(self, numbers):  
 return [self.get\_letter(n) for n in numbers]  
  
 def get\_all\_letters(self):  
 return list(self.letters\_dict.keys())  
  
  
latin\_alfabet = Alphabet(LATIN\_CAPS)  
ukr\_alphabet = Alphabet(UKR\_CAPS)  
rus\_alphabet = Alphabet(RUSSIAN\_CAPS)

Окрім класу Alphabet у цьому модулі описано декілька об’єктів для латини українського алфавіту, російського алфавіту.

**3. Модуль text\_source.**

Модуль text\_source призначено для роботи з текстовими джерелами. Текст міститься у деякому файлі та має бути перетворений перед його подальшою обробкою. Такі перетворення назвемо фільтруванням. Основну функціональність у модулі виконує клас TextSource. Для завдання текстового джерела треба вказати шлях до файлу з текстом, алфавіт тексту та, можливо, назви фільтрів. Кожен фільтр – це метод класу TextSource, який має суфікс \_filter, а початок – назва фільтру. Наприклад, to\_lower\_filter.

Окрім фільтрів клас містить також метод apply\_filter\_chain, який застосовує до тексту ланцюг фільтрів, перетворюючи цей текст.

Після компіляції та запуску цієї програми бачимо очікуваний результат.

from alphabet import Alphabet, ukr\_alphabet  
  
STANDARD\_DELIMETERS = '.,:;()"\'0123456789+-\_№#&!?/|\\%$«»\*—„“–…=[]{}'  
  
  
class TextSource:  
  
 def \_\_init\_\_(self, file\_path=None, alphabet: Alphabet = ukr\_alphabet, \*filters):  
 self.\_alphabet = alphabet  
 self.\_start\_source = ""  
 if file\_path is not None:  
 with open(file\_path, 'r', encoding="utf-8") as f:  
 self.\_start\_source: str = f.read()  
 self.\_filter\_chain = list()  
 self.\_source\_filtered = list()  
 for filter\_name in filters:  
 self.add\_filter(filter\_name)  
  
 def add\_filter(self, filter\_name):  
 filt = getattr(self, filter\_name + "\_filter", None)  
 if filt is None:  
 return False  
  
 self.\_filter\_chain.append(filt)  
 return True  
  
 def apply\_filter\_chain(self):  
 source = self.\_start\_source  
 for filt in self.\_filter\_chain:  
 source = filt(source)  
 self.\_source\_filtered = self.delete\_non\_alphabet\_letters(source)  
  
 @property  
 def source\_filtered(self):  
 return self.\_source\_filtered  
  
 @property  
 def size(self):  
 return len(self.\_source\_filtered)  
  
 @staticmethod  
 def to\_lower\_filter(source: str):  
 return source.lower()  
  
 @staticmethod  
 def to\_upper\_filter(source: str):  
 return source.upper()  
  
 @staticmethod  
 def leave\_one\_space\_filter(source: str):  
 return ' '.join(source.split())  
  
 @staticmethod  
 def delete\_spaces\_filter(source: str):  
 return ''.join(source.split())  
  
 @staticmethod  
 def delete\_delimeters\_filter(source: str, delim=STANDARD\_DELIMETERS):  
 delimeters = set(delim)  
 filtered = [c if c not in delimeters else "" for c in source]  
 return ''.join(filtered)  
  
 @staticmethod  
 def replace\_delimeters\_filter(source: str, delim=STANDARD\_DELIMETERS):  
 delimeters = set(delim)  
 filtered = [c if c not in delimeters else ' ' for c in source]  
 return ''.join(filtered)  
  
 @staticmethod  
 def replace\_ru\_yo\_hard\_filter(source: str, to\_replace=(('ё', 'е'), ('ъ', 'ь'))):  
 to\_replace\_dict = dict(to\_replace)  
 filtered = [c if c not in to\_replace\_dict else to\_replace\_dict[c] for c in source]  
 return ''.join(filtered)  
  
 def delete\_non\_alphabet\_letters(self, source: str):  
 alphabet\_letters = set(self.\_alphabet.get\_all\_letters())  
 return [c for c in source if c in alphabet\_letters]

