

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 H_1 та H_2 за безпосереднім означенням. Підрахувати частоти букв та біграм, а також значення H_1 та H_2 на довільно обраному тексті російською мовою достатньої довжини (щонайменше 1Мб), де імовірності замінити відповідними частотами. Також одержати значення H_1 та H_2 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 = "АБВГГДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ"  
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_alphabet = 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 назовемо 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_DELIMITERS = '.,:;()"'0123456789+~_№#&!/?|\\$%<>*-„“-...=[ ] { } '

class TextSource:

```

Луняка Артем ФБ-24

```

    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_DELIMITERS):
        delimiters = set(delim)
        filtered = [c if c not in delimiters else "" for c in source]
        return ''.join(filtered)

    @staticmethod
    def replace_delimeters_filter(source: str, delim=STANDARD_DELIMITERS):
        delimiters = set(delim)
        filtered = [c if c not in delimiters else ' ' for c in source]
        return ''.join(filtered)

```

```

    @staticmethod
    def replace_ru_yo_hard_filter(source: str, to_replace= (('ё', 'e'), ('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 виконає 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 = "абвгдежзийклмнопрстуфхцчшщъыэюя"
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_delimiters", "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_delimiters", "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 программы

Запустимо програму.

```
E:\3_curs\Cryptography\programs\.venv\Scripts\python.exe E:\3_curs\Cryptography\programs\lab1_reworked.py
Entropies for 1 letter for source with space between words
{1: 4.397167151074844, 2: 4.012759397139816, 12: 4.012652389893153}
Excess of russian language in model with space approximately equals 0.19744812057203676
```

Frequencies table for monograms for source without spaces

Monograms frequencies sorted

'о' 0.1089

'е' 0.0828

'а' 0.0815

'и' 0.0667

'н' 0.0666

'т' 0.0592

'с' 0.0544

'р' 0.0500

'л' 0.0488

'в' 0.0474

'к' 0.0342

'п' 0.0304

'д' 0.0301

'у' 0.0299

'м' 0.0288

'я' 0.0198

'ь' 0.0183

'г' 0.0179

'ы' 0.0177

'з' 0.0173

'б' 0.0160

'ч' 0.0158

'й' 0.0120

'ж' 0.0097

'ш' 0.0084

'х' 0.0077

'ю' 0.0057

'ц' 0.0043

'э' 0.0035

'щ' 0.0032

'ф' 0.0031

Frequencies table for bigrams for source without spaces

Bigrams frequencies sorted

	т	о	в	а	е	с	н	л	р	и	м	д	ь	к	г	б	я	п	у
с	0.01352	0.00365	0.00173	0.00207	0.00384	0.00123	0.00173	0.00304	0.00049	0.00195	0.00104	0.00038	0.00319	0.00535	0.00014	0.00024	0.00485	0.00281	0.00115
т	0.00039	0.01314	0.00331	0.00688	0.00619	0.00220	0.00249	0.00045	0.00458	0.00430	0.00031	0.00047	0.00584	0.00113	0.00020	0.00025	0.00074	0.00155	0.00231
о	0.00146	0.01247	0.00054	0.01101	0.01093	0.00118	0.00387	0.00005	0.00018	0.00835	0.00011	0.00196	0.00106	0.00065	0.00045	0.00017	0.00166	0.00055	0.00438
в	0.00789	0.00188	0.01231	0.00037	0.00270	0.00998	0.00719	0.00773	0.00738	0.00198	0.00676	0.00662	0.00000	0.00377	0.00521	0.00503	0.00094	0.00440	0.00079
а	0.00010	0.01190	0.00001	0.00184	0.00374	0.00004	0.00017	0.00084	0.00777	0.00130	0.00000	0.00000	0.00007	0.00013	0.00002	0.00000	0.00034	0.00036	0.00094
е	0.00099	0.01012	0.00078	0.00779	0.00080	0.00071	0.00129	0.00098	0.00203	0.00417	0.00016	0.00021	0.00000	0.00030	0.00017	0.00018	0.00005	0.00051	0.00230
с	0.00104	0.00986	0.00070	0.01005	0.00761	0.00080	0.00157	0.00027	0.00022	0.00676	0.00056	0.00044	0.00086	0.00049	0.00042	0.00028	0.00177	0.00032	0.00287
н	0.00763	0.00155	0.00447	0.00035	0.00148	0.00682	0.00997	0.00721	0.00904	0.00104	0.00490	0.00425	0.00000	0.00227	0.00348	0.00197	0.00031	0.00361	0.00087
л	0.00613	0.00107	0.00558	0.00051	0.00163	0.00718	0.00796	0.00909	0.00492	0.00125	0.00434	0.00316	0.00000	0.00559	0.00160	0.00159	0.00240	0.00318	0.00060

$$1,4122 < H^{(30)} < 1,9708$$

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 (H_1 , H_2), 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.