**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

Виконала:

Студентка 3 курсу

Дрозд С.Ю

Перевірив:

Київ – 2021

**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:**

Визначити надлишковість російської мови в різних моделях джерела, вираховуючи за допомогою самостійно написанної програми частоту входження буквенних символів в текст(розмір від 1 мб, визначення ентропію Н1 та Н2 для букв та біграм( з букв, які перетинаюся, і не перетинаються) з урахуванням пробілів і без них, за допомогою спеціальної програми CoolPinkProgram оцінити Н10,Н20,Н30

**Хід роботи**

Для початку для визначення частоти входження символів в текст було написано програму на мові python3 у середовищі Jupyter notebook.

**Програма виконує наступні завдання:**

* Підготовка тексту(видалення лишніх символів, заміна великих букв на малі, тощо)
* Підрахунок кількості входжень кожного символі в текст та відповідно визначення частоти
* Пошук біграм, що перетинаються та не перетинаються, та визначення частоту їх входжень у текст
* Обчислення ентропії Н1с та Н2 для тексту з пробілами та без
* Збередження результатів у датафреймах та таблицях ексель

**Код програми**

Весь код програми поданий у фалі code.py

#text prepare

import regex

import collections

import unicodedata

text = open("/home/sofi/1.txt").read()

text = text.lower()

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

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

text = ' '.join(text.split())

import re

#without space

newstring = ''.join(c for c in text if unicodedata.category(c).startswith('L'))

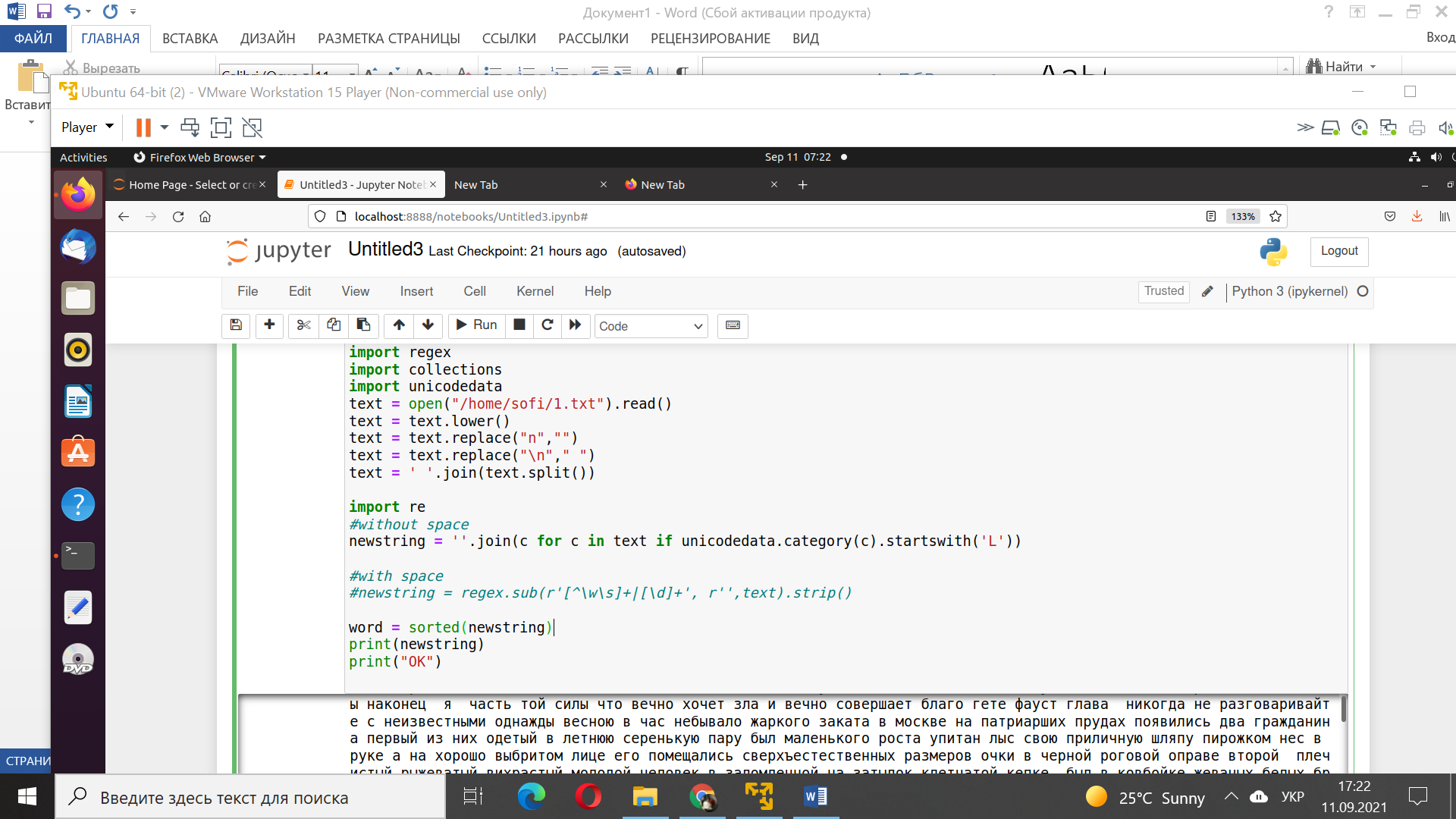
#with space

#newstring = regex.sub(r'[^\w\s]+|[\d]+', r'',text).strip()

word = sorted(newstring)

print(newstring)

print("OK")



#Count

alp = []

for i in range(0,len(word)):

alp.append(word[i])