**4.** **Модуль ngram**

Модуль ngram призначено для роботи з нграмами: уніграмами(окремими символами), біграмами, триграмами тощо. Цю роботу виконаує клас NGrams. Для створення об’єкту класу NGrams треба передати алфавіт, кількість символів нграми, текстове джерело. У конструкторі класу(\_\_init\_\_) створюються два словники: словник кількості входжень нграм у текстове джерело та словник частот входжень нграм у текстове джерело. Самі нграми створюються рекурсивним методом \_construct\_ngrams.

Заповнення вказаних вище словників виконує метод feed, який може враховувати нграми, що не перетинаються, або такі, що перетинаються.

Окрім цих методів у класі описано методи для повернення словників кількостей входжень та частот нграм. Якщо параметр to\_sort встановити рівним True, на початку словників, що повертаються, будуть нграми з більшими значеннями кількості входжень або частот.

from copy import copy  
  
from alphabet import Alphabet  
from text\_source import TextSource  
  
  
class NGrams:  
  
 def \_\_init\_\_(self, alphabet: Alphabet, length, source: TextSource):  
 self.\_alphabet = alphabet  
 self.\_length = length  
 self.\_source = source  
 self.\_ngrams = {ngram: 0 for ngram in self.\_construct\_ngrams(self.\_length)}  
 self.\_frequencies = {ngram: 0.0 for ngram in self.\_ngrams}  
 self.\_errors = dict()  
  
 def \_construct\_ngrams(self, n):  
 if n == 0:  
 return ['']  
  
 ngrams = []  
 for i in range(self.\_alphabet.m):  
 letter = self.\_alphabet.get\_letter(i)  
 for gram in self.\_construct\_ngrams(n - 1):  
 ngrams.append(letter + gram)  
 return ngrams  
  
 def feed(self, intersected=False):  
 step = 1 if intersected else self.\_length  
 *# calculate occurencies* for i in range(0, self.\_source.size, step):  
 text\_ngram = ''.join(self.\_source.source\_filtered[i: i + self.\_length])  
 if text\_ngram not in self.\_ngrams:  
 self.\_errors[text\_ngram] = self.\_errors.get(text\_ngram, 0) + 1  
 continue  
  
 self.\_ngrams[text\_ngram] += 1  
  
 *# calculate frequencies* divisor = self.\_source.size // step  
 if divisor == 0:  
 return  
  
 for ngram in self.\_ngrams:  
 self.\_frequencies[ngram] = self.\_ngrams[ngram] / divisor  
  
 def get\_ngrams\_quantities(self, to\_sort=False):  
 if not to\_sort:  
 return copy(self.\_ngrams)  
  
 return dict(sorted(self.\_ngrams.items(), key=lambda item: item[1], reverse=True))  
  
 def get\_ngrams\_frequencies(self, to\_sort=False):  
 if not to\_sort:  
 return copy(self.\_frequencies)  
  
 return dict(sorted(self.\_frequencies.items(), key=lambda item: item[1], reverse=True))  
  
 @property  
 def errors(self):  
 return self.\_errors

**5. Основна програма**

Основна програма для лабораторної роботи міститься у модулі lab1\_reworked. У цьому модулі, зокрема описано константи для вказаного у роботі російського алфавіту: маленькі літери без «ё», «ъ», без пропуску або з пропуском(пробілом). Так само описано функції:

get\_ngrams\_entropy – отримати ентропію за заданими частотами нграм

calculate\_entropies – порахувати значення ентропії уніграм, біграм, що не перетинаються, біграм, що перетинаються, для заданого текстового джерела. Ентропії повертаються у вигляді словника.

