МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность Информационные системы и технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2 НА ТЕМУ:**

**Элементы теории информации. Параметры и характеристики дискретных информационных систем**

Выполнил студент 3 курса 1 группы

Халалеенко Андрей Николаевич

Минск 2024

**Задание 1:** Рассчитать энтропию двух алфавитов: один на основе латиницы, другой – на кириллице

**Задание 2:** Рассчитать энтропию бинарного алфавита на основе входного файла

**Задание 3:** Используя значения энтропии алфавитов, полученные в заданиях 1 и 2, подсчитать количество информации в сообщении, состоящем из собственных имени, фамилии и отчества

**Задание 4:** выполнить задание 3 при условии, что вероятность ошибочной передачи единичного бита сообщения составляет: 0.1, 0.5, 1.

Энтропию алфавита по К. Шеннону рассчитывают по следующей формуле:

Для выполнения задания были выбраны латышский и болгарский алфавиты. Для расчёта использовался следующий код на языке Typescript:

import \* as fs from 'fs';

class IndividualEntropyCalculator {

    public static calculateIndividualEntropies(text: string, alphabet: string[]): Map<string, number> {

        const totalCharacters = text.length;

        const frequencies = new Map<string, number>();

        // Подсчет частоты появления каждого символа

        for (const c of text) {

            if (frequencies.has(c)) {

                frequencies.set(c, frequencies.get(c)! + 1);

            } else {

                frequencies.set(c, 1);

            }

        }

        // Рассчет вероятности появления каждого символа

        const probabilities = new Map<string, number>();

        for (const [key, value] of frequencies.entries()) {

            probabilities.set(key, value / totalCharacters);

        }

        // Рассчет энтропии для каждого символа

        const entropies = new Map<string, number>();

        for (const [key, value] of probabilities.entries()) {

            const p = value;

            if (p > 0) {

                entropies.set(key, -p \* Math.log2(p));

            } else {

                entropies.set(key, 0);

            }

        }

        return entropies;

    }

    public static printIndividualEntropies(individualEntropies: Map<string, number>): void {

        // Сортировка символов по алфавиту

        const sortedEntropies = new Map([...individualEntropies.entries()].sort());

        // Вывод энтропии для каждого символа

        for (const [key, value] of sortedEntropies.entries()) {

            console.log(`${key} ${value}`);

        }

    }

}

class EntropyCalculator {

    public static calculateEntropy(text: string, alphabet: string[]): number {

        const totalCharacters = text.length;

        const frequencies = new Map<string, number>();

        // Подсчет частоты появления каждого символа

        for (const c of text) {

            const lowercaseChar = c.toLowerCase();

            if (alphabet.includes(lowercaseChar)) {

                const count = frequencies.get(lowercaseChar) || 0;

                frequencies.set(lowercaseChar, count + 1);

            }

        }

        // Расчет вероятности появления каждого символа

        const probabilities = new Map<string, number>();

        for (const c of alphabet) {

            const lowercaseChar = c.toLowerCase();

            if (frequencies.has(lowercaseChar)) {

                probabilities.set(lowercaseChar, frequencies.get(lowercaseChar)! / totalCharacters);

            } else {

                probabilities.set(lowercaseChar, 0);

            }

        }

        // Расчет энтропии

        let entropy = 0;

        for (const p of probabilities.values()) {

            if (p > 0) {

                entropy += -p \* Math.log2(p);

            }

        }

        return entropy;

    }

    public static calculateBinaryEntropy(binaryText: string): number {

        const totalCharacters = binaryText.length;

        const onesCount = binaryText.split('1').length - 1;

        const zerosCount = totalCharacters - onesCount;

        const onesProbability = onesCount / totalCharacters;

        const zerosProbability = zerosCount / totalCharacters;

        return -(onesProbability \* Math.log2(onesProbability) + zerosProbability \* Math.log2(zerosProbability));

    }

    public static calculateInformationSize(entropy: number, message: string): number {

        const totalCharacters = message.length;

        return entropy \* totalCharacters;

    }

}

function main(): void {

    // Задание алфавитов для латышского и таджикского языков

    const latvianAlphabet = ['a', 'ā', 'b', 'c', 'č', 'd', 'e', 'ē', 'f', 'g', 'ģ', 'h', 'i', 'ī', 'j', 'k', 'ķ', 'l', 'ļ', 'm', 'n', 'ņ', 'o', 'p', 'r', 's', 'š', 't', 'u', 'ū', 'v', 'z', 'ž'];

    const tajikAlphabet = ['а', 'б', 'в', 'г', 'ғ', 'д', 'е', 'ё', 'ж', 'з', 'и', 'ӣ', 'й', 'к', 'қ', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ӯ', 'ф', 'х', 'ҳ', 'ч', 'ҷ', 'ш'];

