МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

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

**ЭЛЕМЕНТЫ ТЕОРИИ ИНФОРМАЦИИ. ПАРАМЕТРЫ И ХАРАКТЕРИСТИКИ ДИСКРЕТНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ**

Выполнил студент 3 курса 5 группы

Дмитрук Илья Игоревич

Минск 2024

**Задание 1.** Для выполнениялабораторной работы были выбраны македонский язык(кириллица) и турецкий(латиница).

Вычисления выполнялись в программе, написанной на языке программирования C#.

Энтропия сообщения вычислялась по формуле К. Шеннона:

Для вычисления этой энтропии была создана функция GetEntropy. В ней проверяется, является ли рассматриваемое сообщение бинарным, если да, то вычисление энтропии происходит по формуле для бинарного алфавита, если нет, то по формуле К. Шеннона. Функция представлена в листинге 1.1

public static double GetEntropy()

{

double entropy = 0d;

if (NumberSymbols.Count == 2)

{

int onesCount = text.Count(c => c == '1');

double probabilityOfOne = (double)onesCount / text.Length;

double probabilityOfZero = 1 - probabilityOfOne;

entropy = -(probabilityOfOne \* Math.Log(probabilityOfOne, 2) +

probabilityOfZero \* Math.Log(probabilityOfZero, 2));

return Math.Round(entropy, 3);

}

foreach (var symbol in NumberSymbols)

{

double P = (double)symbol.Value / text.Length;

entropy -= P \* Math.Log2(P);

}

return Math.Round(entropy, 3);

}

Листинг 1.1 – Функция для вычисления энтропии алфавита

Для хранения данных, необходимых для вычисления энтропий, были созданы два файла. Данные представляют из себя текста на соответствующих языках. Данные, во время работы программы, считывались с файлов, после чего выполняются вычисления. Результат вычислений представлен на рисунке 1.1.



Рисунок 1.1 – Результаты вычисления энтропии алфавитов

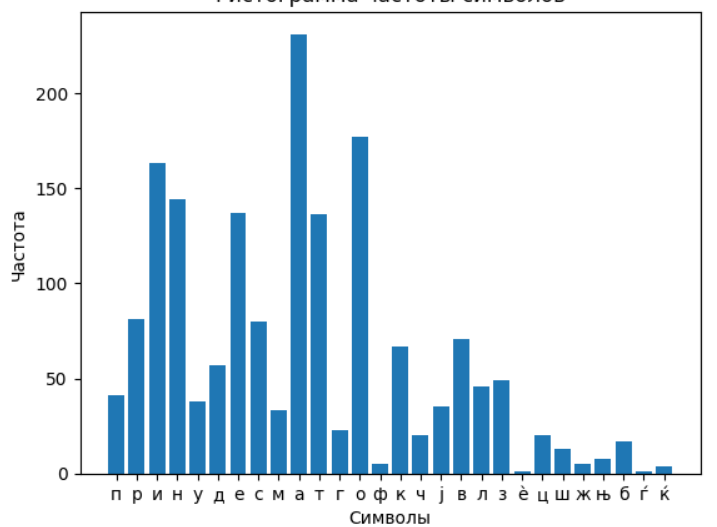


Рисунок 1.2 – Гистограмма частот появления символов македонского алфавита

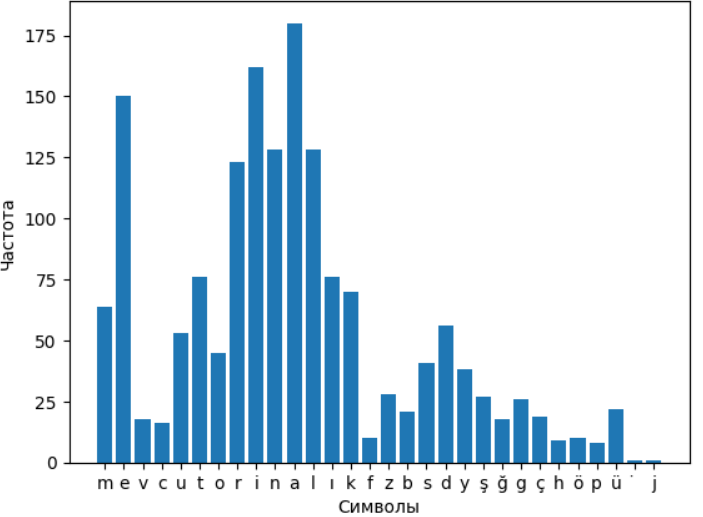


Рисунок 1.3 – Гистограмма частот появления символов турецкого алфавита

**Задание 2.** Для выполнения этого задания был создан ещё один файл. В него были записаны данные, представляющие собой текст в бинарном виде. Данные с файла считывались программой, и вычисление энтропии происходило по формуле для бинарного алфавита:

Формула была реализована в функции GetEntropy, которая была описана ранее. Результат вычисления энтропии бинарного алфавита представлен на рисунке 2.1.



