

Федеральное государственное бюджетное образовательное учреждение высшего
образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

Лабораторная работа № 1
по дисциплине «Теория Информации»

Выполнил:
студент группы ИП-712
Алексеев Степан
Владимирович
ФИО студента

Работу проверил:
доцент кафедры ПМИК Мачикина Е.П.
ФИО преподавателя

Новосибирск 2021 г.

Оглавление

ЗАДАНИЕ	2
Решение	4
Анализ	4
Скриншоты	4
Листинг кода	4

ЗАДАНИЕ

Теория информации

Практическая работа №1

Вычисление энтропии Шеннона

Цель работы: Экспериментальное изучение свойств энтропии Шеннона.

Язык программирования: C, C++, C#, Python

Результат: программа, тестовые примеры, отчет.

Задание:

1. Для выполнения работы необходимо предварительно сгенерировать два файла. Каждый файл содержит последовательность символов, количество различных символов больше 2 (3,4 или 5). Объем файлов больше 10 Кб, формат txt. Символы **последовательно и независимо** генерируются с помощью датчика псевдослучайных чисел и записываются в файл.

Первый файл (назовем его F1) должен содержать последовательность символов с равномерным распределением, т.е. символы должны порождаться равновероятно и независимо.

Для генерации второго файла (F2) необходимо сначала задать набор вероятностей символов, а затем **последовательно и независимо** генерировать символы с соответствующей вероятностью и записывать их в файл.

2. Составить программу, определяющую несколько оценок энтропии созданных текстовых файлов. Вычисление значения по формуле Шеннона **настоятельно рекомендуется** оформить в виде отдельной функции, на вход которой подается массив (список) вероятностей, выходной параметр – значение, вычисленное по формуле Шеннона.

Оценки энтропии необходимо вычислить по формуле Шеннона двумя способами.

Первый способ. Сначала определить частоты отдельных символов файла, т.е. отношения количества отдельного символа к общему количеству символов в файле. Далее используя полученные частоты как оценки вероятностей, рассчитать оценку энтропии по формуле Шеннона.

Второй способ. Определить частоты всех последовательных пар символов в файле. Для того, чтобы правильно рассчитать оценку энтропии для двойных комбинаций символов пары символов нужно рассматривать следующим образом.
Пусть имеется такая последовательность фывафпро

Теория информации

Под парами понимаются пары соседних символов, т.е.
фы ыв ва аф фп пр ро

Для подсчета энтропии подсчитайте частоту встречаемости для каждой пары. Полученное значение энтропии следует разделить на 2. По желанию можно продолжить процесс вычисления оценок с использованием частот троек, четверок символов и т.д.

3. После тестирования программы необходимо заполнить таблицу для отчета и **проанализировать** полученные результаты. Для получения теоретических значений энтропии используйте наборы вероятностей, которые использовались при генерации файлов.

	Оценка энтропии (частоты отдельных символов)	Теоретическое значение энтропии (отдельные символы)	Оценка энтропии (частоты пар символов)	Теоретическое значение энтропии (для пар символов)
F1				
F2				

4. Оформить отчет, загрузить отчет и файл с исходным кодом в электронную среду. Отчет обязательно должен содержать заполненную таблицу и анализ полученных результатов.

По желанию в отчет можно включить описание программной реализации. В отчет не нужно включать содержимое этого файла.