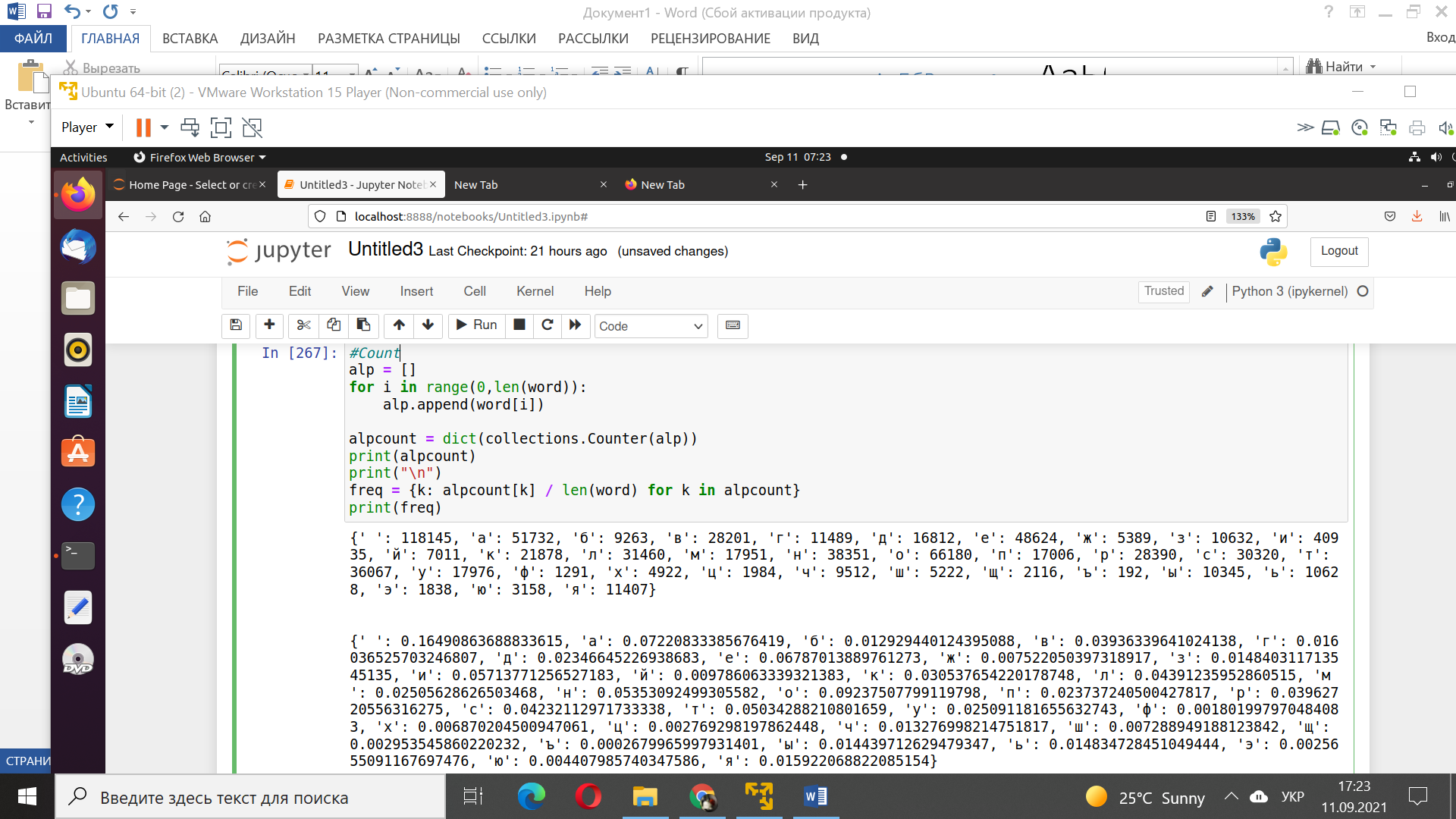
alpcount = dict(collections.Counter(alp))

print(alpcount)

print("\n")

freq = {k: alpcount[k] / len(word) for k in alpcount}

print(freq)



#print count

sot = sorted(alpcount, key=lambda x : alpcount[x],reverse=1)

temp = []

for i in range(0,len(sot)):