show\_monogram\_frequencies – показати таблицю частот монограм по порядку спадання частот

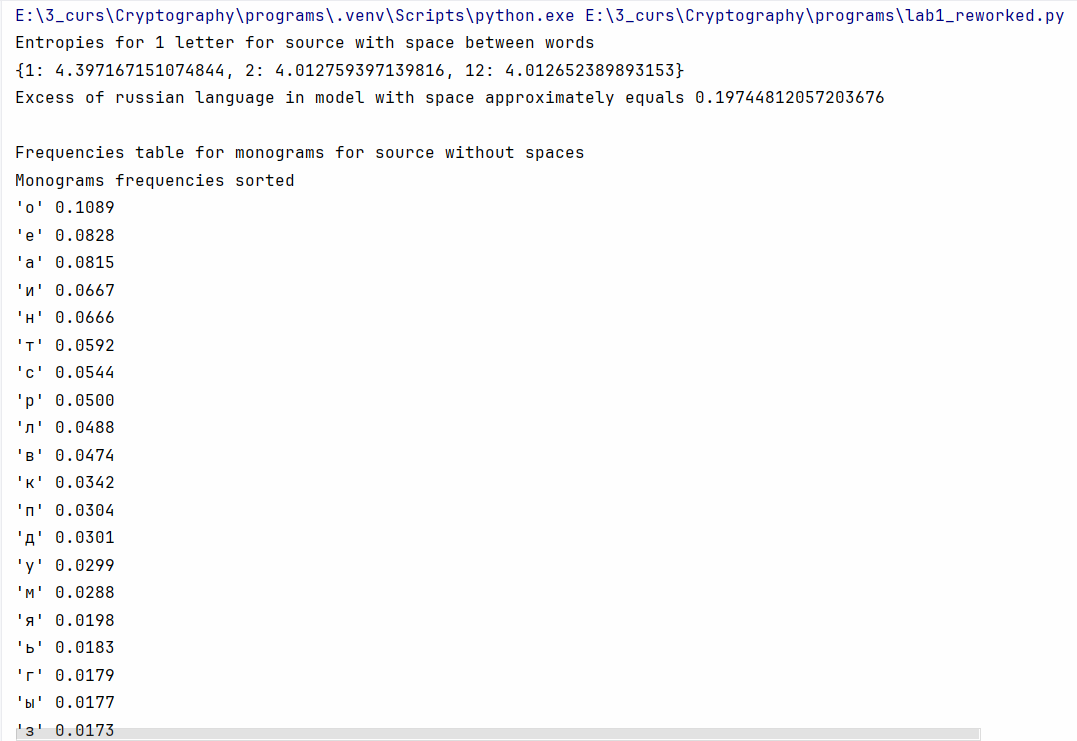
show\_bigram\_frequencies – показати таблицю частот біграм. Найбільш часті біграми розташовані біля лівого верхнього кута.