Рисунок 2.1 – Результат вычисления энтропии бинарного алфавита

**Задание 3.** Для вычисления количества информации сообщения, представляющее из себя моё ФИО, каждая из энтропий была умножена на длину текста сообщения по формуле:

*,*

где *H(A*) – энтропия алфавита с соответствующим распределением вероятностей *p(ai*)

В программе формула использовалась в функции GetInformationAmount. Её код представлен в листинге 3.1.

public static double GetInformationAmount()

{

double informationAmount = GetEntropy() \* name.Length;

return Math.Round(informationAmount, 3);

}

Листинг 3.1 – Функция для вычисления эффективной энтропии

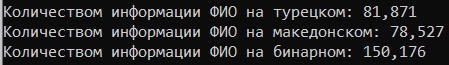


Рисунок 3.1 – Результат вычисления количества информации для каждого сообщения

**Задание 4.** Для нахождения количества информации сообщений, с вероятностью ошибочной передачи единичного бита 0,1; 0,5; 1 необходимо было найти эффективную энтропию:

*H(Y|X)* – условная энтропия, находящаяся по формуле:

,

где *p* – вероятность ошибочной передачи бита, *q* – вероятность правильного передачи бита.

В программе, эффективная энтропия вычислялась в функции GetEffectiveEntropy, код которой представлен в листинге 4.1.

public static double GetEffectiveEntropy(double p)

{

double q = 1 - p;

return 1 - (-p \* Math.Log2(p) - q \* Math.Log2(q));

}

Листинг 4.1 – Функция для вычисления эффективной энтропии

Далее полученный результат необходимо было умножить на количество символов текста и энтропию. Для вычисления количества информации, с вероятностью ошибочной передачи единичного бита сообщения, была разработана функция GetInformationAmount, которая принимает в параметрах вероятность ошибочной передачи. Её код представлен в листинге 4.2.

public static double GetInformationAmount(double p)

{

if ((p == 1 || p == 0) && NumberSymbols.Count == 2)

return GetInformationAmount();

if((p == 1 || p == 0)) return 0;

double informationAmount = GetEntropy() \* name.Length \*

GetEffectiveEntropy(p);

return Math.Round(informationAmount, 3);

}

Листинг 4.2 – Функция для вычисления количества информации с вероятностью ошибочной передачи единичного бита сообщения

Вероятностью ошибочной передачи единичного бита сообщения передавалась в функцию в виде параметра. После чего подставлялась в формулы, после высчитывалась количество информации. Причём, если вероятность ошибки 1, то функция возвращает 0, но если при этом сообщение в бинарном виде, то вычисление количества информации происходило в функции, которая не учитывает вероятность ошибки. Результат вычислений количества информации, с разной вероятностью ошибочной передачи единичного бита сообщения, представлен на рисунке 4.1.

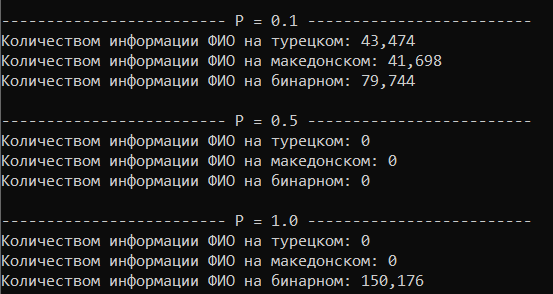


Рисунок 4.1 – Результат вычисления количества информации с вероятностью ошибочной передачи единичного бита

**Вывод:** в ходе лабораторной работы была разработана программа, вычисляющая энтропию текста на различных алфавитах (латиница – турецкий, кириллица – македонский) и в бинарном представлении. Также проводился анализ количества информации в сообщении без вероятности ошибочной передачи единичного бита и с учётом этой вероятности. Рассмотрев результаты, можно сделать вывод, что энтропия текста на разных языках отличная друг от друга. При рассмотрении передачи информации с разными вероятностями ошибок было установлено, что при вероятности ошибки 0,1 передается неполная информация, при 0,5 – никакая информация, а при 1 – передаётся вся информация без изменений.