    // Чтение текстов из файлов

    const latvianText = fs.readFileSync('latish.txt', 'utf-8').toLowerCase();

    const tajikText = fs.readFileSync('tadzh.txt', 'utf-8').toLowerCase();

    // Расчет энтропии для текстов на латышском и таджикском языках

    const latvianEntropy = EntropyCalculator.calculateEntropy(latvianText, latvianAlphabet);

    const tajikEntropy = EntropyCalculator.calculateEntropy(tajikText, tajikAlphabet);

    console.log('Latvian Entropy:', latvianEntropy);

    console.log('Tajik Entropy:', tajikEntropy);

    // Преобразование текстов в бинарный код

    const binaryLatvianText = stringToBinary(latvianText);

    const binaryTajikText = stringToBinary(tajikText);

    // Расчет бинарной энтропии для текстов на латышском и таджикском языках

    const binaryLatvianEntropy = EntropyCalculator.calculateBinaryEntropy(binaryLatvianText);

    const binaryTajikEntropy = EntropyCalculator.calculateBinaryEntropy(binaryTajikText);

    console.log('Binary Latvian Entropy:', binaryLatvianEntropy);

    console.log('Binary Tajik Entropy:', binaryTajikEntropy);

    // Пример подсчета количества информации в сообщении на основе латышского алфавита

    const message = 'Halaleenko Andrejs Nikolajevičs ';

    const informationSize = EntropyCalculator.calculateInformationSize(latvianEntropy, message.toLowerCase());

    console.log('Information Size (Name; Latvian alphabet):', informationSize);

    // Преобразование текста в бинарный код (кодировка ASCII)

    const binaryMessage = stringToBinary(message.toLowerCase());

    const binaryInformationSize = EntropyCalculator.calculateInformationSize(1, binaryMessage); // При кодировке ASCII вероятность появления каждого символа равна 1

    console.log('Latish (Based on ASCII):', binaryInformationSize);

    // Пример подсчета количества информации в сообщении на основе таджикского алфавита

    const tajikMessage = 'Халалеенко Андрей Николаевич';

    const tajikInformationSize = EntropyCalculator.calculateInformationSize(tajikEntropy, tajikMessage.toLowerCase());

    console.log('Information Size (Name; Tajik alphabet):', tajikInformationSize);

    // Преобразование текста в бинарный код (кодировка ASCII)

    const binaryTajikMessage = stringToBinary(tajikMessage.toLowerCase());

    const binaryTajikInformationSize = EntropyCalculator.calculateInformationSize(1, binaryTajikMessage); // При кодировке ASCII вероятность появления каждого символа равна 1

    console.log('Tajik (Based on ASCII):', binaryTajikInformationSize);

    // Дополнительные вычисления для пункта (г)

    const p1 = 0.1;

    const p2 = 0.5;

    const p3 = 1.0;

    const errorInformationSize1 = EntropyCalculator.calculateInformationSize(latvianEntropy, message.toLowerCase()) \* p1;

    const errorInformationSize2 = EntropyCalculator.calculateInformationSize(latvianEntropy, message.toLowerCase()) \* p2;

    const errorInformationSize3 = EntropyCalculator.calculateInformationSize(latvianEntropy, message.toLowerCase()) \* p3;

    console.log('Information Size with Error Probability 0.1:', errorInformationSize1);

    console.log('Information Size with Error Probability 0.5:', errorInformationSize2);

    console.log('Information Size with Error Probability 1.0:', errorInformationSize3);

    // Вывод энтропии для каждого символа алфавитов

    console.log('Individual Entropy for Latvian Alphabet:');

    const latvianIndividualEntropies = IndividualEntropyCalculator.calculateIndividualEntropies(latvianText, latvianAlphabet);

    IndividualEntropyCalculator.printIndividualEntropies(latvianIndividualEntropies);

    console.log('\n\n');

    console.log('Individual Entropy for Tajik Alphabet:');

    const tajikIndividualEntropies = IndividualEntropyCalculator.calculateIndividualEntropies(tajikText, tajikAlphabet);

    IndividualEntropyCalculator.printIndividualEntropies(tajikIndividualEntropies);

}

function stringToBinary(text: string): string {

    let binary = '';

    for (const c of text) {

        binary += c.charCodeAt(0).toString(2).padStart(8, '0');

    }

    return binary;

}

main();

Листинг 1 – задания 1 – 2 – 3 – 4

В результате выполнения в консоль выведутся значения энтропии латышского и болгарского алфавитов.

