БУ ВО «Сургутский государственный университет»

Политехнический институт

Кафедра автоматизированных систем обработки информации и управления

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

ПО ДИСЦИПЛИНЕ «Теория Информации»

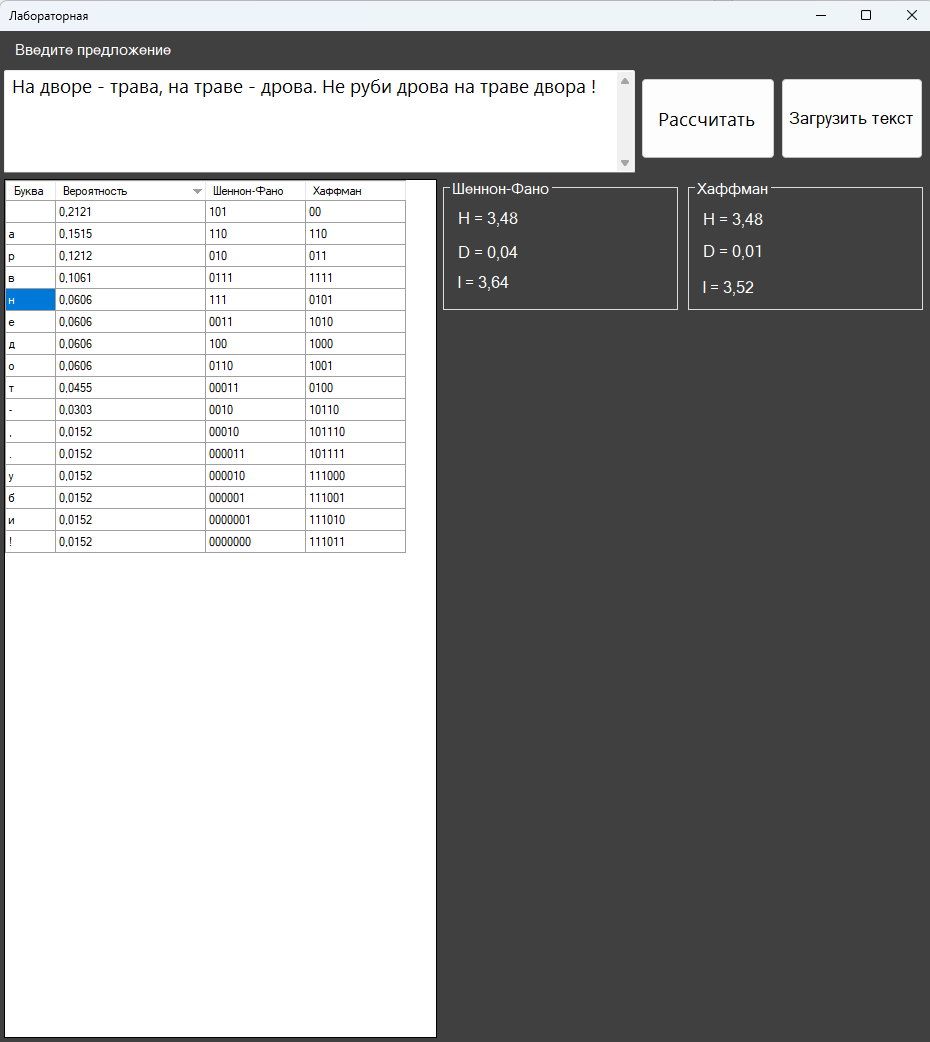
Выполнил: студент группы №606-12,

Речук Дмитрий Максимович

Принял: ст. преподаватель кафедры АСОИУ,

Гавриленко Анна Владимировна

Сургут 2025



using System;

using System.Collections.Generic;

using System.Data;

using System.IO;

using System.Linq;

using System.Windows.Forms;

namespace Kod

{

public partial class Form1 : Form

{

private string textVariable;

private bool check = false;

private string str;

private const string ShannonColumn = "ShannonFano";

private const string HuffmanColumn = "Huffman";

public Form1()