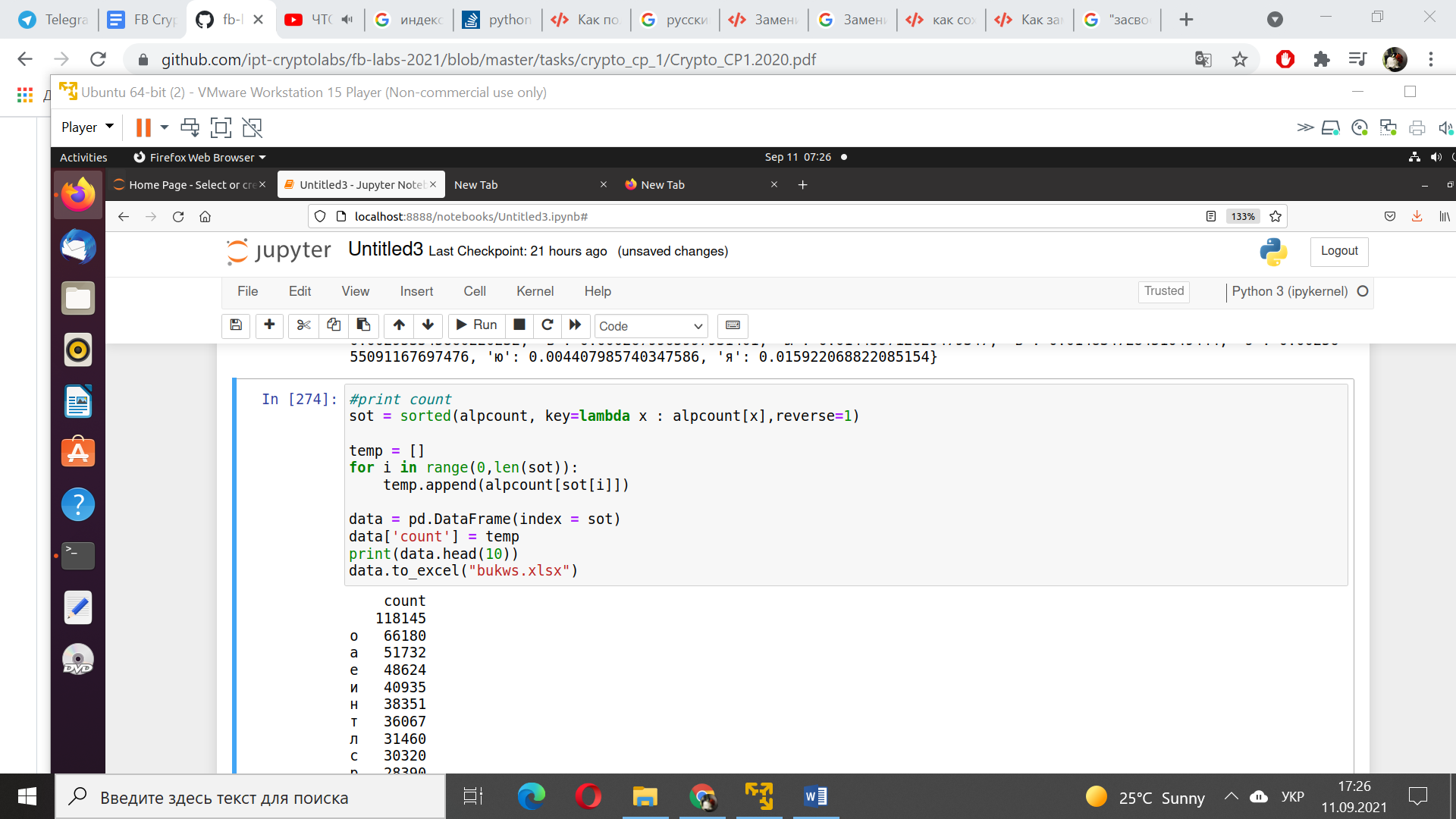
temp.append(alpcount[sot[i]])

data = pd.DataFrame(index = sot)

data['count'] = temp

print(data.head(10))

data.to\_excel("bukws.xlsx")



#print freq

fr= sorted(freq, key=lambda x : alpcount[x],reverse=1)

temp = []

for i in range(0,len(sot)):

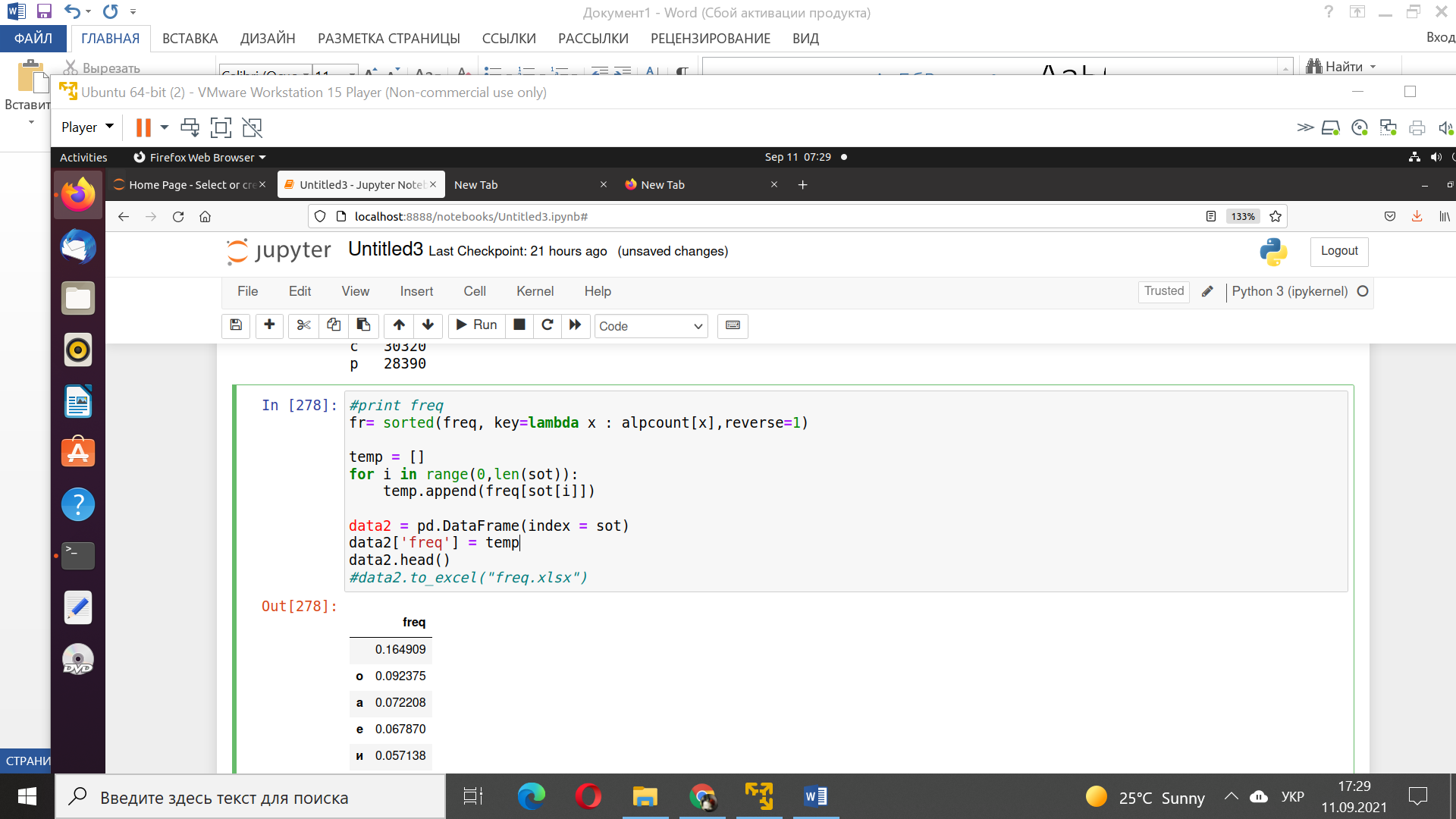
temp.append(freq[sot[i]])

