Лабораторная работа №5

студента группы ИТ – 32

Курбатовой Софьи Андреевны

Выполнение:_____ Защита _____

РЕАЛИЗАЦИЯ АЛГОРИТМА ОБНАРУЖЕНИЯ НЕЧЕТКИХ ДУБЛИКАТОВ.

Цель работы: научиться реализовывать на выбранном языке программирования алгоритмы обнаружения нечетких дубликатов на основе алгоритма шинглов.

Содержание работы

Написать программу на выбранном языке программирования, реализующую поиск нечетких дубликатов заданных текстов описанным выше алгоритмом. Программа должна запрашивать имена входных файлов и выводить схожие документы и степень их схожести (в процентах)..

Ход работы

1. Для того, чтобы сравнить два текста необходимо выполнить следующие этапы:

1. Провести канонизацию текста. Здесь предполагается удаление из текста знаков препинания, стоп-слов.

2. Выполнить разбиение на шинглы;

3. Вычислить хэши шинглов с помощью 84х статических функций;

4. Провести случайная выборка 84 значений контрольных сумм. В данной реализации ищется минимальное значение;

5. Выполнить пункты 1-4 для каждого текста.

6. Произвести по полученным хеш-таблицам из полученных в п.4 значений.

7. Определить результат как процентное соотношение схожести текстов.

Листинг 1.1. Главный модуль.

```
class Program
    {
        static void Main(string[] args)
        {
            TextComparisonEngine engine = new TextComparisonEngine();

            while (true)
            {
                Console.WriteLine("Enter names of local text files to compare:");
                Console.Write("File #1: ");
                string textA = File.ReadAllText(Console.ReadLine(),
System.Text.Encoding.GetEncoding(1251));
                Console.Write("File #2: ");
                string textB = File.ReadAllText(Console.ReadLine(),
System.Text.Encoding.GetEncoding(1251));

                engine.Compare(textA, textB);
                engine.PrintResults();
            }
```

```
        }
    }
```