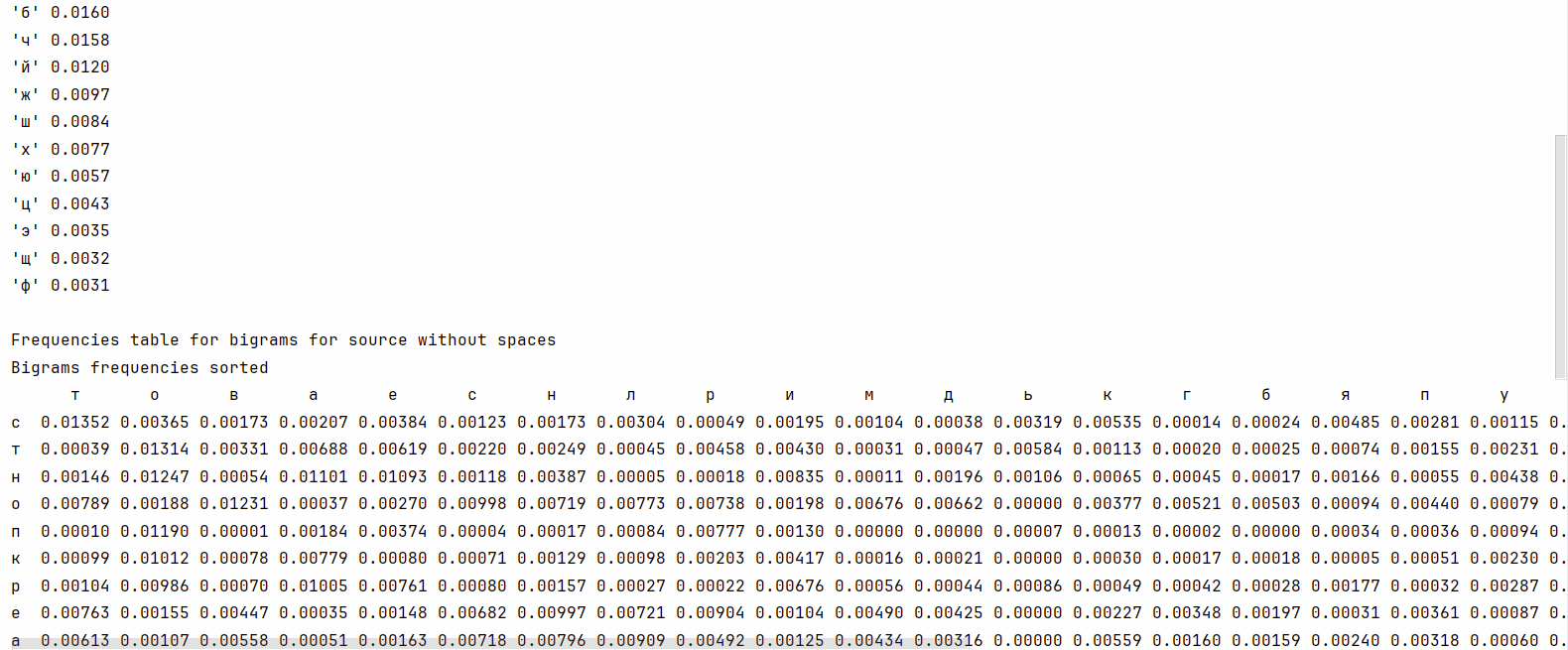
Основна програма створює текстові джерела з пропуском та без пропуску на базі великого тексту російською мовою та підраховує значення ентропії, а також наближене значення надлишковості російською мови.