data2 = pd.DataFrame(index = sot)

data2['freq'] = temp

data2.head()

data2.to\_excel("freq.xlsx")



#Bigram

text = newstring

bigram = []

bigramtwo = []

for j in range(0, len(text)-1):

bigram.append(text[j]+text[j+1])

for j in range(0, len(text)-2,2):

bigramtwo.append(text[j]+text[j+1])

from collections import Counter

bigramcount = dict(Counter(bigram))

freqbigram = {k: bigramcount[k] / len(bigram) for k in bigramcount}

bigramtwocount = dict(Counter(bigramtwo))

freqbigramtwo = {k: bigramtwocount[k] / len(bigramtwo) for k in bigramtwocount}

#H1

import math

h1 = []

for f in freq.values():

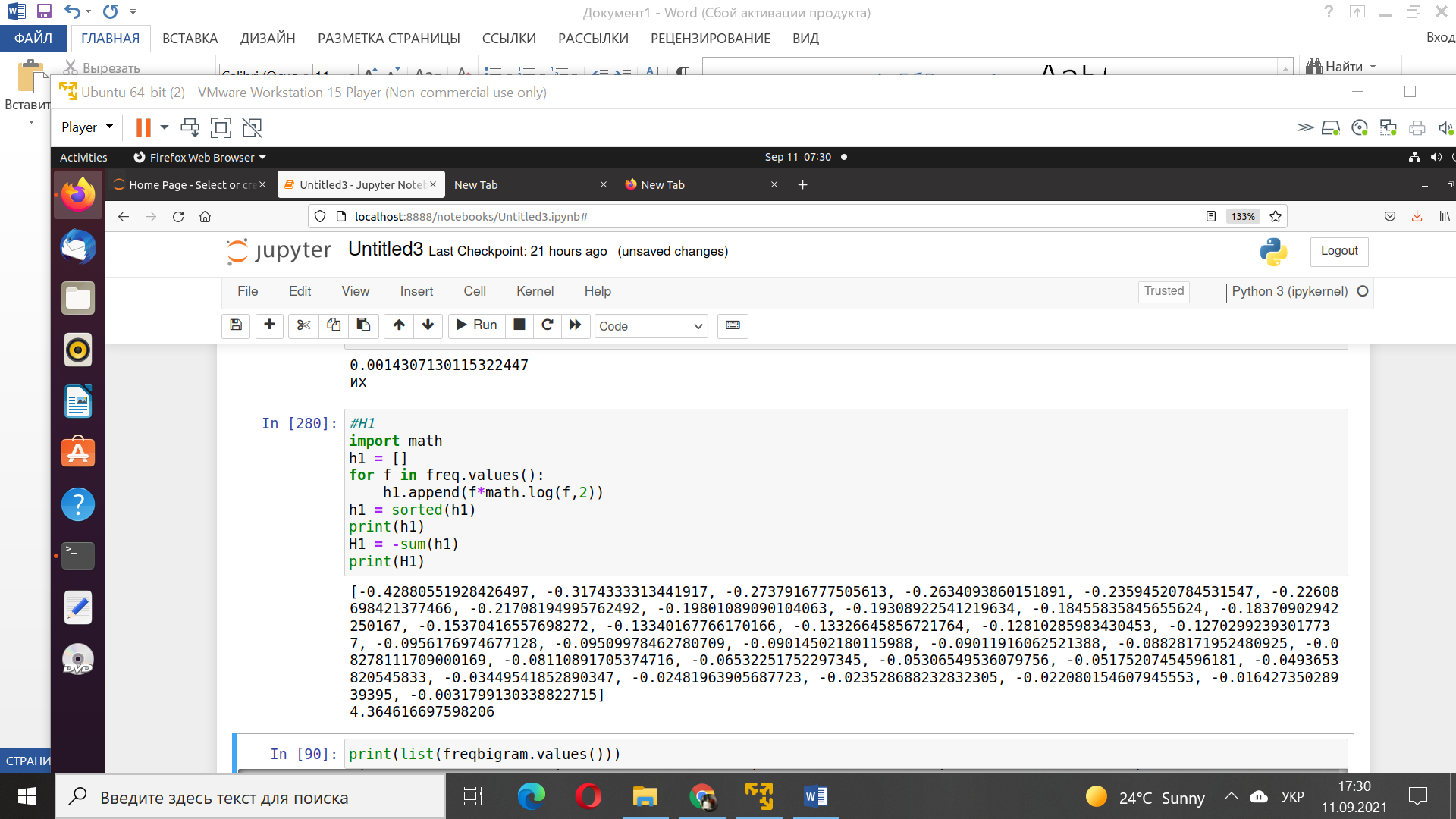
h1.append(f\*math.log(f,2))

h1 = sorted(h1)

print(h1)