Гистограмма частоты появления символов в тексте на латышском языке:

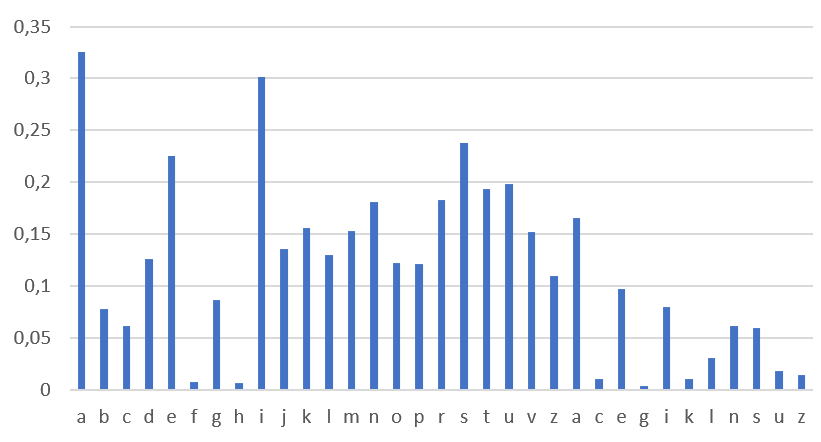


Рис 1 ­­­– Гистограмма текста на латышском языке

Гистограмма частоты появления символов в тексте на болгарском языке:

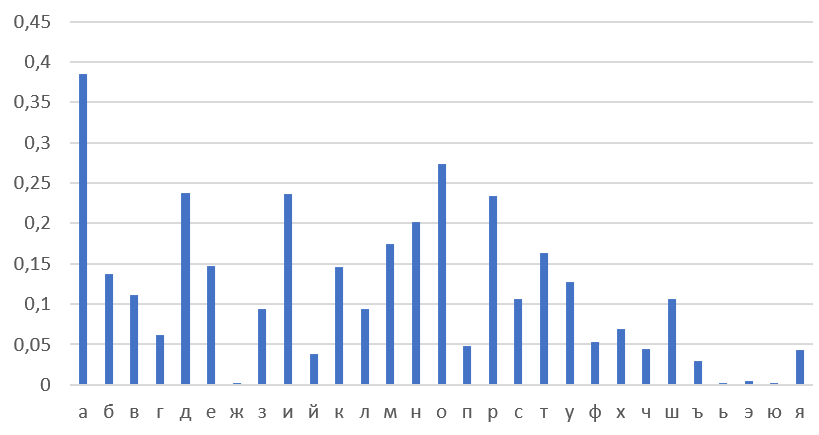


Рис 2 ­­­– Гистограмма текста на таджикском языке

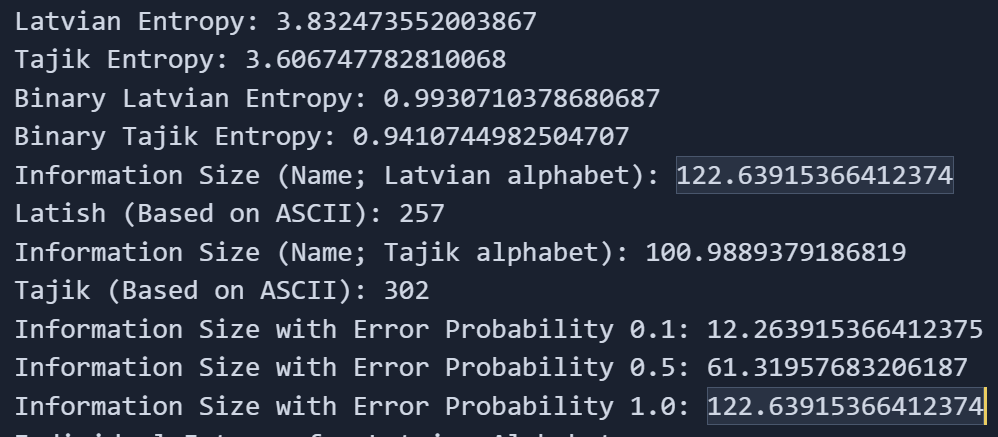


Рис 3 ­­­- Результат работы программы

**Вывод:** в ходе лабораторной работы были изучены методики расчёта энтропии алфавитов и количества информации в сообщениях на основе этих алфавитов. Приобретённые знания были применены для латышского и таджикского алфавитов.