Решение

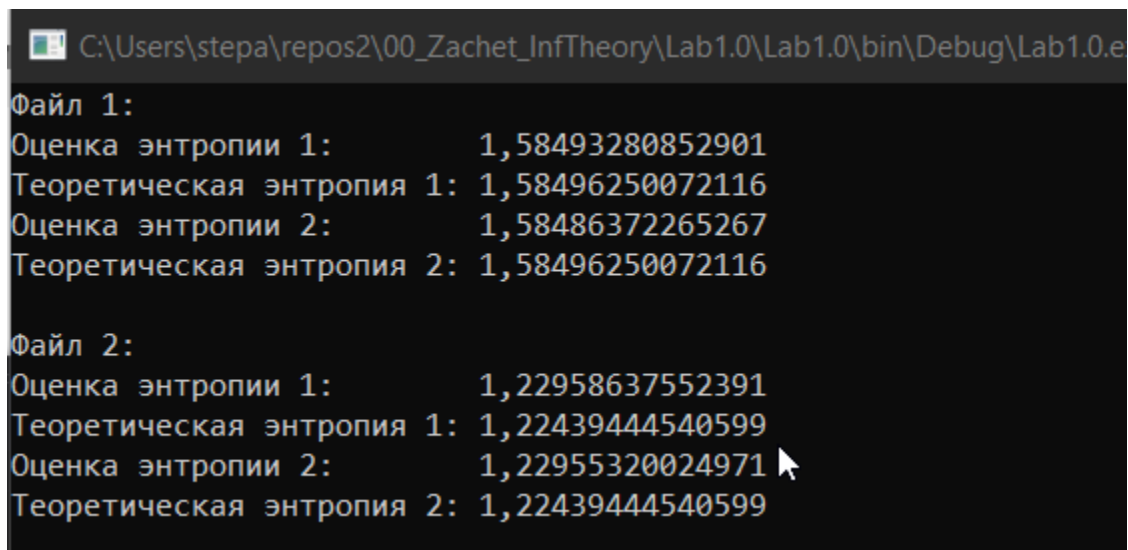
	Оценка энтропии (частоты отдельных символов)	Теоретическое значение энтропии (отдельные символы)	Оценка энтропии (частоты пар символов)	Теоретическое значение энтропии (для пар символов)
F1	1,58493280852901	1,58496250072116	1,58486372265267	1,58496250072116
F2	1,22958637552391	1,22439444540599	1,22955320024971	1,22439444540599

Анализ

При увеличении блоков текста энтропия сильно не изменяется, т.к. мы имеем набор независимых друг от друга символов. Это касается как равновероятных, так и неравновероятных символов. Дело в том, что последующие символы не зависят от предыдущих.

При более частом использовании одних и тех же символов или блоков символов информативность уменьшается (что-то вроде синтаксического сахара получается).
Определено по уменьшению информативности в Файле 2.

Скриншоты



```
C:\Users\stepa\repos2\00_Zachet_InfTheory\Lab1.0\Lab1.0\bin\Debug\Lab1.0.e
Файл 1:
Оценка энтропии 1:      1,58493280852901
Теоретическая энтропия 1: 1,58496250072116
Оценка энтропии 2:      1,58486372265267
Теоретическая энтропия 2: 1,58496250072116

Файл 2:
Оценка энтропии 1:      1,22958637552391
Теоретическая энтропия 1: 1,22439444540599
Оценка энтропии 2:      1,22955320024971
Теоретическая энтропия 2: 1,22439444540599
```

Листинг кода

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```

using System.IO;
using System.Text.RegularExpressions;

namespace Lab1._0
{
    //Не понятен пункт 2 задания. Сказано передать массив(список) вероятностей в
    //функцию. Однако использовать не его, а заново
    //подсчитанные частоты отдельных символов(отношений числа определённых
    //символов к общему числу символов).
    //Зачем тогда передавать массив в функцию, если он не используется? Или нужно
    //посчитать теоретическую и фактическую энтропию?
    //Но как это сделать для пар символов? Ведь вероятности пар не заданы. Или
    //нужно сначала посчитать эти вероятности по
    //фактическому числу символов и потом уже передавать этот список
    //вероятностей?
    class Program
    {
        static int numberOfChars = 30000;
        // static double[] probabilities = {(double) 1 / (double)5, (double)1 / (double)3, (double)1 /
        (double)2 };
        // static string[] alphabet = { "a", "b", "c" };
        //static double[] probabilities = { 0.1, 0.3, 0.6 };//this needs to be sorted in ascending order.
        The above array of probabilities
        //should also be sorted, so that the indexes of symbols and their probabilities are the
        same.
        static Dictionary<string, double> dicti1 = new Dictionary<string, double>();
        static Dictionary<string, double> dicti2 = new Dictionary<string, double>();
        static Dictionary<string, double> dicti3 = new Dictionary<string, double>();
        static int numberOfLettersInABlock = 1;

