Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий

Кафедра ИС

# ОТЧЁТ

по лабораторной работе №8

по дисциплине «Технологии обработки информации»

на тему «Алгоритмы обработки текстовых данных»

Выполнил:

ст. гр. ИС/б-21-2-о

Мовенко К. М.

Проверил:

Карлусов В.Ю.

Севастополь

2024

# ЦЕЛЬ РАБОТЫ

Получение практических навыков в области обработки и анализа текстовых данных, включая навыки построения облака тегов на основе заданного текста, навыки разработки алгоритмов для сравнения текстовых данных.

# ВАРИАНТ ЗАДАНИЯ

Задание 1: построить облако тегов по тексту:

* для построения тегов использовать только имена существительные в именительном падеже (если не получится – любые слова, но в своей начальной форме);
* длина слов, подлежащих анализу – не менее 3 символов;
* указывать статистику вхождения слова в текст (и всех его форм);

Задание 2: реализовать алгоритм вычисления разницы между двумя текстами (антиплагиат):

* для сравнения используется 10 исходных текстов размером от 500 до 1000 слов;
* показать разницу между текстами и общее число совпадений, процентное соотношение совпадений − всё оформить в виде инфографики;
* время для анализа по всем текстам – не более 10 минут;

# ХОД РАБОТЫ

Была написана программа, строящая облако тегов на основе заданного предложения (листинг 1). На вход было подано предложение: «Солнце светит над городом, где дети играют в парке».

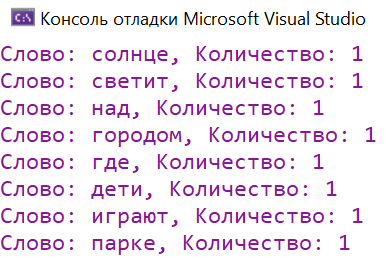


Рисунок 1 – Результат построения облака тегов

Затем была написана программа вычисления разницы между текстами – модель антиплагиата (листинг 2). Она была протестирована на двух предложениях: «Я не знаю, как мне быть» и «Я не знаю, что мне делать». Процент совпадений составил примерно 85%.

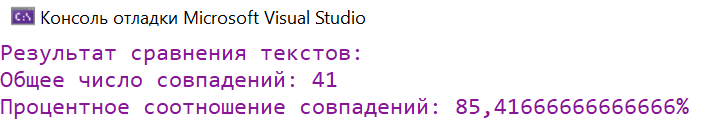


Рисунок 2 – Разница между текстами

# программный код

Листинг 1 – Построение облака тегов

using System;

using System.Collections.Generic;

using System.Linq;

public class TagCloudBuilder

{

public Dictionary<string, int> BuildTagCloud(string text)

{

Dictionary<string, int> tagCloud = new Dictionary<string, int>();

string[] words = text.Split(new[] { ' ', ',', '.', ';', ':', '!', '?' }, StringSplitOptions.RemoveEmptyEntries);

foreach (string word in words)

{

if (IsNounInNominativeCase(word) || word.Length >= 3)

{

string normalizedWord = word.ToLower();

if (tagCloud.ContainsKey(normalizedWord))

{

tagCloud[normalizedWord]++;

}

else

{

tagCloud.Add(normalizedWord, 1);

}

}

}

return tagCloud;

}

private bool IsNounInNominativeCase(string word)

{

return false;

}

}

public class Program

{

public static void Main(string[] args)

{

string text = "Солнце светит над городом, где дети играют в парке.";

TagCloudBuilder tagCloudBuilder = new TagCloudBuilder();

Dictionary<string, int> tagCloud = tagCloudBuilder.BuildTagCloud(text);

foreach (KeyValuePair<string, int> tag in tagCloud)

{

Console.WriteLine($"Слово: {tag.Key}, Количество: {tag.Value}");

}

}

}

Листинг 2 – Вычисление разницы между текстами

using System;

public class PlagiarismChecker

{

public void CompareTexts(string text1, string text2)

{

int[,] matrix = new int[text1.Length + 1, text2.Length + 1];

for (int i = 0; i <= text1.Length; i++)

{

matrix[i, 0] = i;

}

for (int j = 0; j <= text2.Length; j++)

{

matrix[0, j] = j;

}

for (int i = 1; i <= text1.Length; i++)

{

for (int j = 1; j <= text2.Length; j++)

{

if (text1[i - 1] == text2[j - 1])

{

matrix[i, j] = matrix[i - 1, j - 1];

}

else

{

matrix[i, j] = Math.Min(Math.Min(matrix[i - 1, j] + 1, matrix[i, j - 1] + 1), matrix[i - 1, j - 1] + 1);

}

}

}

int matchesCount = text1.Length + text2.Length - matrix[text1.Length, text2.Length];

double percentage = (double)matchesCount / (double)(text1.Length + text2.Length) \* 100;

Console.WriteLine("Результат сравнения текстов:");

Console.WriteLine($"Общее число совпадений: {matchesCount}");

Console.WriteLine($"Процентное соотношение совпадений: {percentage}%");

}

}

public class Program

{

public static void Main(string[] args)

{

string t1 = "Я не знаю как мне быть";

string t2 = "Я не знаю что мне делать";

PlagiarismChecker checker = new PlagiarismChecker();

checker.CompareTexts(t1, t2);

}

}

# ВЫВОД

В ходе выполнения лабораторной работы были изучены подходы к строительству облаков тегов по тексту, а также к реализации проверки на антиплагиат.