Листинг 1.2. Класс с методами для сравнения текстов

```
  class ShinglesComparer : ITextComparer
    {
        public double Result { get; private set; }
        public List<int> HashesA { get; private set; }
        public List<int> HashesB { get; private set; }

        public int HashCount { get; private set; }
        public int ShingleSize { get; private set; }

        public ShinglesComparer(int HashCount, int ShingleSize)
        {
            this.HashCount = HashCount;
            this.ShingleSize = ShingleSize;
        }

        public double ProcessTexts(string textA, string textB)
        {
            HashesA = CalcMinHashesFromText(textA, HashCount, ShingleSize);
            HashesB = CalcMinHashesFromText(textB, HashCount, ShingleSize);
            return EstimateSimilarity(HashCount);
        }

/*------------------------Метод для сравнения сумм в хеш-таблицах двух текстов-----------------*/
        static public int CompareHashes(List<int> hashes1, List<int> hashes2)
        {
            int sim = 0;
            for (int i = 0; i < hashes1.Count; i++)
            {
                if (hashes1[i] == hashes2[i])
                {
                    sim += 1;
                }
            }
            return sim;
        }
/*-----------------------------------------------------------------------------------------------*/

        private double EstimateSimilarity(int hashCount)
        {
            double sim = CompareHashes(HashesA, HashesB);
            sim /= (double)hashCount;
            Result = sim;
            return sim;
        }


        private List<int> CalcMinHashesFromText(string text,int hashCount = 36, int shingleSize =
10)
        {
                     return
          GetMinHashes(
          CalcHashes(
          GetShingles(Canonisation(text),shingleSize),//получим шинглы из текста
          hashCount));
        }

/*-------------------Метод удаления стоп-символов и стоп-слов из текста-------------------------
*/
        private List<string> Canonisation(string text)
        {
            var resText = new List<string>();
            foreach (var word in text.ToLower().Split(Constants.StopSymbols))//стоп-символы
например: , . !
```

```
            {
                if (!Constants.StopWords.Contains(word))//если слова нет в списке стоп-слова
                {
                    resText.Add(word);//добавим его в результат
                }
            }
            return resText;
        }
/*-------------------------------------------------------------------------------------------*/
/*---------------------------Метод получения шинглов-----------------------------------*/
     private List<string> GetShingles(List<string> words, int count = 10)
        {
            List<string> shingles = new List<string>();//список содержит строку из 10 слов
                                                        //без пробелов
            for (int i = 0; i < words.Count - count + 1; i++) //после каждой итерации сдвиг
            {                                                 //на одно слово
                string shingle = "";
                foreach(var w in words.Skip(i).Take(count))
                {
                    shingle += w;
                }
                shingles.Add(shingle);
            }
            return shingles;
        }
        /*Example:
        count = 2;
        TEXT: сборник задач физике старшей школы
        i = 1: shingle = сборникзадач
        i = 2: shingle = задачфизике ... etc*/
/*-------------------------------------------------------------------------------------------*/
/*-----------------------------Вычисление хеш-значений для каждой строки----------------*/
        private List<List<int>> CalcHashes(List<string> shingles, int countHashes = 36)
        {
            List<List<int>> hashes = new List<List<int>>();
            for (int i = 0; i < countHashes; i++)
            {
                hashes.Add(new List<int>());
                foreach(var shingle in shingles)
                {
                    hashes[i].Add(HashFunc(shingle, i));//возвращаемое хеш-значение добавим в
список
                }
            }
            return hashes;
        }
/*-------------------------------------------------------------------------------------------*/
/*-------------------------Поиск строки с минимальным весом------------------------------*/
        private List<int> GetMinHashes(List<List<int>> hashes)
        {
            List<int> minHashes = new List<int>();
            int hashNum = 0;
            foreach (var hash in hashes)
            {
                var min = hash.First();
                int k = 0;
                int shinMin = 0;

                foreach (var sh in hash)
                {
                    if (min > sh)
                    {
                        min = sh;
                        shinMin = k;
                    }
                    k++;
                }
```

```
                Console.WriteLine("Hash #" + hashNum + " - Shingle #" + shinMin);
                hashNum++;
                minHashes.Add(min);
            }
            return minHashes;
        }
/*-------------------------------------------------------------------------------------------*/
/*--------------------Вычисление хеш-значения-------------------------------------------------*/
        private int HashFunc(string shingle, int num)
        {
            num += 84;
            return Hash(shingle, 2, Constants.SimpleNumbers[num]);
        }

        private int Hash(string shingle, int p, int mod)
        {
            int hash = (int)shingle[0];
            int m = 1;
            for (int i = 1; i < shingle.Length; i++, m*=p)
            {
                hash = (hash * p) % mod + (int)shingle[i]; //по схеме Горнера посчитаем значение
в точке за линейное время от max степени
            }
            return hash % mod;//уменьшение по модулю?
        }
//посимвольно.
//Например: Строка: AAA . p=2, mod = 439 – простое число.
//для русской (int)A = 1079, тогда hash = (1079*2)%439 + 1079 и т.д. для каждого следующего
символа
/*-------------------------------------------------------------------------------------------*/
    }
```

Листинг 1.3. Класс с константами

```
public static class Constants
    {
        public static int[] SimpleNumbers =
        {
            2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79,
83, 89, 97, 101, 103, 107, 109, 113,
            127, 131, 137, 139, 149,
151,157,163,167,173,179,181,191,193,197,199,211,223,227,229,233,239,241,251,257,263,269,271,277,2
81,

283,293,307,311,313,317,331,337,347,349,353,359,367,373,379,383,389,397,401,409,419,421,431,433,4
39,

2861,2879,2887,2897,2903,2909,2917,2927,2939,2953,2957,2963,2969,2971,2999,3001,3011,3019,3023,30
37,3041,3049,3061,3067,3079,

3083,3089,3109,3119,3121,3137,3163,3167,3169,3181,3187,3191,3203,3209,3217,3221,3229,3251,3253,32
57,

3259,3271,3299,3301,3307,3313,3319,3323,3329,3331,3343,3347,3359,3361,3371,3373,3389,3391,3407,34
13,

3433,3449,3457,3461,3463,3467,3469,3491,3499,3511,3517,3527,3529,3533,3539,3541,3547,3557,3559,35
71,
        };
        public static string[] StopWords =
        {
            "они", "себе", "сам", "ее", "его", "им", "был", "были",
            "быть", "есть", "", "его", "себя", "я", "ты", "все",
            "мы", "это", "как", "так", "и", "в", "над", "к", "до",
            "не", "на", "но", "за", "то", "с", "ли", "а", "во", "от",
            "со", "для", "о", "же", "ну", "вы", "бы", "что", "кто", "он", "она"
        };
        public static char[] StopSymbols =
```

```
        {
            '.', ',', '!', '?', ':',
            ';', '-', '\n', '\r', '(',
            ')', '{', '}', '[', ']',
            '"', ' '
        };

    }
```

