УДК 025.4.03

ИНФОРМАЦИОННО-ПОИСКОВАЯ СИСТЕМА «ПЛАГИАТ»

А.А.ТОРОЕВ, Д.В.БОРЩЕВА
E.mail. ksucta@elcat.kg

Бул макала тексттеги таасын эмес көчүрмөлөрдү издөөнүн информациялык издөө системасын иштеп чыгуу проблемасына арналган. Тексттеги көчүрмөлөрдү издөө үчүн «шинглов» алгоритмин колдонуу менен системаны түзүү процесси баяндалат. Макала эки бөлүккө бөлүнө алат. Биринчи бөлүктө «шинглов» алгоритминин иш жүргүзүү принциби сүрөттөлөт. Сүрөттөлүү мазмунуна таасир бербей турган сөздөн текстти тазалоо методикасынан, текстти саптарга бөлүү, CRC32 алгоритми боюнча контролдук суммасын аныктоо, жыйынтыгын алуудан башталат. Тексттин акыркы бөлүгү маалымат системасынын долбоорлоо процессин, «шинглов» алгоритминин бүт пункттарын иш жүзүнө ашуусун козгойт.

Статья посвящена проблеме разработки информационно поисковой системы поиска не-четких дубликатов в тексте. Описывает процесс построения системы с использованием алгоритма «шинглов» для поиска дубликатов в тексте. Статья может быть разделена на 2 части. В первой части описывается принцип работы алгоритма «шинглов». Описание начинается с методики очистки текста от слов, не влияющих на смысловую нагрузку, разбивка текста на подстроки внахлест, определения контрольных сумм по алгоритму CRC32, получение результата. Последняя часть текста затрагивает процессы проектирования информационной системы, реализацию всех пунктов алгоритма «шинглов».

Article is devoted a working out problem is information search system of search not-accurate duplicates in the text. Describes process of construction of system with algorithm use «Shinglov» for search of duplicates in the text. Article can be divided into 2 parts. In the first part the principle of work of algorithm «Shinglov» is described. The description begins with a technique of clearing of the text from the words which are not influencing semantic loading, text breakdown on подстроки внахлест, definitions of the control sums on algorithm CRC32, reception result. Last part of the text mentions processes of designing of information system, realisation of all points of algorithm «Shinglov».

Современный этап развития цивилизации характеризуется переходом наиболее развитой части человечества от индустриального общества к информационному. Одним из ярких явлений этого процессов является вхождение компьютера и компьютерных технологий в разные сферы человеческой деятельности и развитие глобальной информационной компьютерной сети. Вхождение компьютерных технологий дает возможность управлять информацией. С каждым годом глобальная сеть растет невиданными темпами, информация накапливается, и сегодня любой пользователь может получить любую информацию. И тут вот возникает большая проблема – проблема плагиата или авторства текста /1/.

Плагиат – умышленное присвоение авторства чужого произведения науки или мыслей, или искусства, или изобретения. Плагиат может быть нарушением авторско-правового законодательства и патентного законодательства и в качестве таковых может повлечь за собой юридическую ответственность.

Наиболее часто плагиат выражается в публикации под своим именем чужого произведения или чужих идей, а также в заимствовании фрагментов чужих произведений без указания источника заимствования. Обязательным признаком плагиата является присвоение авторства, так как неправомерное использование, опубликование, копирование и т.п. произведения, охраняемого авторским правом, само по себе является не плагиатом, а пиратством. Пиратство становится плагиатом при неправомерном использовании результатов интеллектуального труда и присвоении публикующим лицом

авторства. В данной статье будем рассматривать тему поиска плагиата как поиска нечетких дубликатов текста.

Алгоритм Шинглов

Для решения задачи поиска нечетких дубликатов Udi Manber (Уди Манбер) в 1994 году предложил идею, а Andrei Broder (Андрей Бродер) в 1997 придумал название и доработал **алгоритм «шинглов»** (от слова shingles, «черепички, чешуйки»). Поиск нечетких дубликатов позволяет предположить, являются ли два объекта частично одинаковыми или нет /2/. Под объектом могут пониматься текстовые файлы и другие типы данных. Задачей не стоит определить абсолютное значение схожести объектов, а также выделения в каждом из объектов схожих частей. Необходимо только предположить, являются ли объекты почти дубликатами или нет.

