## Your Title Here

### 1 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 ентропії на символ джерела.

## 2 Хід роботи

Для виконання поставленого завдання, після короткого аналізу, я вирішив розбити його на три різних частини:

- робота з файлами та вхідним текстом
- робота з літерами
- робота з біграмами

#### 2.1 Робота з файлами та вхідним текстом

He буду вдаватись в деталі реалізації, так як це не є настільки важливим в даній роботі, лише хочу зазначити, що вхідний текст має назву boloto.txt, цей же текст, але вже опрацьований препроцесором має назву boloto\_processed.txt, а вже оброблений файл та ще й без пробілів має назву boloto\_without\_spaces.txt.

### 2.2 Робота з літерами

В даній секції ми розв'язуємо декілька задач:

• підрахунок загальної к-ті літер у тексті

```
fn letters_count(letter_frequencies: &HashMap < char, i64 >) -> i64
      {
        let mut count = 0;
        for (_key, _value) in letter_frequencies {
            count += _value;
        }
        count
}
```

• підрахунок к-ті кожної літери

```
fn get_letter_frequency(text: &str) -> HashMap < char, i64 > {
    let mut frequencies: HashMap < char, i64 > = HashMap::new();

for c in text.chars() {
        *frequencies.entry(c).or_insert(0) += 1;
}

frequencies
}
```

• підрахунок ймовірності зустірти кожну окрему літеру

• вивід знайдених значень

```
fn print_letters_probabilities(probabilities: &HashMap < char, f64
     >) {
     let mut sorted_probabilities: Vec<(&char, &f64)> =
     probabilities.iter().collect();
      sorted_probabilities.sort_by(|a, b| b.1.partial_cmp(a.1).
     unwrap());
      for (&letter, &probability) in sorted_probabilities {
          println!("{}: {}", letter, probability);
5
6
      println!();
7
8 }
10 fn print_letter_frequencies(letter_frequencies: &HashMap<char,</pre>
     i64>) {
     let mut sorted_frequencies: Vec<(&char, &i64)> =
11
     letter_frequencies.iter().collect();
      sorted_frequencies.sort_by_key(|&(_, frequency)| *frequency);
      for (&letter, &frequency) in sorted_frequencies.iter().rev()
13
     {
          println!("{}: {}", letter, frequency);
14
      println!();
16
17 }
```

#### 2.3 Робота з біграмами

Аналогічні задачі доводиться розв'язувати і у випадку з біграмами, правда реалізація буде досить сильно відрізнятись.

• підрахунок загальної к-ті біграм у тексті

```
fn bigram_count(bigram_frequencies: &HashMap < String, i64 >) -> i64
      {
        let mut count = 0;
        for (_key, _value) in bigram_frequencies {
            count += _value;
        }
        count
7 }
```

• підрахунок к-ті кожної з біграм

```
1 fn get_bigram_frequency(text: &str) -> HashMap<String, i64> {
      let mut frequencies: HashMap < String, i64 > = HashMap::new();
      let mut chars = text.chars().peekable();
      while let (Some(curr), Some(&next)) = (chars.next(), chars.
     peek()) {
          if curr.is_alphabetic() && next.is_alphabetic() {
6
              let bigram = format!("{}{}", curr.to_lowercase(),
     next.to_lowercase());
              *frequencies.entry(bigram).or_insert(0) += 1;
          } else if curr.is_alphabetic() && next.is_whitespace() {
9
              let bigram = format!("{} ", curr.to_lowercase());
10
              *frequencies.entry(bigram).or_insert(0) += 1;
          } else if curr.is_whitespace() && next.is_alphabetic() {
              let bigram = format!(" {}", next.to_lowercase());
13
              *frequencies.entry(bigram).or_insert(0) += 1;
      }
      frequencies
17
18 }
```

• підрахунок ймовірності зустірти кожну окрему біграму

```
fn count_bigram_probabilities(bigram_frequencies: &HashMap < String
    , i64 > ) -> HashMap < String, f64 > {
    let mut probabilities: HashMap < String, f64 > = HashMap::new();
    let number_of_bigrams = bigram_count(bigram_frequencies) as
    f64;

for (_key, _value) in bigram_frequencies {
        probabilities.insert(_key.clone(), (*_value as f64) /
        number_of_bigrams);
    }

probabilities
```

• вивід знайдених значень

# 3 Результати роботи

## 3.1 Текст з пробілами

## 3.1.1 Робота з літерами

Літера	Кількість	Ймовірність
, ,	248583	0.0149
a	115586	0.06929
б	23656	0.01418
В	65348	0.03918
Г	23164	0.01389
Д	43783	0.02625
e	113939	0.06831
ë	2136	0.00128
Ж	12555	0.00753
3	21398	0.01283
И	107358	0.06436
Й	17005	0.01019
K	47901	0.02872
Л	56102	0.03363
M	46423	0.02783
Н	92312	0.05534
О	158172	0.09482
П	41612	0.02495
p	67768	0.04063
c	77815	0.04665
Т	96940	0.05811
у	37574	0.02253
ф	4406	0.00264
X	11742	0.00704
Ц	6460	0.00387
Ч	20695	0.01241
Ш	9662	0.00579
Щ	4685	0.00281
Ы	25661	0.01538
Ь	24874	0.01491
Э	6029	0.00361
Ю	10427	0.00625
Я	25800	0.01547
Ъ	513	0.00031

