МГТУ имени Н. Э. Баумана

Базовые компоненты интернет технологий

Отчет по лабораторной работе №5

Кочетков Михаил Дмитриевич
Группа ИУ5-31б

10 октября 2018 г.

Москва
2018

Задание

Разработать программу, реализующую вычисление расстояния Левенштейна с использованием алгоритма Вагнера-Фишера.

1. Программа должна быть разработана в виде библиотеки классов на языке C#
2. Использовать самый просто вариант алгоритма без оптимизации
3. Дополнительно возможно реализовать вычисление расстоягия Дамерау-Левенштейна (с учетом перестановок соседних символов)
4. Модифицировать предыдущую лабораторную работу, вместо поиска подстроки используется вычисление расстояния Левенштейна
5. Предусмотреть отдельное поле ввода для максимального расстояния. Если расстояние Левенштейна между двумя строками больше максимального, то строки считаются несовпадающими и не выводятся в список результатов

Код

Program.cs

```
using System;
using Eto.Forms;
using Eto.Drawing;

namespace Lab_5.Desktop
{
    class Program
    {
        [STAThread]
        static void Main(string[] args)
        {
            new Application(Eto.Platform.Detect).Run(new MainForm());
        }
    }
}
```

MainForm.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using Eto.Forms;
using Eto.Drawing;

namespace Lab_5
{
    public class MainForm : Form
    {
        public MainForm()
        {
            ClientSize = new Size(400, 400);
            Title = "Lab 5";

            var wordList = new List<string>();

            var timeLabel = new Label();

            var openFileButton = new Button { Text = "Open File" };
            openFileButton.Click += delegate
            {
                var openFileDialog = new OpenFileDialog
                {
                    MultiSelect = false,
                    Filters = {"Text|*.txt"}
                };

                var stopWatch = new Stopwatch();

                if (openFileDialog.ShowDialog(this) == DialogResult.Ok)
                {
                    stopWatch.Start();
                    var file = File.ReadAllText(openFileDialog.FileName);
                    foreach (var word in file.Split(' '))
                    {
                        if (!wordList.Contains(word))
                        {
                            wordList.Add(word);
                        }
                    }
                }

                stopWatch.Stop();
                timeLabel.Text = "Time of opening and scanning: " +
                ↪  stopWatch.ElapsedMilliseconds + " ms";
            };

            var textBox = new TextBox();
```

```
50              var listBox = new ListBox();
51              var timeFindLabel = new Label();
52              var MaxDistLabel = new Label();
53              var MaxDistTextBox = new TextBox();
54
55              MaxDistLabel.Text = "Enter max distance between words";
56              textBox.PlaceholderText = "Enter word to find";
57
58              var findButton = new Button { Text = "Find word" };
59              findButton.Click += delegate
60              {
61                  listBox.Items.Clear();
62
63                  var expectedSubstring = textBox.Text;
64                  if (expectedSubstring.Trim(' ') == "")
65                  {
66                      listBox.Items.Add("Empty field");
67                      return;
68                  }
69
70                  if (MaxDistTextBox.Text.Trim(' ') == "")
71                  {
72                      listBox.Items.Add("Empty max distance field");
73                      return;
74                  }
75
76                  var maxDist = Int32.Parse(MaxDistTextBox.Text);
77                  var isFinded = false;
78
79                  var stopWatch = new Stopwatch();
80                  stopWatch.Start();
81
82                  foreach (var word in wordList)
83                  {
84                      if (DistDamerau(word, expectedSubstring) <= maxDist)
85                      {
86                          listBox.Items.Add(word);
87                          isFinded = true;
88                      }
89                  }
90
91                  stopWatch.Stop();
92                  if (!isFinded)
93                  {
94                      listBox.Items.Add("No matches");
95                  }
96
97                  timeFindLabel.Text = "Time of searching: " + stopWatch.ElapsedMilliseconds
                    ↪  + " ms";
98              };
99
100             var layout = new TableLayout
```