H1 = -sum(h1)

print(H1)



#H2

def H(bigram,freqbigram):

hn = []

for f in freqbigram.values():

hn.append(f\*math.log(f,2))

hn = sorted(hn)

H = -sum(hn)/2

return H

H2 = H(bigram,freqbigram)

H22 = H(bigramtwo,freqbigramtwo)

print(H2)

print(H22)

Alp = ['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']

#Alp = [' ', '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']

df = pd.DataFrame(index = Alp, columns=Alp)

bg = []

for i in Alp:

for j in Alp:

bg.append(i+j)

n = 0

for i in range(0,len(Alp)):

df[Alp[i]] = bg[n:len(Alp)+n]

n = len(Alp)+n

df = df.T

import numpy as np

for b in list(freqbigram.keys()):

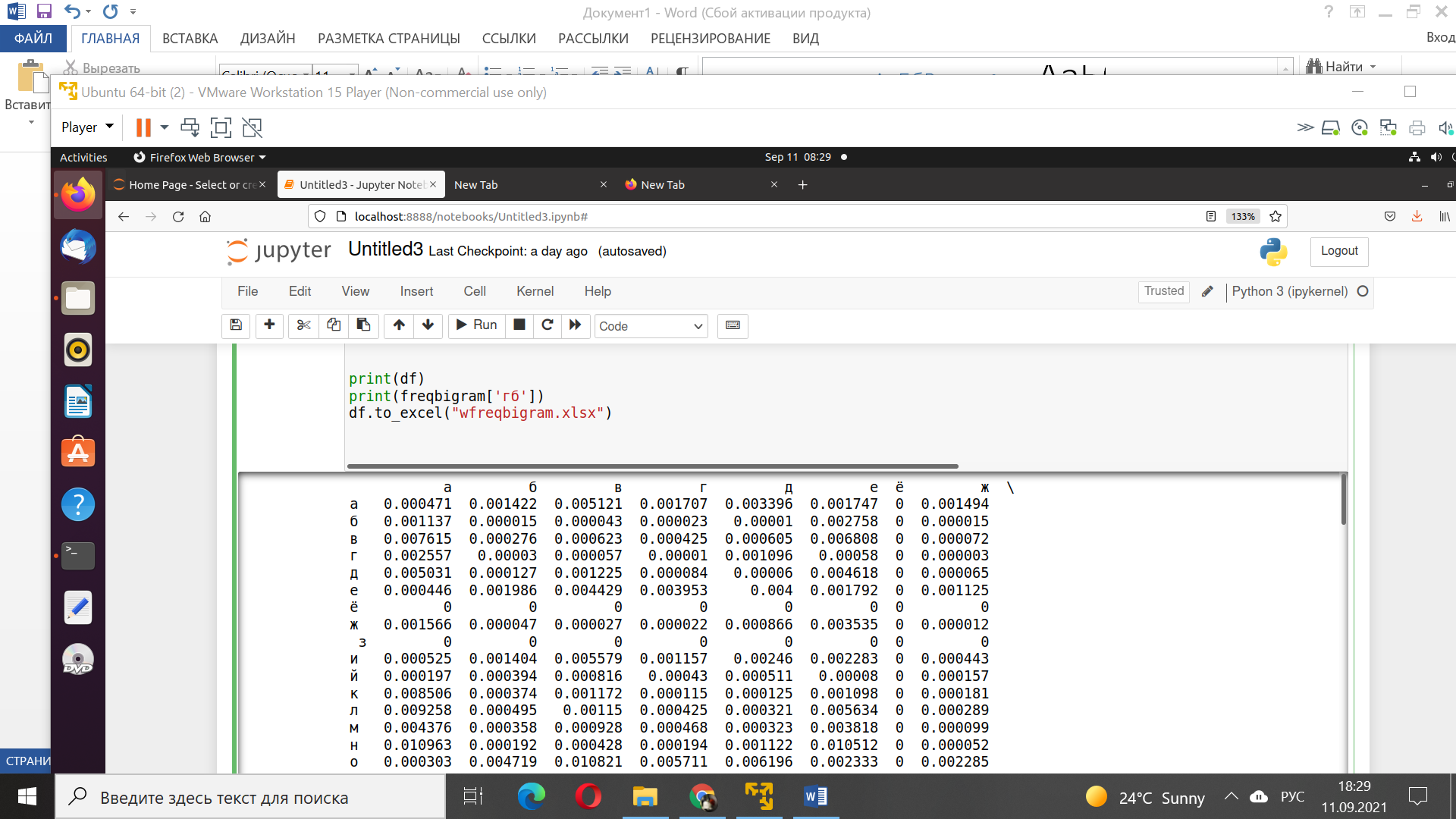
a,c = np.where(df == b)

df.iloc[a,c] = freqbigram[b]

for b in bg:

a,c = np.where(df == b)

df.iloc[a,c] = 0



df = pd.DataFrame(index = Alp, columns=Alp)

bg = []

for i in Alp:

for j in Alp:

bg.append(i+j)

n = 0

for i in range(0,len(Alp)):

df[Alp[i]] = bg[n:len(Alp)+n]

n = len(Alp)+n

df = df.T

for b in list(freqbigramtwo.keys()):

a,c = np.where(df == b)

df.iloc[a,c] = freqbigramtwo[b]