        static void Main(string[] args)
        {
            dicti1.Add("a", (double)1 / (double)3);
            dicti1.Add("b", (double)1 / (double)3);
            dicti1.Add("c", (double)1 / (double)3);
            fileCreation_3(dicti1, "F1"); //created a first file

            dicti2.Add("a", (double)1 / (double)9);
            dicti2.Add("b", (double)2 / (double)9);
            dicti2.Add("c", (double)6 / (double)9);
            fileCreation_3(dicti2, "F2"); //CREATED A SECOND FILE

            countProbabilitiesBasedOnRealFrequencyInFile("C:/Users/stepa/repos2/00_Zachet_InfTheory/
            Lab1.0/F1.txt", dicti3, numberOfLettersInABlock);
            Console.WriteLine("Файл 1:\n" + "Оценка энтропии 1: " +
            ShannonFormulaForEntropy(dicti3, numberOfLettersInABlock));
        }
    }
}

```

```
Console.WriteLine("Теоретическая энтропия 1: " + ShannonFormulaForEntropy(dicti1,
numberOfLettersInABlock));
```

```
dicti1 = new Dictionary<string, double>();
dicti1.Add("aa", (double)1 / (double)9);
dicti1.Add("ab", (double)1 / (double)9);
dicti1.Add("ac", (double)1 / (double)9);
dicti1.Add("ba", (double)1 / (double)9);
dicti1.Add("bb", (double)1 / (double)9);
dicti1.Add("bc", (double)1 / (double)9);
dicti1.Add("ca", (double)1 / (double)9);
dicti1.Add("cb", (double)1 / (double)9);
dicti1.Add("cc", (double)1 / (double)9);
numberOfLettersInABlock = 2;
dicti3 = new Dictionary<string, double>();
```

```
countProbabilitiesBasedOnRealFrequencyInFile("C:/Users/stepa/repos2/00_Zachet_InfTheory/
Lab1.0/F1.txt", dicti3, numberOfLettersInABlock);
```

```
Console.WriteLine("Оценка энтропии 2: " + ShannonFormulaForEntropy(dicti3,
numberOfLettersInABlock));
```

```
Console.WriteLine("Теоретическая энтропия 2: " + ShannonFormulaForEntropy(dicti1,
numberOfLettersInABlock));
```

```
/* numberOfLettersInABlock = 3;
dicti3 = new Dictionary<string, double>();
```

```
countProbabilitiesBasedOnRealFrequencyInFile("C:/Users/stepa/repos2/00_Zachet_InfTheory/
Lab1.0/F1.txt", dicti3, numberOfLettersInABlock);
```

```
Console.WriteLine("Оценка энтропии 3: " + ShannonFormulaForEntropy(dicti3,
numberOfLettersInABlock));
```

```
numberOfLettersInABlock = 4;
dicti3 = new Dictionary<string, double>();
```

```
countProbabilitiesBasedOnRealFrequencyInFile("C:/Users/stepa/repos2/00_Zachet_InfTheory/
Lab1.0/F1.txt", dicti3, numberOfLettersInABlock);
```

```
Console.WriteLine("Оценка энтропии 4: " + ShannonFormulaForEntropy(dicti3,
numberOfLettersInABlock));
```

```
numberOfLettersInABlock = 20;
dicti3 = new Dictionary<string, double>();
```

```
countProbabilitiesBasedOnRealFrequencyInFile("C:/Users/stepa/repos2/00_Zachet_InfTheory/
Lab1.0/F1.txt", dicti3, numberOfLettersInABlock);
```

```
Console.WriteLine("Оценка энтропии 20: " + ShannonFormulaForEntropy(dicti3,
numberOfLettersInABlock));*/
```