```
101             {
102                 Padding = new Padding(10),
103                 Spacing = new Size(5, 5),
104                 Rows =
105                 {
106                     new TableRow(openFileButton, timeLabel),
107                     new TableRow(textBox, findButton),
108                     new TableRow(MaxDistLabel, MaxDistTextBox),
109                     new TableRow(listBox, timeFindLabel)
110                 }
111             };
112 
113             Content = layout;
114         }
115 
116         private static int Dist(string s1, string s2)
117         {
118             if (s1 == s2)
119             {
120                 return 0;
121             }
122 
123             var M = s1.Length + 1;
124             var N = s2.Length + 1;
125 
126             var dist = new int[M, N];
127 
128             for (var i = 0; i < M; i++)
129             {
130                 dist[i, 0] = i;
131             }
132 
133             for (var j = 0; j < N; j++)
134             {
135                 dist[0, j] = j;
136             }
137 
138             for (var i = 1; i < M; i++)
139             {
140                 for (var j = 1; j < N; j++)
141                 {
142                     var diff = (s1[i - 1] == s2[j - 1]) ? 0 : 1;
143 
144                     dist[i, j] = Math.Min(
145                         Math.Min(
146                             dist[i - 1, j] + 1,
147                             dist[i, j - 1] + 1
148                         ),
149                         dist[i - 1, j - 1] + diff
150                     );
151                 }
152             }
```

```
153
154             return dist[M - 1, N - 1];
155         }
156
157         private static int DistDamerau(string s1, string s2)
158         {
159             if (s1 == s2)
160             {
161                 return 0;
162             }
163
164             var M = s1.Length + 1;
165             var N = s2.Length + 1;
166
167             var dist = new int[M, N];
168
169             for (var i = 0; i < M; i++)
170             {
171                 dist[i, 0] = i;
172             }
173
174             for (var j = 0; j < N; j++)
175             {
176                 dist[0, j] = j;
177             }
178
179             for (var i = 1; i < M; i++)
180             {
181                 for (var j = 1; j < N; j++)
182                 {
183                     if (s1[i - 1] == s2[j - 1])
184                     {
185                         dist[i, j] = dist[i - 1, j - 1];
186                     }
187
188                     var diff = (s1[i - 1] == s2[j - 1]) ? 0 : 1;
189
190                     dist[i, j] = Math.Min(
191                         Math.Min(
192                             dist[i - 1, j] + 1,
193                             dist[i, j - 1] + 1
194                         ),
195                         dist[i - 1, j - 1] + diff
196                     );
197
198                     if (i > 1 && j > 1 && s1[i - 2] == s2[j - 1] && s1[i - 1] == s2[j -
                    ↪  2])
199                     {
200                         dist[i, j] = Math.Min(dist[i, j], dist[i - 2, j - 2] + 1);
201                     }
202                 }
203             }
```