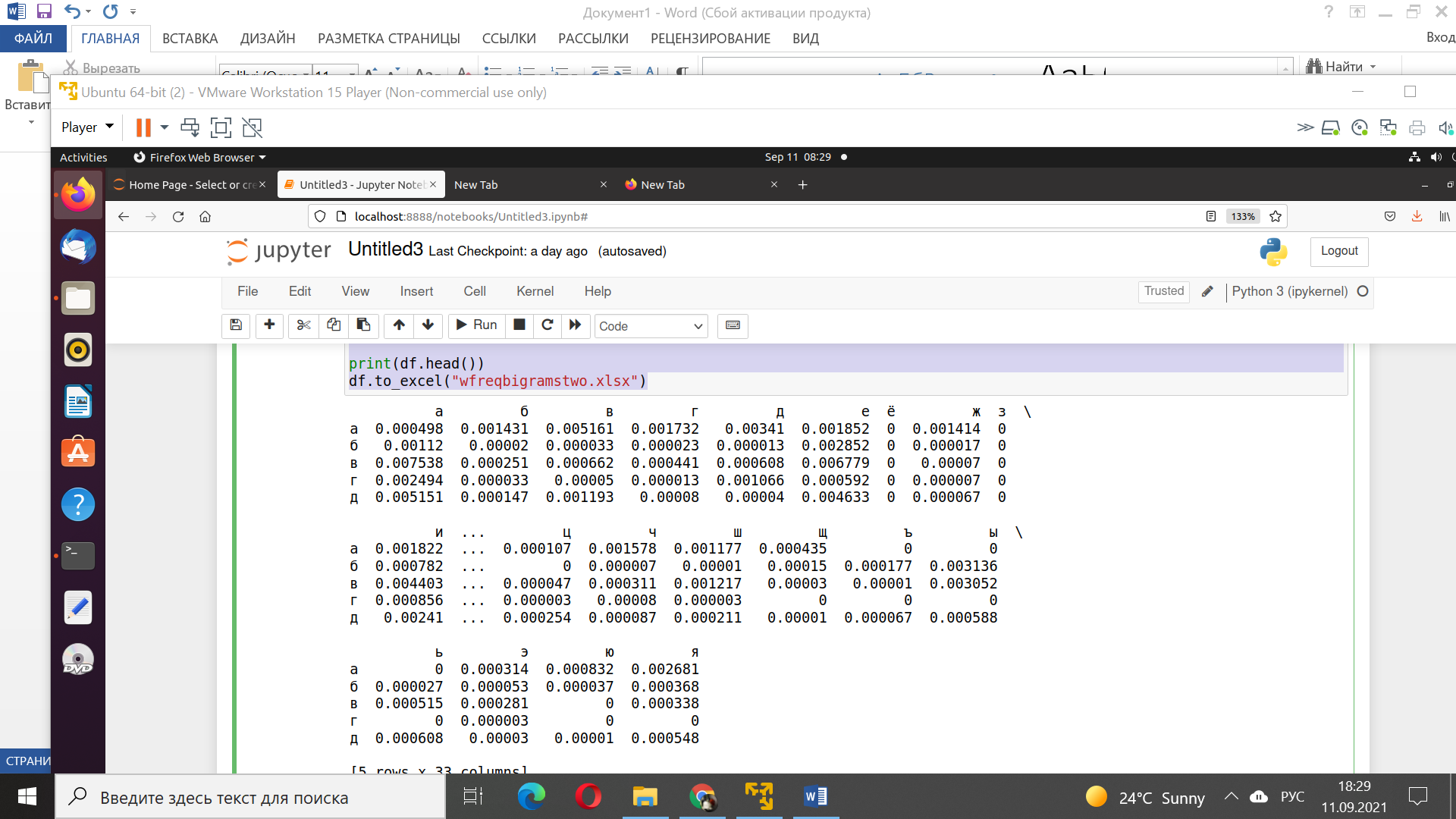
for b in bg:

a,c = np.where(df == b)

df.iloc[a,c] = 0

print(df.head())

df.to\_excel("wfreqbigramstwo.xlsx")



# **Результати**

**Частоти**

Таблиці частот наведені у файлах:

