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](#_Toc65882147)

[Решение 4](#_Toc65882148)

[Анализ 4](#_Toc65882149)

[Скриншоты 4](#_Toc65882150)

[Листинг кода 4](#_Toc65882151)

# ЗАДАНИЕ

*Теория информации*

Практическая работа №1

Вычисление энтропии Шеннона

Цель работы: Экспериментальное изучение свойств энтропии Шеннона.

Язык программирования: С, С++, С#, 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.

1. 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.

1. 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 |  |  |  |  |

1. 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 | 1,58495764 | 1,58496250 | 2,16983445 | 2,16992500 |
| F2 | 1,22398110 | 1,22439444 | 1,44784936 | 1,44878889 |

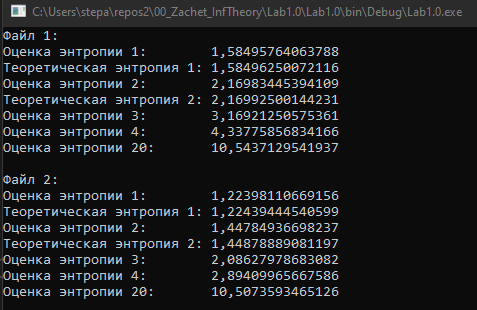
# Анализ

Очевидно, что при увеличении числа символов в сообщении(блоке символов) информативность сообщений и всего текста увеличивается(определено по росту битов с ростом блоков кода).

Также 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.

Интересно, что при длине блока 20 информативность у обоих файлов почти сравнялась. Видимо это потому, что такие длинные слова в таком небольшом количетсве всего текста в итоге несут одну и ту же полезную информацию.

# Скриншоты



# 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");

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");

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

Console.WriteLine("Файл 1:\n" + "Оценка энтропии 1: " + ShennonFormulaForEnthropy(dicti3, numberOfLettersInABlock));

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

dicti1 = new Dictionary<string, double>();

dicti1.Add("aa", (double)2 / (double)9);

dicti1.Add("ab", (double)2 / (double)9);

dicti1.Add("ac", (double)2 / (double)9);

dicti1.Add("ba", (double)2 / (double)9);

dicti1.Add("bb", (double)2 / (double)9);

dicti1.Add("bc", (double)2 / (double)9);

dicti1.Add("ca", (double)2 / (double)9);

dicti1.Add("cb", (double)2 / (double)9);

dicti1.Add("cc", (double)2 / (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: " + ShennonFormulaForEnthropy(dicti3, numberOfLettersInABlock));

Console.WriteLine("Теоретическая энтропия 2: " + ShennonFormulaForEnthropy(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: " + ShennonFormulaForEnthropy(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: " + ShennonFormulaForEnthropy(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: " + ShennonFormulaForEnthropy(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: " + ShennonFormulaForEnthropy(dicti3, numberOfLettersInABlock));

Console.WriteLine("Теоретическая энтропия 1: " + ShennonFormulaForEnthropy(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: " + ShennonFormulaForEnthropy(dicti3, numberOfLettersInABlock));

dicti2 = new Dictionary<string, double>();

dicti2.Add("aa", (double)2 / (double)81);

dicti2.Add("ab", (double)2 \* 2 / (double)81);

dicti2.Add("ac", (double)2 \* 2 / (double)27);

dicti2.Add("ba", (double)2 \* 2 / (double)81);

dicti2.Add("bb", (double)2 \* 4 / (double)81);

dicti2.Add("bc", (double)2 \* 12 / (double)81);

dicti2.Add("ca", (double)2 \* 2 / (double)27);

dicti2.Add("cb", (double)2 \* 12 / (double)81);

dicti2.Add("cc", (double)2 \* 36 / (double)81);

Console.WriteLine("Теоретическая энтропия 2: " + ShennonFormulaForEnthropy(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: " + ShennonFormulaForEnthropy(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: " + ShennonFormulaForEnthropy(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: " + ShennonFormulaForEnthropy(dicti3, numberOfLettersInABlock));

Console.ReadKey();

}

static double ShennonFormulaForEnthropy(Dictionary<string, double> dict, int numberOfLettersInABlock)

{//Количество информации, которое мы получаем, достигает максимального значения, если события равновероятны... Здесь, видимо,

//сравниваются значения, полученные применениемформулы 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;

}

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))

{

dict[block] += ((double)1 / ((double)numberOfChars / (double)numberOfLettersInABlock));

}

else

dict.Add(block, ((double)1 / ((double)numberOfChars / (double)numberOfLettersInABlock)));

}//up to here all occurences of blocks are counted and frequencies(counted probabilities) are counted.

//Time to use Shennon's formula

}

static void fileCreation\_3(Dictionary<string, double> dict, string fileName)

{

var rand = new Random();

string path = @"C:/Users/stepa/repos2/00\_Zachet\_InfTheory/Lab1.0/" + fileName + ".txt";

{

using (StreamWriter sw = File.CreateText(path))

{

for (int i = 0; i < numberOfChars; i++)

{

double num = rand.NextDouble();//here the randomity occures

double sum = 0;

foreach (var item in dict)

{

sum += item.Value;

if (num <= sum)//have found the position of the random number among the probabilities

{

sw.Write(item.Key);

break;

}

}

}

}

}

}

}

}