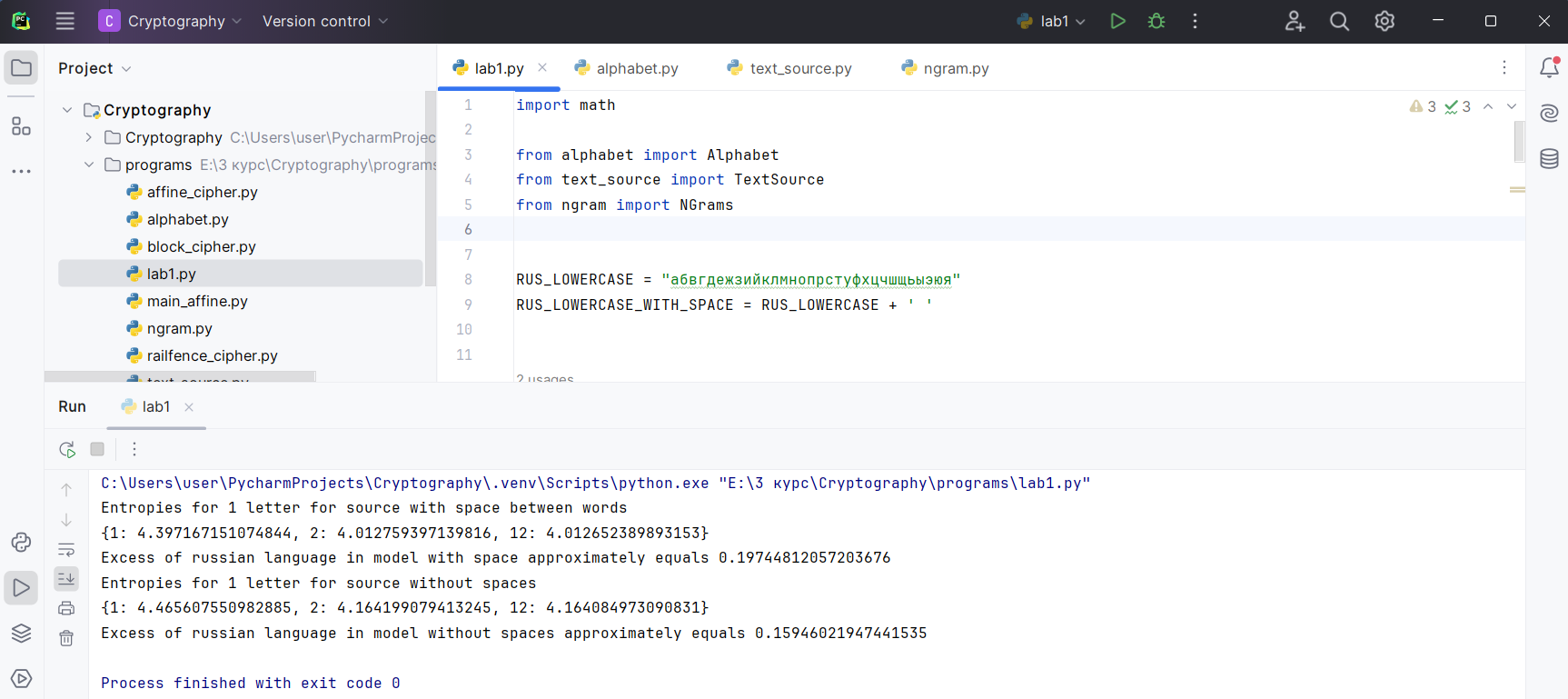
import math  
  
from alphabet import Alphabet  
from text\_source import TextSource  
from ngram import NGrams  
  
  
RUS\_LOWERCASE = "абвгдежзийклмнопрстуфхцчшщьыэюя"  
RUS\_LOWERCASE\_WITH\_SPACE = RUS\_LOWERCASE + ' '  
  
  
def get\_ngrams\_entropy(frequencies, length):  
 return -sum(frequencies[gram] \* (math.log2(frequencies[gram]))  
 for gram in frequencies if frequencies[gram] > 0) / length  
  
  
def calculate\_entropies(alphabet: Alphabet, source: TextSource, max\_length=2):  
 entropies = dict()  
 intersected = False  
 key = 1  
 for length in range(1, max\_length + 1):  
 ngrams = NGrams(alphabet, length, source)  
 ngrams.feed(intersected)  
 entropies[key] = get\_ngrams\_entropy(ngrams.get\_ngrams\_frequencies(), length)  
 key += 1  
  
 *# print(f"Errors: {ngrams.errors}")* intersected = True  
 key = 12  
 for length in range(2, max\_length + 1):  
 ngrams = NGrams(alphabet, length, source)  
 ngrams.feed(intersected)  
 entropies[key] = get\_ngrams\_entropy(ngrams.get\_ngrams\_frequencies(), length)  
 key += 1  
 return entropies  
  
  
def show\_monogram\_frequencies(alphabet: Alphabet, source: TextSource):  
 ngrams = NGrams(alphabet, 1, source)  
 ngrams.feed()  
 frequencies = ngrams.get\_ngrams\_frequencies(to\_sort=True)  
 print("Monograms frequencies sorted")  
 for letter, frequency in frequencies.items():  
 print(f"'{letter}' {frequency:6.4f}")  
  