```
204
205             return dist[M - 1, N - 1];
206         }
207     }
208 }
```

Тесты

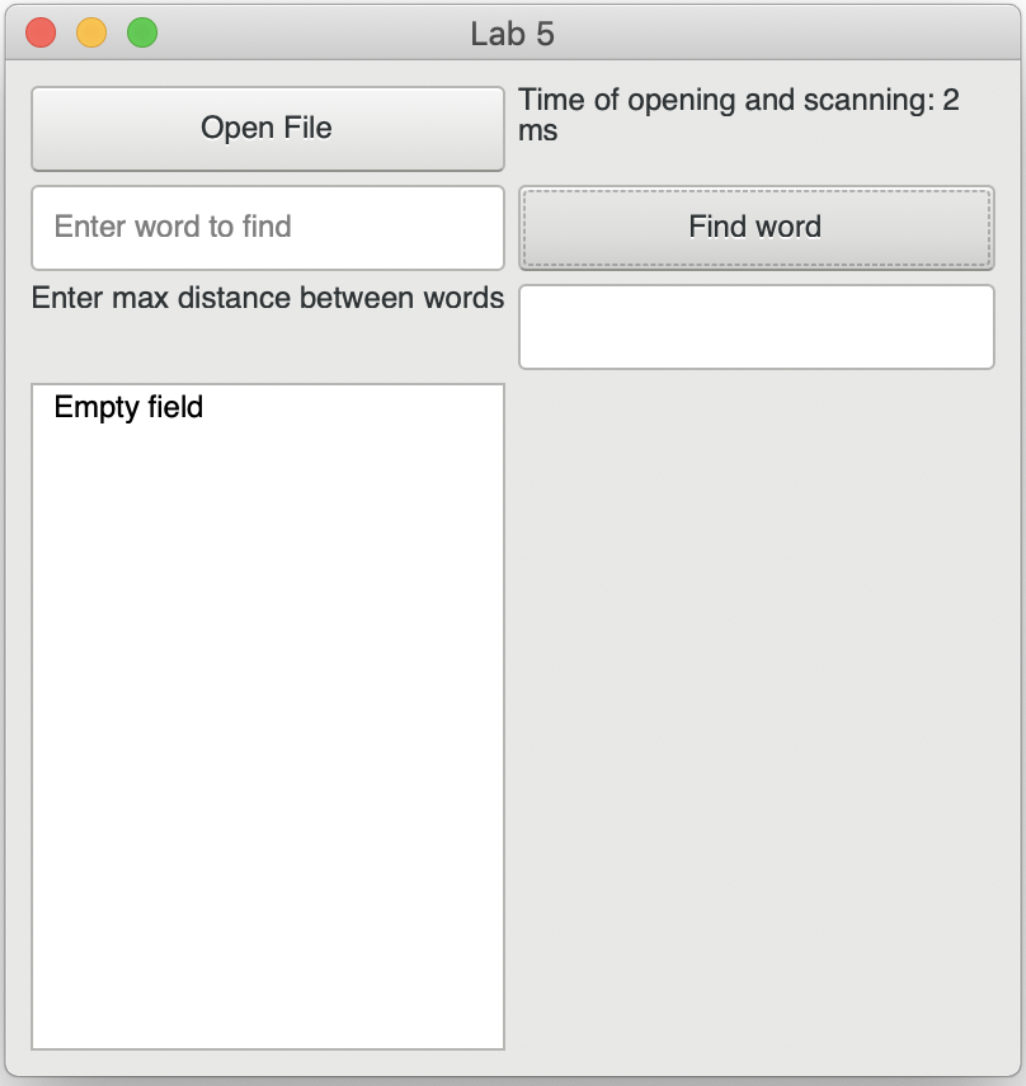

Рис. 1: Пустой ввод

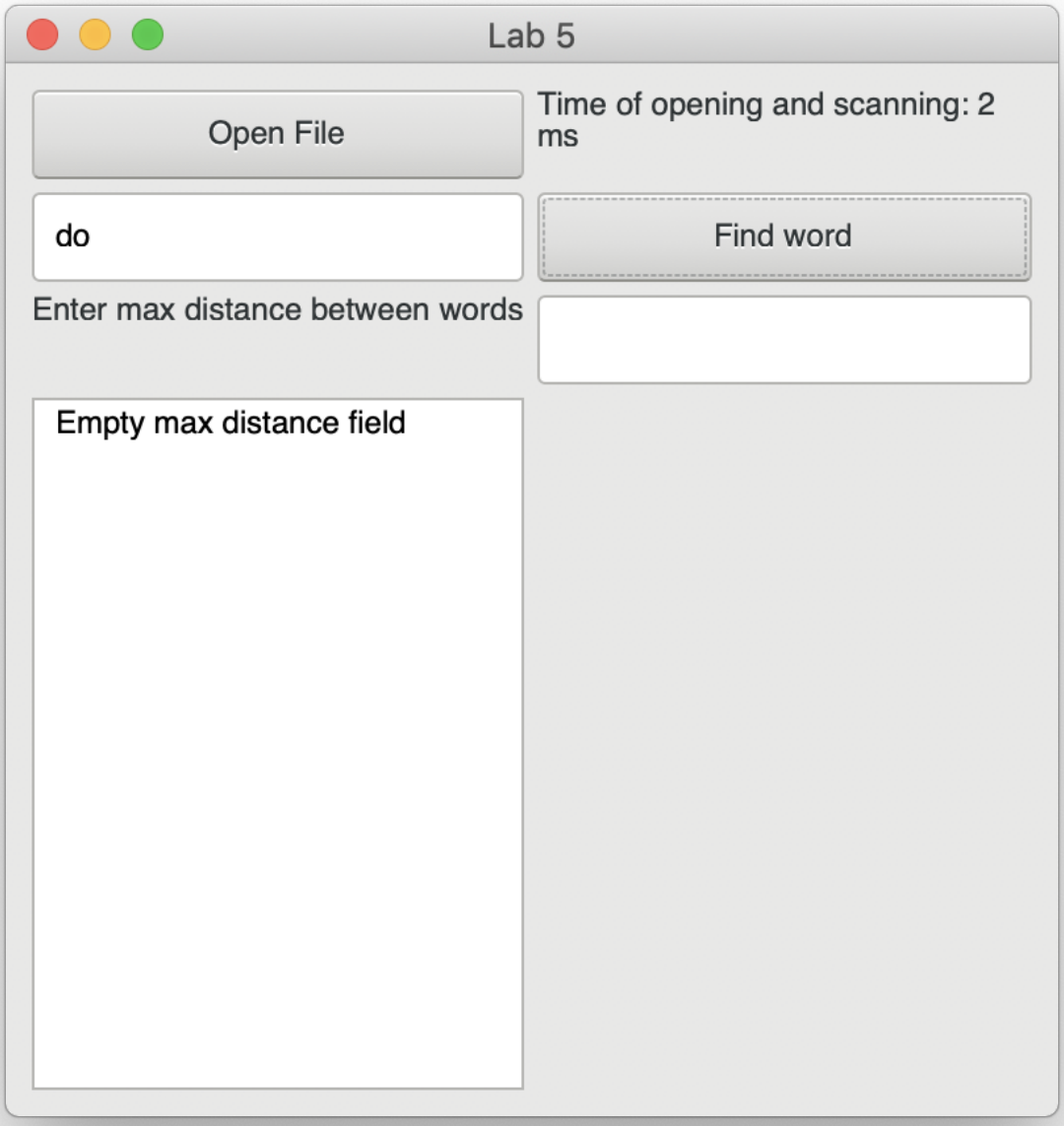