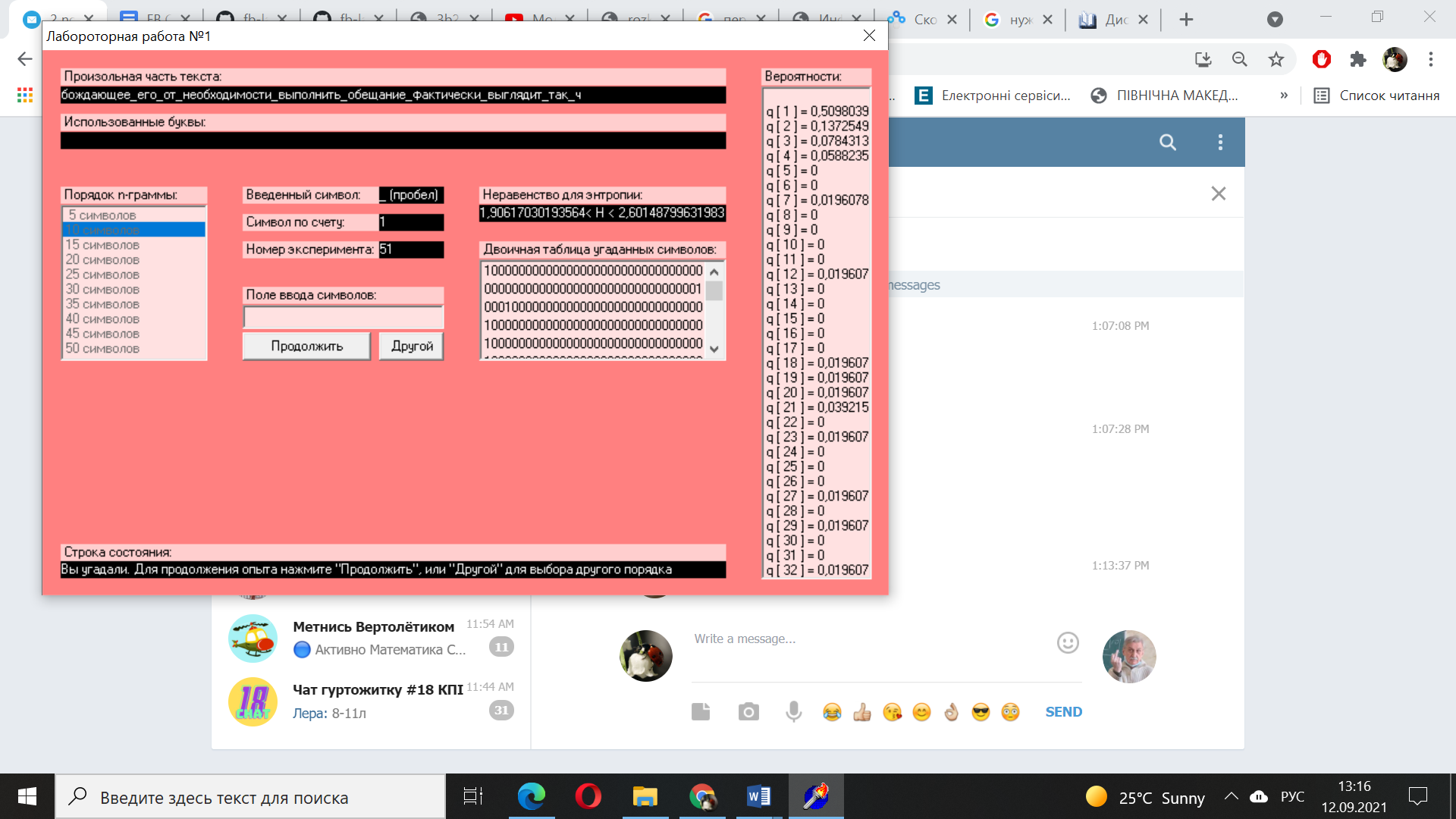
|  |  |  |
| --- | --- | --- |
|  | Текст з пробілами | Текст без пробілів |
| Букви | freq.xlsx | Freqs.xlsx |
| Біграми | freqbigram.xlsx | wfreqbigram.xlsx |
| Біграми з кроков 2 | freqbigramtwo.xlsx | wfreqbigramtwo.xlsx |

**Ентропія**

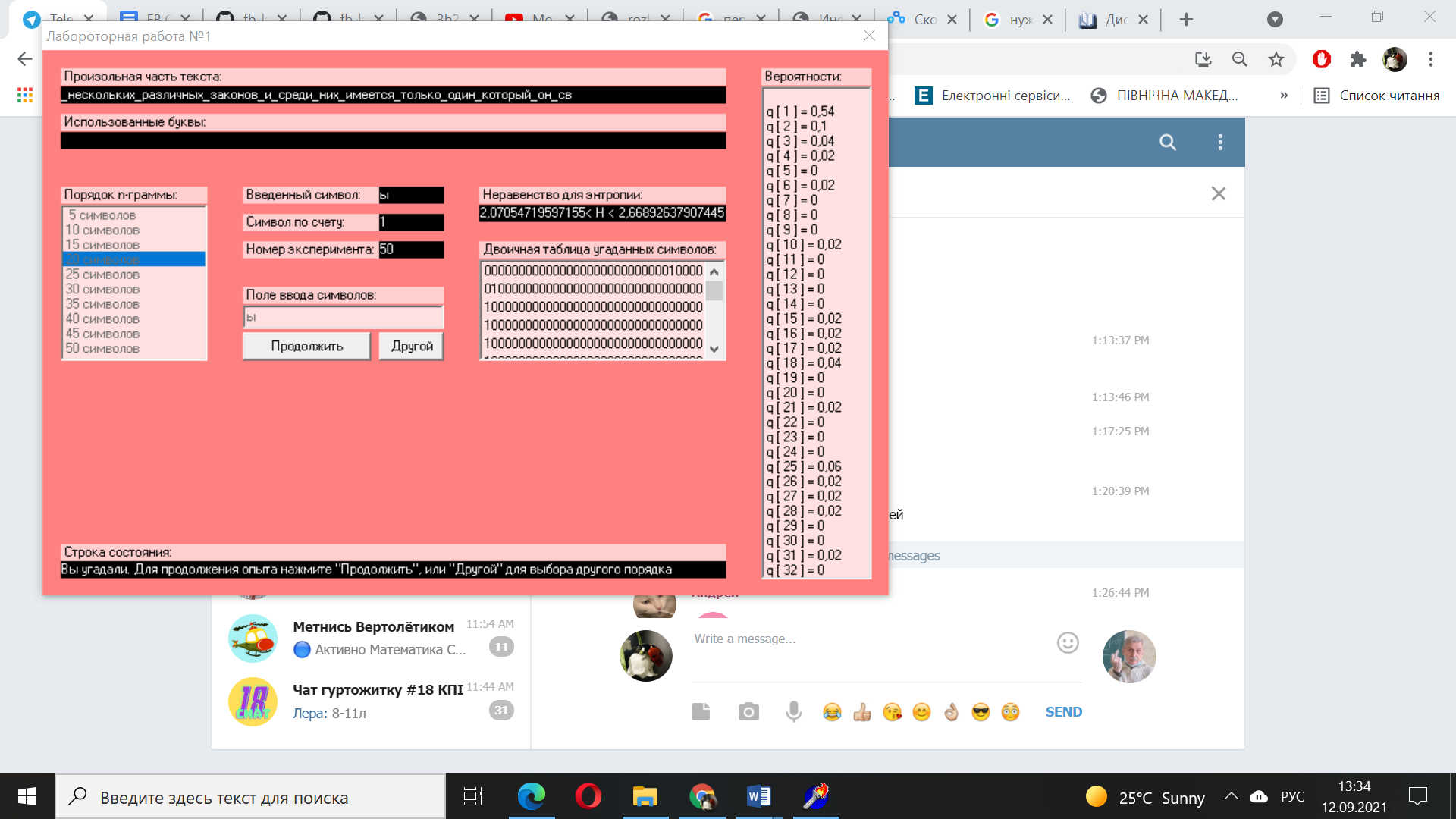
|  |  |  |
| --- | --- | --- |
|  | Текст з пробілами | Текст без пробілів |
| Н1 | 4.364616697598206 | 4.453036585414568 |
| Н2 | 3.989200044367577 | 4.147762969657709 |
| Н2(біграми з кроком 2) | 3.9887080981913745 | 4.147556479282259 |
|  |  |  |