```

//Настраиваемые вероятности:
    numberOfLettersInABlock = 1;
    dicti3 = new Dictionary<string, double>();

countProbabilitiesBasedOnRealFrequencyInFile("C:/Users/stepa/repos2/00_Zachet_InfTheory/
Lab1.0/F2.txt", dicti3, numberOfLettersInABlock);
    Console.WriteLine("\nФайл 2:\nОценка энтропии 1:      " +
ShannonFormulaForEntropy(dicti3, numberOfLettersInABlock));
    Console.WriteLine("Теоретическая энтропия 1: " + ShannonFormulaForEntropy(dicti2,
numberOfLettersInABlock));

    numberOfLettersInABlock = 2;
    dicti3 = new Dictionary<string, double>();

countProbabilitiesBasedOnRealFrequencyInFile("C:/Users/stepa/repos2/00_Zachet_InfTheory/
Lab1.0/F2.txt", dicti3, numberOfLettersInABlock);
    Console.WriteLine("Оценка энтропии 2:      " + ShannonFormulaForEntropy(dicti3,
numberOfLettersInABlock));
    dicti2 = new Dictionary<string, double>();
    dicti2.Add("aa", (double)1 / (double)81);
    dicti2.Add("ab", (double) 2 / (double)81);
    dicti2.Add("ac", (double) 2 / (double)27);
    dicti2.Add("ba", (double) 2 / (double)81);
    dicti2.Add("bb", (double) 4 / (double)81);
    dicti2.Add("bc", (double) 12 / (double)81);
    dicti2.Add("ca", (double) 2 / (double)27);
    dicti2.Add("cb", (double) 12 / (double)81);
    dicti2.Add("cc", (double) 36 / (double)81);

    Console.WriteLine("Теоретическая энтропия 2: " + ShannonFormulaForEntropy(dicti2,
numberOfLettersInABlock));

    /*numberOfLettersInABlock = 3;
    dicti3 = new Dictionary<string, double>();

countProbabilitiesBasedOnRealFrequencyInFile("C:/Users/stepa/repos2/00_Zachet_InfTheory/
Lab1.0/F2.txt", dicti3, numberOfLettersInABlock);
    Console.WriteLine("Оценка энтропии 3:      " + ShannonFormulaForEntropy(dicti3,
numberOfLettersInABlock));

    numberOfLettersInABlock = 4;
    dicti3 = new Dictionary<string, double>();

countProbabilitiesBasedOnRealFrequencyInFile("C:/Users/stepa/repos2/00_Zachet_InfTheory/
Lab1.0/F2.txt", dicti3, numberOfLettersInABlock);
    Console.WriteLine("Оценка энтропии 4:      " + ShannonFormulaForEntropy(dicti3,
numberOfLettersInABlock));

```

```

        numberOfLettersInABlock = 20;
        dicti3 = new Dictionary<string, double>();

countProbabilitiesBasedOnRealFrequencyInFile("C:/Users/stepa/repos2/00_Zachet_InfTheory/
Lab1.0/F2.txt", dicti3, numberOfLettersInABlock);
        Console.WriteLine("Оценка энтропии 20: " + ShennonFormulaForEntropy(dicti3,
numberOfLettersInABlock));*/

        Console.ReadLine();
    }
    static double ShennonFormulaForEntropy(Dictionary<string, double> dict, int
numberOfLettersInABlock)
    {
        //Количество информации, которое мы получаем, достигает максимального
значения, если события равновероятны... Здесь, видимо,
        //сравниваются значения, полученные применением формулы Хартли...
        //Формула Шеннона позволяет высчитать среднее кол-во информации,
передаваемое любым сообщением(блоком символов).
        double sum = 0;
        foreach (var item in dict)
        {
            sum += item.Value * Math.Log(1 / item.Value, 2);
        }
        return sum / numberOfLettersInABlock;//ВЕЗДЕ Д.Б. примерно 1.58
    }
    static void countProbabilitiesBasedOnRealFrequencyInFile(string path, Dictionary<string,
double> dict, int numberOfLettersInABlock)
    {
        string str;
        using (StreamReader sr = File.OpenText(path))
        {
            str = sr.ReadToEnd();
        }
        //1. Apparently I need to split the string by words... No. A whitespace is also a symbol.
        //2. I need to split it on chars.
        //3. Then to get words of needed length by hand
        char[] str_chars = str.ToCharArray();
        for (int i = 0; i < numberOfChars - numberOfLettersInABlock; i++)
        {
            string block = str_chars[i].ToString();
            for (int j = 1; j < numberOfLettersInABlock; j++)
            {
                block += str_chars[i + j].ToString();
            }
            if (dict.ContainsKey(block))
            {

```