Рис. 2: Не введено максимальное растояние Левенштейна

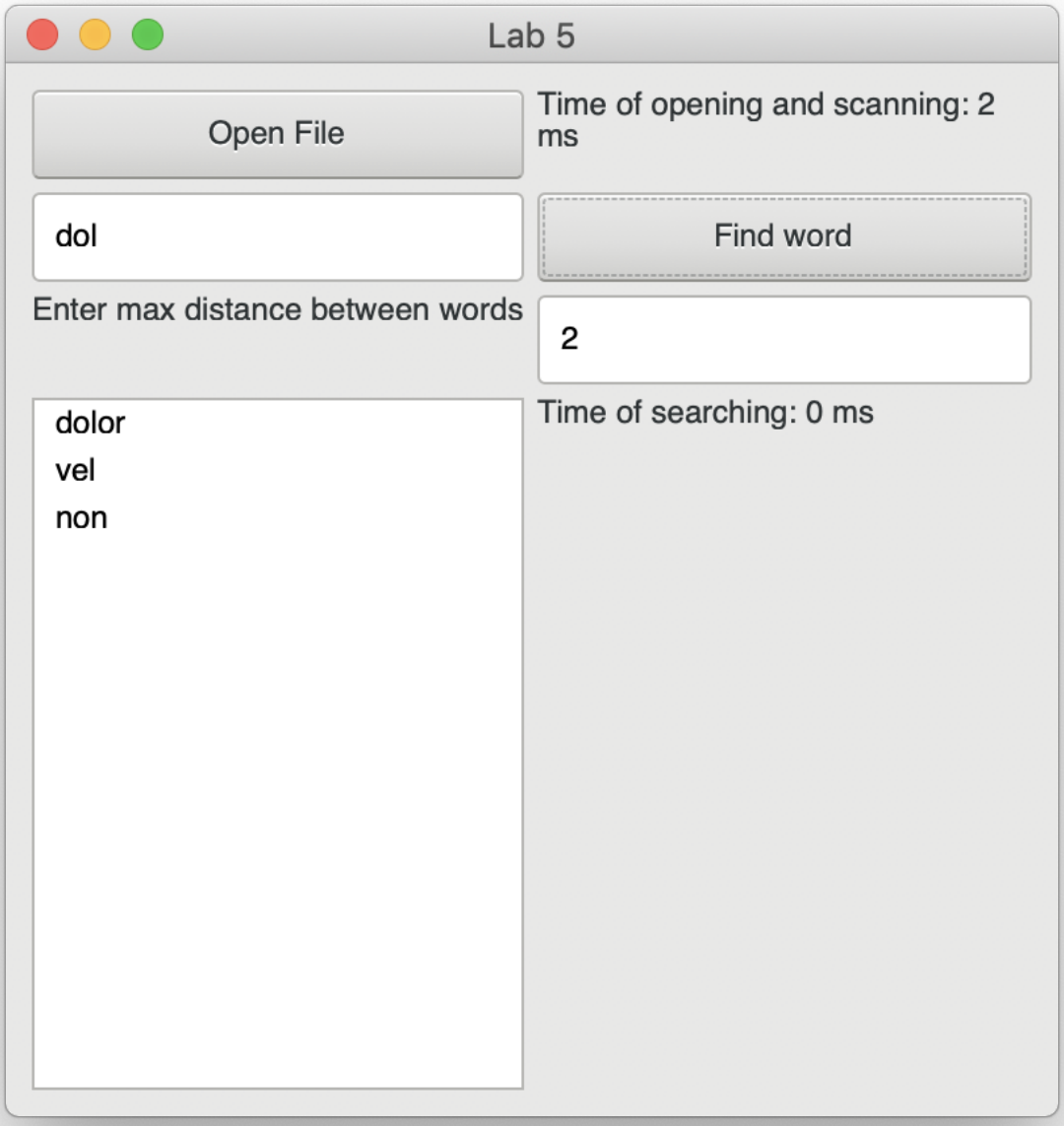

Рис. 3: Поиск с использованием алгоритма Дамерау-Левенштейна