{

InitializeComponent();

InitializeDataGridView();

}

private void InitializeDataGridView()

{

resDataGridView.Columns.Add(ShannonColumn, "Шеннон-Фано");

resDataGridView.Columns.Add(HuffmanColumn, "Хаффман");

}

private void Button1\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(TextBox.Text))

{

MessageBox.Show("Пожалуйста, введите текст или загрузите файл.", "Неверный ввод");

return;

}

try

{

if (textVariable != TextBox.Text)

check = false;

str = check ? textVariable.ToLower() : TextBox.Text.ToLower();

// Calculate probabilities

Dictionary<char, double> probabilities = CalculateProbabilities(str);

// Shannon-Fano

var (shannonCodes, lFan) = CalculateShannonFano(probabilities);

// Entropy

double hFan = CalculateEntropy(probabilities.Values.ToArray());

double dFan = lFan > 0 ? (lFan - hFan) / lFan : 0;

// Huffman

var (huffmanCodes, lHof) = CalculateHuffman(probabilities);

double dHof = lHof > 0 ? (lHof - hFan) / lHof : 0;

// Update UI

PopulateDataGridView(probabilities, shannonCodes, huffmanCodes);

ShIlabel.Text = $"l = {Math.Round(lFan, 2)}";

ShHlabel.Text = $"H = {Math.Round(hFan, 2)}";

ShDlabel.Text = $"D = {Math.Round(dFan, 2)}";

HafILabel.Text = $"l = {Math.Round(lHof, 2)}";

HafHlabel.Text = $"H = {Math.Round(hFan, 2)}";

HafDlabel.Text = $"D = {Math.Round(dHof, 2)}";

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка обработки данных: {ex.Message}", "Ошибка");

}

}

private void Button2\_Click(object sender, EventArgs e)

{

check = true;

OpenFileDialog openFileDialog = new OpenFileDialog

{

Filter = "Текстовые файлы (\*.txt)|\*.txt|Все файлы (\*.\*)|\*.\*",

Title = "Выберите текстовый файл"

};

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

string filePath = openFileDialog.FileName;

textVariable = File.ReadAllText(filePath);

TextBox.Text = textVariable;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка чтения файла: {ex.Message}", "Ошибка файла");

check = false;

}

}

}

private Dictionary<char, double> CalculateProbabilities(string text)

{

if (string.IsNullOrEmpty(text))

return new Dictionary<char, double>();

return text.GroupBy(c => c)

.ToDictionary(

g => g.Key,

g => (double)g.Count() / text.Length

);

}

private void PopulateDataGridView(Dictionary<char, double> probabilities, string[] shannonCodes, List<string> huffmanCodes)

{

resDataGridView.Rows.Clear();

var keys = probabilities.Keys.ToArray();

for (int i = 0; i < keys.Length; i++)

{

resDataGridView.Rows.Add(

keys[i],

Math.Round(probabilities[keys[i]], 4),

shannonCodes[i],

huffmanCodes[i]

);

}

}

private double CalculateEntropy(double[] probabilities)

{

return probabilities.Sum(p => F(p));

}

private double F(double x)

{

return x > 0 ? -x \* Math.Log(x, 2) : 0;

}

private (string[], double) CalculateShannonFano(Dictionary<char, double> probabilities)

{

ShannonFanoEncoder encoder = new ShannonFanoEncoder(probabilities.Values.ToArray());

string[] codes = encoder.Encode();

double avgLength = probabilities.Values

.Zip(codes, (prob, code) => prob \* code.Length)

.Sum();

return (codes, avgLength);

}

private (List<string>, double) CalculateHuffman(Dictionary<char, double> probabilities)

{

HuffmanTree huffmanTree = new HuffmanTree();

huffmanTree.Build(probabilities);

List<string> codes = huffmanTree.ReturnAlphabet();

double avgLength = probabilities.Values

.Zip(codes, (prob, code) => prob \* code.Length)

.Sum();

return (codes, avgLength);

}

}

}