Листинг 1.4. Класс для вывода результата

```
class TextComparisonEngine
    {
        private ShinglesComparer ShComp;
        public TextComparisonEngine()
        {
            this.ShComp = new ShinglesComparer(84, 10);
        }

        public void Compare(string textA, string textB)
        {
            ShComp.ProcessTexts(textA, textB);
        }

        public void PrintResults()
        {
            Console.WriteLine("");
            Console.WriteLine("Shingles engine preferences: ");
            Console.WriteLine("\tHashCount==" + ShComp.HashCount);
            Console.WriteLine("\tShingleSize==" + ShComp.ShingleSize);
            Console.WriteLine("");
            Console.WriteLine("Results:");
            Console.WriteLine("\tPercentage of matching shingles:  " + ShComp.Result*100 + "%");
            Console.WriteLine("");
        }
    }
```

2. Тестирование:

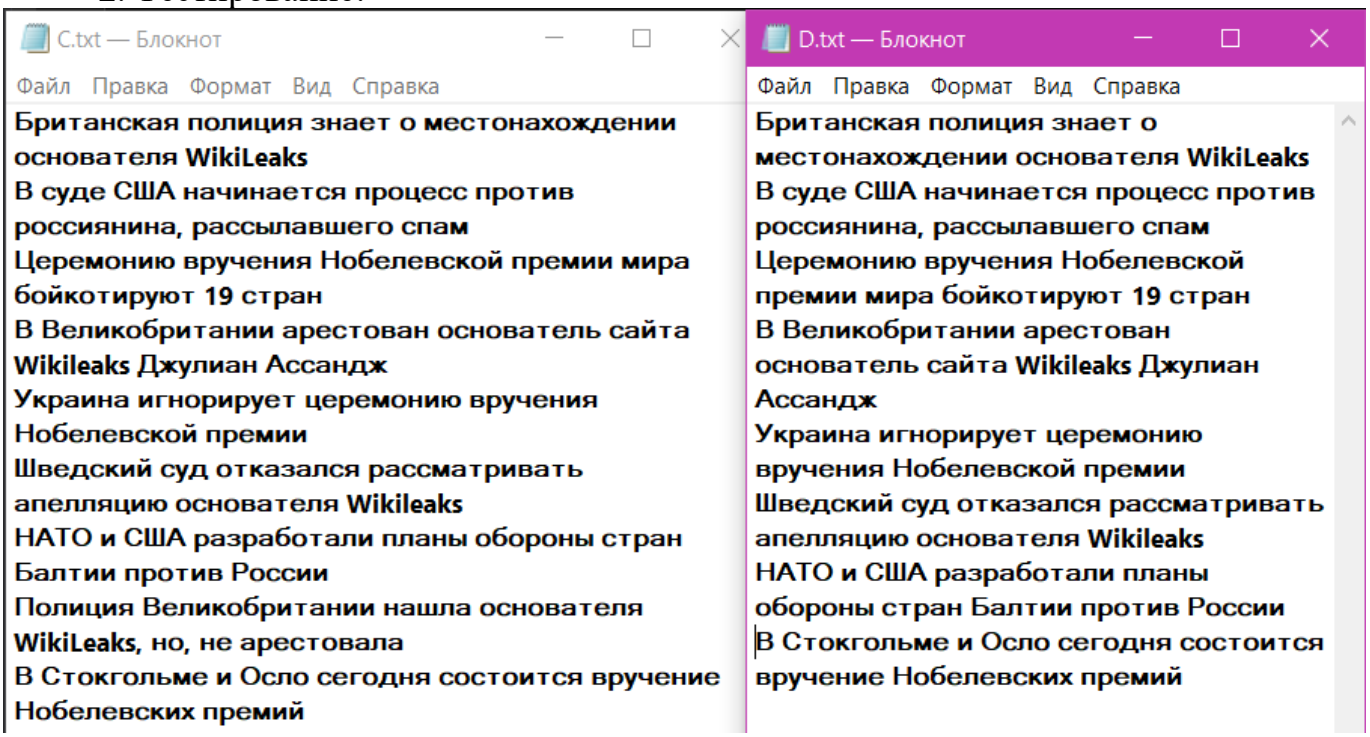

Рис. 5.1. Содержимое корпуса документов

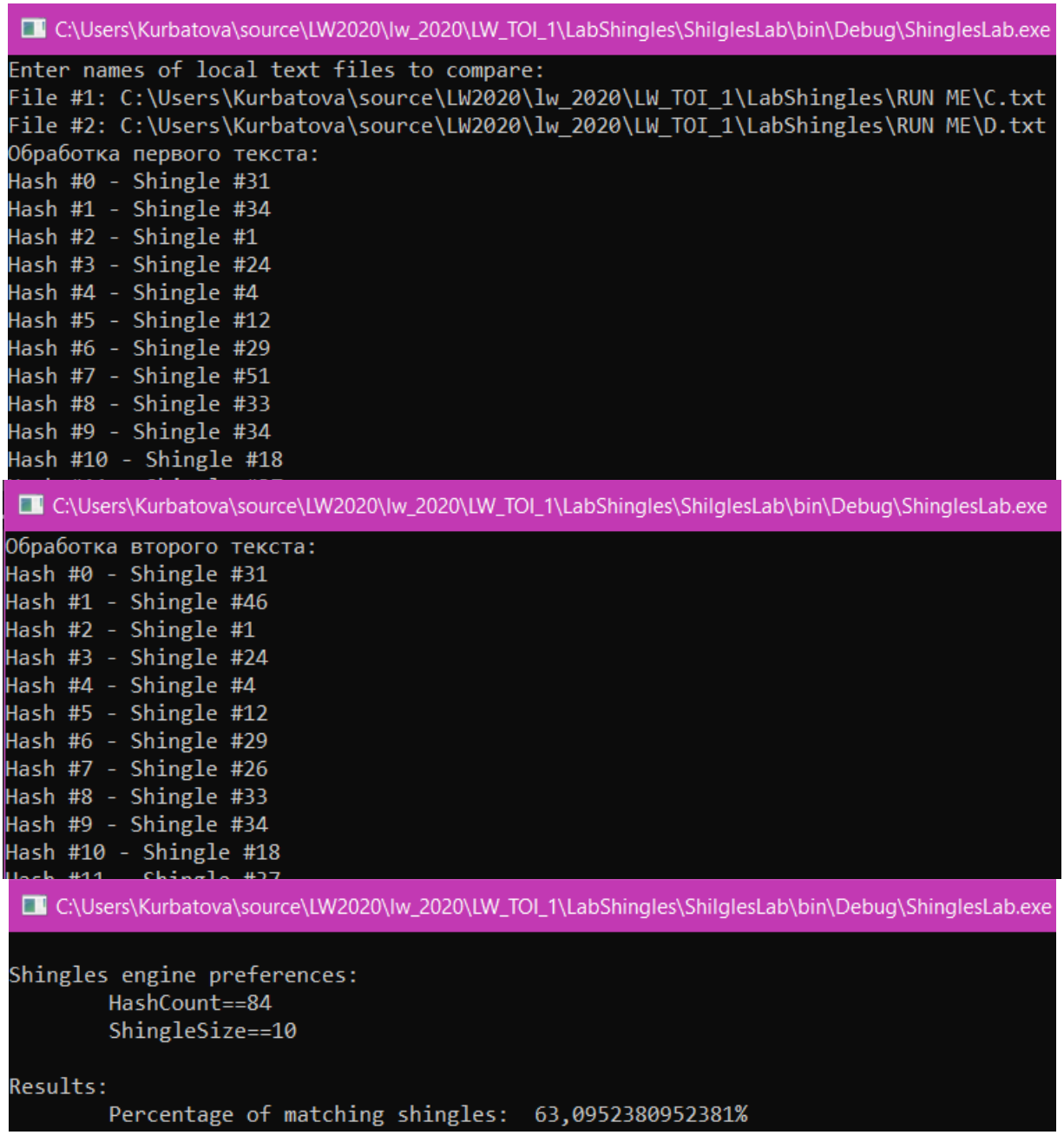

Рис. 5.2. Результат работы

Вывод: Таким образом, в ходе выполнения лабораторной работы на языке C# был реализован алгоритм обнаружения нечетких дубликатов с использованием алгоритма шинглов.