Звіт по комп’ютерному практикуму №1

Виконували:

Ракович Дарина ФБ-73

Пекарчук Данило ФБ-74

**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 ентропіх на символ джерела

**Постановка задачі**:

Основною метою було написання програми для підрахунку чатот букв і частот біграм в тексті, а також підрахунку H1 та Н2 за безпосереднім означенням. Підрахувати частоти букв та біграм, а також значення H1 та Н2 на тому ж тексті, в якому вилучено всі пробіли.

**Порядок виконання роботи**

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 H10 , H20 , H30 .

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

**Код програми:**

import sys

import math

import re

from collections import defaultdict

def get\_file\_lines(fname):

with open(fname) as f:

return f.readlines()

def remove\_not\_alpha(lines, need\_space):

return\_list = list()

for line in lines:

re\_check = '[^а-яА-Я{}]+'.format(' ' if need\_space else '')

fixed\_line = re.sub(re\_check, '', line.lower())

return\_list.append(fixed\_line)

return return\_list

def get\_ngram\_dict(lines, ngram\_num=2, step=2):

ngram\_counter = 0

ngram\_dict = defaultdict(int)

for line in lines:

for char\_counter in range(0, len(line), step):

ngram = line[char\_counter:char\_counter+ngram\_num]

if len(ngram) != ngram\_num:

continue

ngram\_dict[ngram] += 1

ngram\_counter += 1

return\_dict = dict()

for elem, amount\_in\_text in ngram\_dict.items():

return\_dict[elem] = amount\_in\_text / ngram\_counter

return return\_dict

def get\_entropy(ngram\_dict, num):

result = 0

grouped\_dict = dict()

for item, value in ngram\_dict.items():

try:

grouped\_dict[value].append(item)

except KeyError:

grouped\_dict[value] = [item]

for number, elements in grouped\_dict.items():

result += (number \* math.log(number, 2)) \* len(elements)

return result \* (-1/num)

def get\_redundancy(entropy):

return 1-(entropy/(math.log(33, 2)))

def write\_to\_file(

ngram\_dict,

entropy\_val,

redundancy\_val,

need\_space,

num,

fname

):

with open('out.txt', 'w') as f:

f.write(f'Got text from {fname}...filtering out non-alphas{" and spaces" if not need\_space else ""}...looking for {num}-grams\nResult:\n')

for item, value in ngram\_dict.items():

f.write(item + ' -> ' + str(value) + '\n')

f.write("Entropy: " + str(entropy\_val) + '\n')

f.write("Redundancy: " + str(redundancy\_val) + '\n')

def main():

num = int(sys.argv[1]) if len(sys.argv) >= 2 else 2

need\_space = eval(sys.argv[2]) if len(sys.argv) >= 3 else False

fname = sys.argv[3] if len(sys.argv) >= 4 else 'TEXT'

step = int(sys.argv[4]) if len(sys.argv) >= 5 else num

lines = get\_file\_lines(fname)

filtered\_lines = remove\_not\_alpha(

lines,

need\_space

)

ngram\_dict = get\_ngram\_dict(

filtered\_lines,

num,

step=step

)

entropy\_val = get\_entropy(

ngram\_dict,

num

)

redundancy\_val = get\_redundancy(entropy\_val)

for item, value in ngram\_dict.items():

print(item, ' -> ', value)

print("entropy: ", entropy\_val)

print("redundancy: ", redundancy\_val)

write\_to\_file(

ngram\_dict,

entropy\_val,

redundancy\_val,

need\_space,

num,

fname

)

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Хід роботи**: В ході роботи виникли деякі труднощі з структурою даних та як найшвидше та найкраще їх відфільтрувати. Для фільтрації ми використали regex, а для структури даних найкраще підходить словник через те що всі операції в ньому це O(1).