  
def \_build\_axes\_lists(bigrams\_list):  
 rows = list()  
 cols = list()  
 while bigrams\_list:  
 first, second = bigrams\_list.pop(0) *# unpack bigram into 2 letters* if first not in rows:  
 rows.append(first)  
 if second not in cols:  
 cols.append(second)  
 return rows, cols  
  
  
def show\_bigram\_frequencies(alphabet: Alphabet, source: TextSource):  
 ngrams = NGrams(alphabet, 2, source)  
 ngrams.feed()  
 frequencies = ngrams.get\_ngrams\_frequencies(to\_sort=True)  
 bigrams\_list = list(frequencies.keys())  
 *# build lists of letters for rows and columns in sorted order* rows, cols = \_build\_axes\_lists(bigrams\_list)  
  
 print("Bigrams frequencies sorted")  
 print(" " + " ".join(cols))  
 for i, first in enumerate(rows):  
 f\_string = f"{first} "  
 for j, second in enumerate(cols):  
 f\_string += f" {frequencies[first + second]:7.5f}"  
 print(f\_string)  
  
  
PATH\_LAB1 = "..\\lab1\\"  
LONG\_FILE\_NAME = "rus\_text.txt"  
  
  
alphabet = Alphabet(RUS\_LOWERCASE\_WITH\_SPACE)  
source = TextSource(PATH\_LAB1 + LONG\_FILE\_NAME, alphabet, "to\_lower", "replace\_ru\_yo\_hard",  
 "replace\_delimeters", "leave\_one\_space")  
source.apply\_filter\_chain()  
  
entropies = calculate\_entropies(alphabet, source)  
print("Entropies for 1 letter for source with space between words")  
print(entropies)  
h0 = math.log2(alphabet.m)  
r = 1 - entropies[2] / h0  
print(f"Excess of russian language in model with space approximately equals {r}")  
  
alphabet = Alphabet(RUS\_LOWERCASE)  
source = TextSource(PATH\_LAB1 + LONG\_FILE\_NAME, alphabet, "to\_lower", "replace\_ru\_yo\_hard",  
 "delete\_delimeters", "delete\_spaces")  
source.apply\_filter\_chain()  
  
print("\nFrequencies table for monograms for source without spaces")  
show\_monogram\_frequencies(alphabet, source)  
  
print("\nFrequencies table for bigrams for source without spaces")  
show\_bigram\_frequencies(alphabet, source)  
  
print("\nEntropies for 1 letter for source without spaces")  
entropies = calculate\_entropies(alphabet, source)  
print(entropies)  
h0 = math.log2(alphabet.m)  
r = 1 - entropies[2] / h0  
print(f"Excess of russian language in model without spaces approximately equals {r}")

**6. Запуск програми**

Запустимо програму.