### 3.1.2 Робота з біграмами



Рис. 1: к-ть біграм

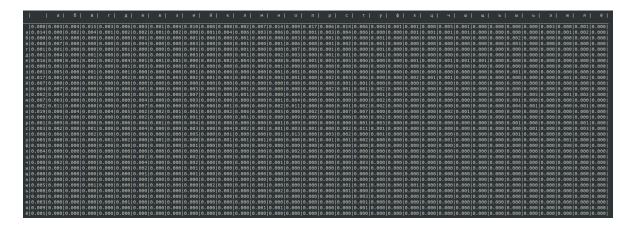


Рис. 2: ймовірність появи біграм

## 3.2 Текст без пробілів

### 3.2.1 Робота з літерами

Літера	Кількість	Ймовірність
a	115586	0.08143
б	23656	0.01667
В	65348	0.04604
Г	23164	0.01632
д	43783	0.03084
e	113939	0.08027
ë	2136	0.00150
Ж	12555	0.00884
3	21398	0.01507
И	107358	0.07563
й	17005	0.012
K	47901	0.03374
Л	56102	0.03952
M	46423	0.0327
Н	92312	0.06503
О	158172	0.11143
П	41612	0.02931
p	67768	0.04774
c	77815	0.05482
Т	96940	0.06829
y	37574	0.02647
ф	4406	0.0031
X	11742	0.00827
Ц	6460	0.00455
Ч	20695	0.01458
III	9662	0.00681
Щ	4685	0.0033
ы	25661	0.01808
Ь	24874	0.01752
Э	6029	0.00425
Ю	10427	0.00735
Я	25800	0.01818
Ъ	513	0.00036

#### 3.2.2 Робота з біграмами

Рис. 3: к-ть біграм

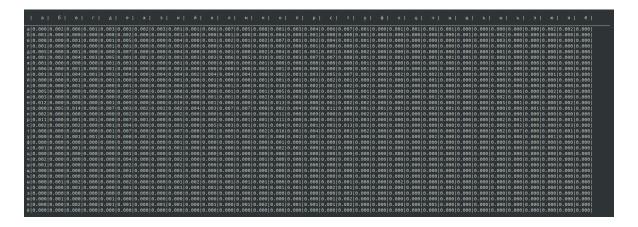


Рис. 4: ймовірність появи біграм

## f 4 Обчислення ентропій $H_1$ та $H_2$

• Обчислення  $H_1$ :

```
fn compute_h1(letter_frequencies: &HashMap < char, i64 >) -> f64 {
    let mut h1 = 0.0;
    let probabilities = count_letters_probabilities(&
    letter_frequencies);

for (_key, _value) in probabilities {
    h1 += _value * f64::log2(_value);
    }

h1 = -h1;
h1
}
```

#### • Обчислення $H_2$ :

```
fn compute_h2(bigram_frequencies: &HashMap < String, i64 >) -> f64 {
    let mut h2 = 0.0;
    let probabilities = count_bigram_probabilities(&
    bigram_frequencies);

for (_key, _value) in probabilities {
        h2 += _value * f64::log2(_value);
    }
    h2 = -h2/2.0;

h2

h2

h3
```

Якщо у тексті наявні пробіли, то  $H_1=4.404$ , якщо ж їх немає, то  $H_1=4.461$ . Так само й для  $H_2$ , з пробілами  $H_2=4.021$ , без  $H_2=4.152$ .

# 5 Оціночні значення величин $H^{(10)},\,H^{(20)}$ та $H^{(30)}$

Використовуючи програму CoolPinkProgram.exe визначаємо приблизні значення:

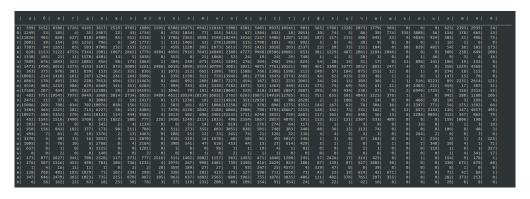


Рис. 5: значення  $H^{(10)}$ 

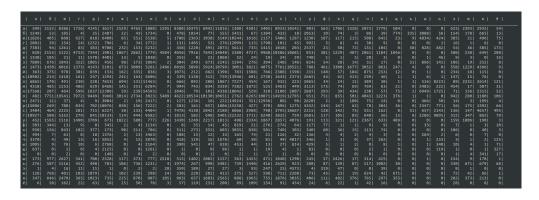


Рис. 6: значення  $H^{(20)}$ 

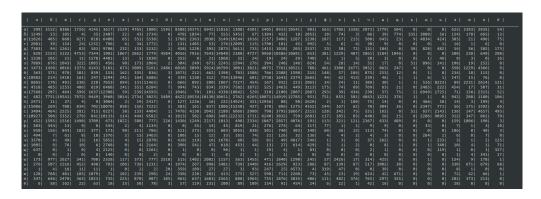


Рис. 7: значення  $H^{(30)}$ 

Маємо, що:

$$\begin{array}{l} a \leq H^{(10)} \leq c \\ a \leq H^{(20)} \leq c \\ a \leq H^{(30)} \leq c \end{array}$$

Тоді можемо обчислити надлишковість  $_p$ осійської мови R використовуючи формулу:

$$R=1-rac{H_{
m inf}}{H_0},$$
де  $H_{
m inf}=H^{(30)},\quad H_0=\log 32pprox 5$ 

звідси а  $\leq R \leq c$  Тобто надлишковість  $_p$ осійської мови  $\approx 0.23$