Потім запустили CoolPinkProgram, та намагались підібрати наступну букву для різних комбінацій, результати можна побачити на картинках. Результати виконання нашої програми  
  
1-gram without spaces

entropy:  4.459208086129053

redundancy:  0.11600719915671953

1-gram with spaces

entropy:  4.353445819478285

redundancy:  0.13697349642617596

2-gram without spaces and without overlaying

entropy:  4.141179852289086

redundancy:  0.17905307271752957

2-grams with spaces without overlaying

entropy:  3.9717881434490683

redundancy:  0.21263326190020204

2-gram with spaces with overlaying

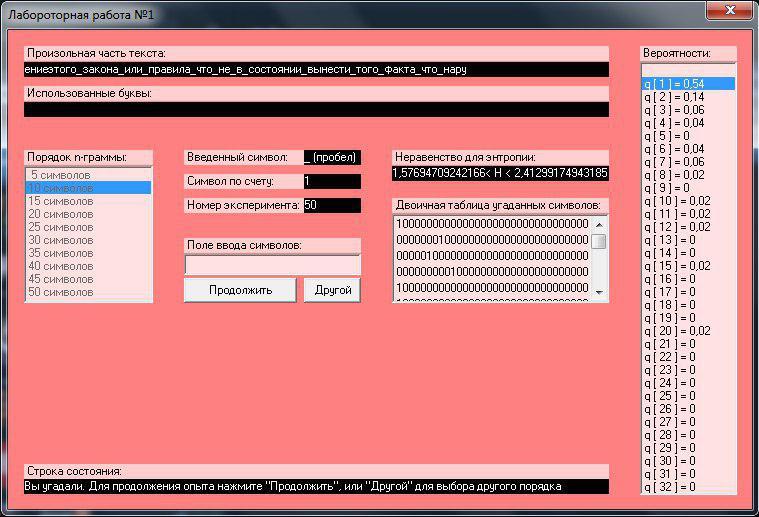
entropy:  3.972171958395299

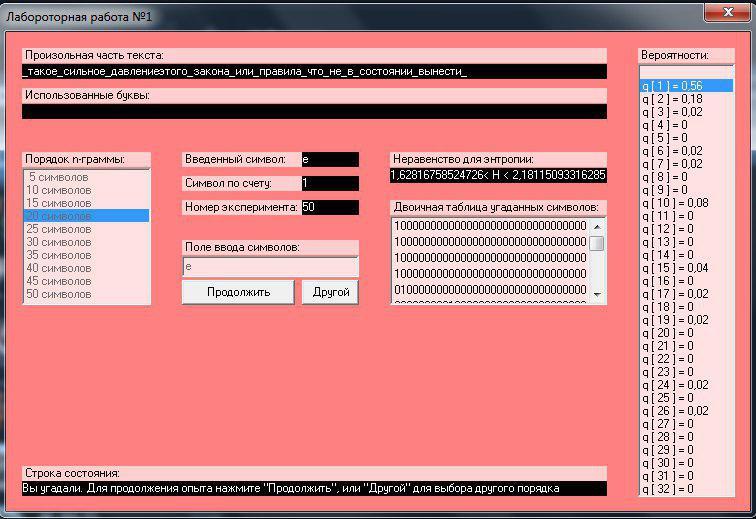
redundancy:  0.21255717447777844

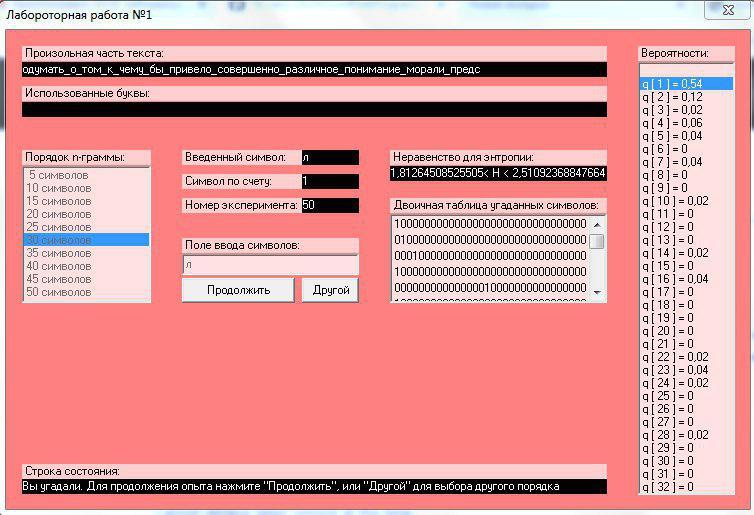
2-grams without spaces with overlaying

entropy:  4.138469481393185

redundancy:  0.1795903762730745







**Висновки**: В цьому комп’ютерному практикумі ми засвоїли поняття ентропії на символ джерела та його надлишковості, вивчили та порівняли різниі моделі джерела відкритого тексту для наближеного визначення ентропії, набули практичних навичок щодо оцінки ентропії на символ джерела.