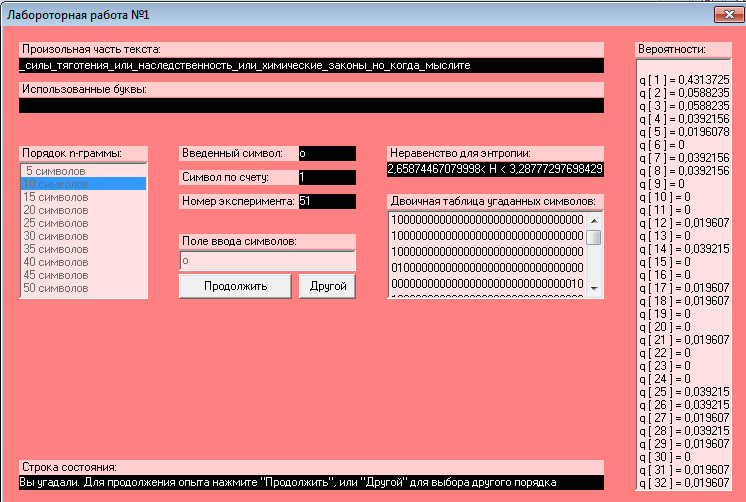


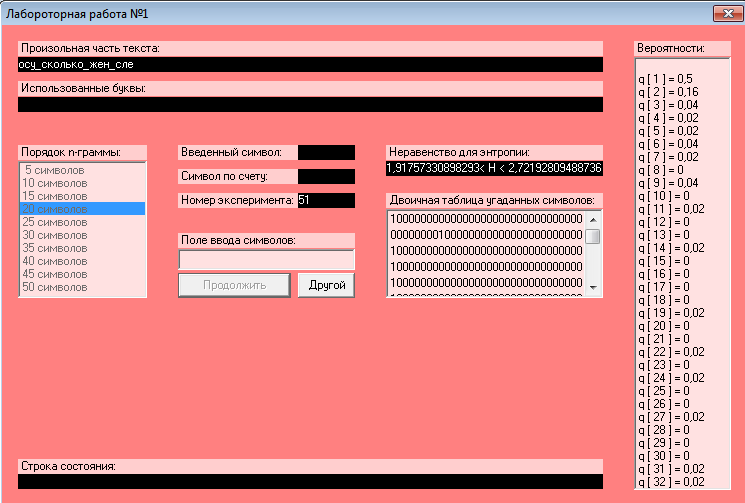


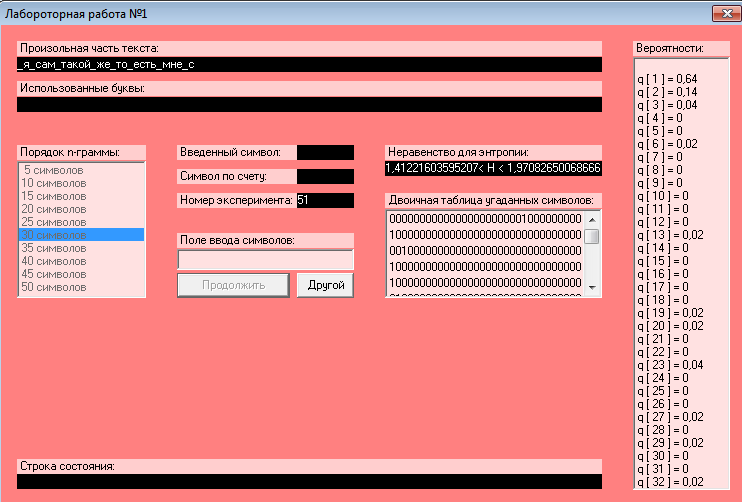
Бачимо пораховані значення ентропії алфавіту з пропуском та без пропуску. Надлишковість мови обчислюємо наближено з використанням H2. Також бачимо таблиці частот монограм та біграм.

**7. Використання** **Coolpinkprogram**

Coolpinkprogram дозволяє наближено обчислити умовну ентропію. Будемо обчислювати H(10) , H(20) , H(30). Для кожного обчислення проведемо по 50 експериментів. Стан програми після проведення 50 експериментів для 10, 20, 30 символів наведено нижче.







Значення умовної ентропії наближено задаються нерівностями:

2,6587<H(10)<3,2877

1,9175<H(20)<2,7219

1,4122<H(30)<1,9708

Бачимо, що із збільшенням кількості символів значення умовної ентропії на символ спадає.

Висновок

У цій роботі було проаналізовано текст, написаний російською мовою, обчислено кількості та частоти уніграм та біграм у цьому тексті. На підставі цих даних було наближено обчислено ентропію російської мови(Н1, Н2), а також надлишковість російської мови в різних моделях джерела.

Використовуючи готову програму, було обчислено умовну ентропію для 10, 20, 30 символів. Зі збільшенням кількості символів значення умовної ентропії на символ спадає.