**Робота з CoolPinkProgram**

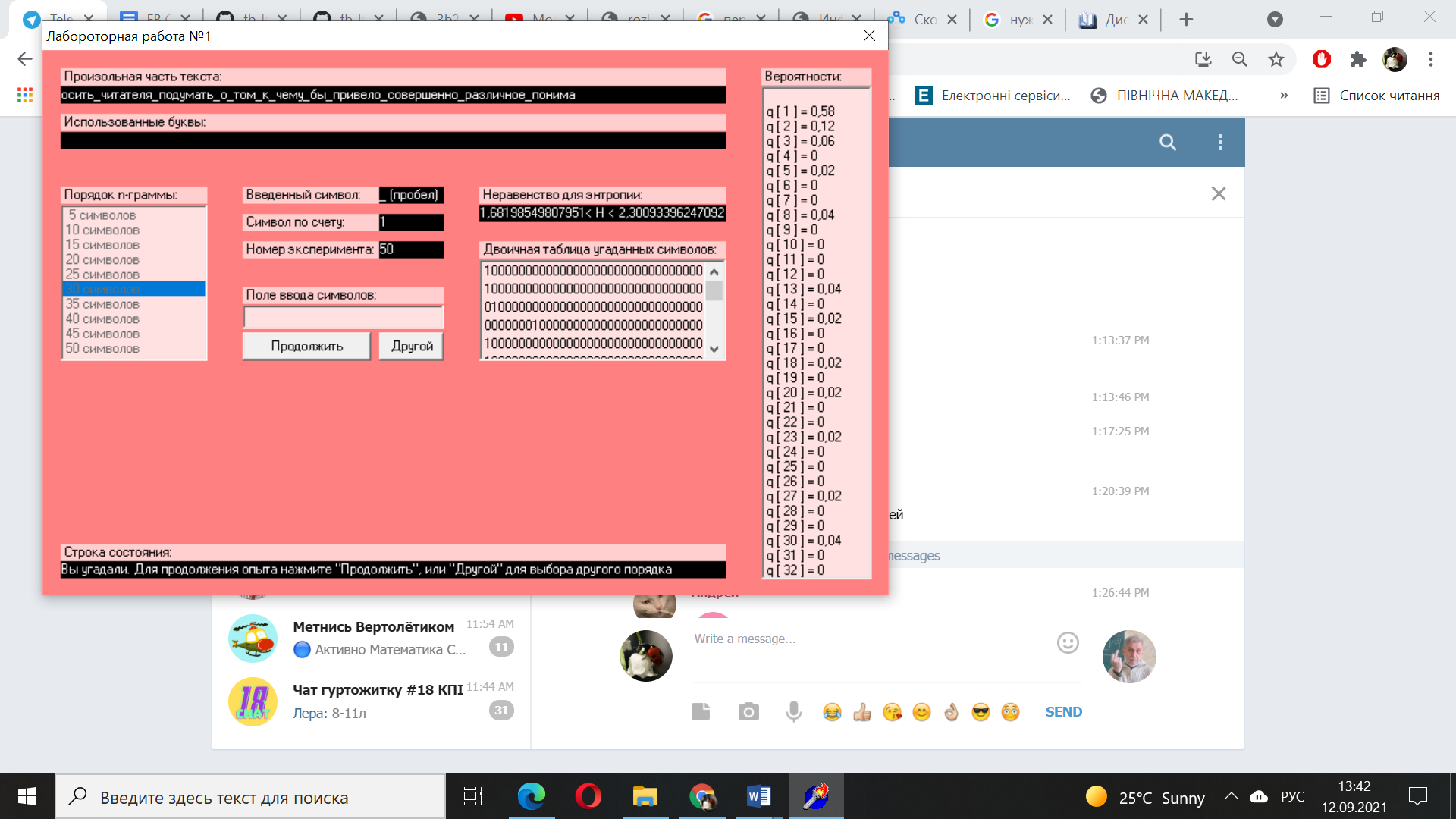
1.906<H(10) <2.601



2.07<H(20) <2.66



1.68<H(30) <2.03



**Оцінка надлишковості мови:**

|  |  |  |
| --- | --- | --- |
|  | R(Текст з пробілами) | R(Текст без пробілів) |
| Н1 | **0.134** | **0.117** |
| Н2 | **0.20911** | **0.17768** |
| Н2(з кроком 2) | **0.2092** | **0.17772** |

**Висновки:**

Під час виконання даної роботи я навчилася вимірювати частоти повторювання символів в тексті, визначати ентропію та надлишковість мови. Я побачила, що коли алфавіт містить пробіл, то ентропія менша. Також я побачила, що зі зростанням порядку n-грами Н(n) спадає. Я оцінила надлишковість російської мови і побачила, що вона не перевищує 0.21