Как работает алгоритм Шинглов?

В публикации об алгоритмах поиска почти дубликатов предлагается алгоритм шинглов, использующий случайную выборку 84-х случайных шинглов. Почему именно 84? Использование 84-х случайно выбранных значений контрольных сумм позволит легко модифицировать алгоритм до алгоритма супершинглов и мегашинглов, которые гораздо менее ресурсоемки.

Этапы алгоритма:

1) канонизация текста;
2) разбиение на шинглы;
3) вычисление хэшей шинглов с помощью 84-х статических функций;
4) случайная выборка 84 значений контрольных сумм;
5) сравнение, определение результата.

1. Канонизация текста

Канонизация текста приводит оригинальный текст к единой нормальной форме.

Текст очищается от предлогов, союзов, знаков препинания, HTML тегов, и др., которые не должны участвовать в сравнении. В большинстве случаев также предлагается удалять из текста прилагательные, так как они не несут смысловой нагрузки. Также на этапе канонизации текста можно приводить существительные к именительному падежу, единственному числу либо оставлять от них только корни /3/.

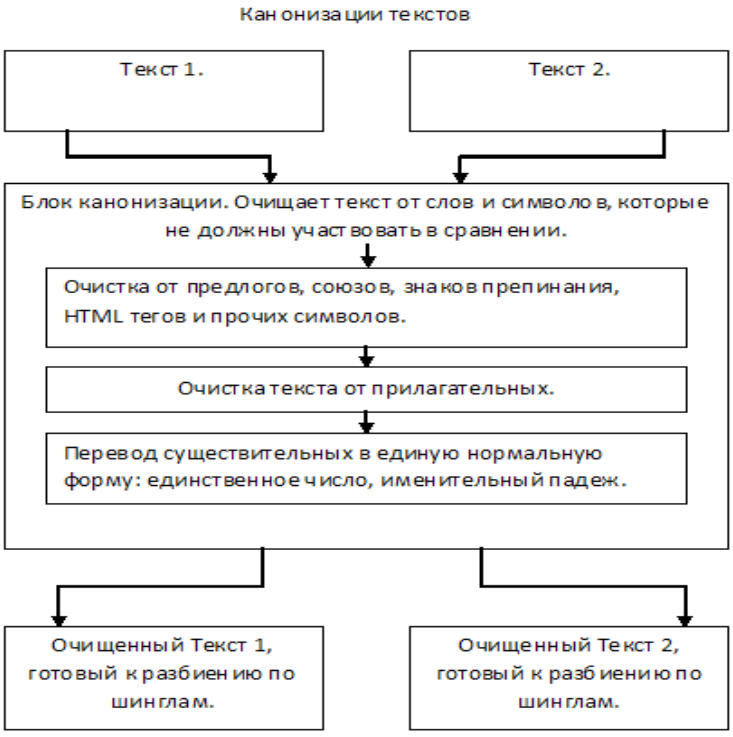

Рис.1. Канонизация текстов

2. Разбиение на шинглы

Шинглы (англ) – чешуйки, выделенные из текста подпоследовательности слов. Необходимо из сравниваемых текстов выделить подпоследовательности слов, идущих друг за другом по 10 штук (длина шингла). Выборка происходит внахлест, а не встык. Таким образом, разбивая текст на подпоследовательности, получаем набор шинглов в количестве, равном количеству слов минус длина шингла плюс один (кол-во_слов – длина_шингла + 1). Если запоминать контрольные суммы для строчек фиксированной длины, идущих встык, то вставка и удаление одного символа (особенно в начале текста) разрушит их все. Однако если отменить фиксацию длины и считать подстрочки от одной характерной точки в тексте до другой (например, от буквы “ю” до буквы “ю”, или от двухбуквия, сумма численных значений символов (букв) которого кратна 50, до следующего такого же), вставка (или удаление) с большой вероятностью разрушит только тот шингл, где она случилась. Поначалу, кажется, что считать контрольные суммы по всем строчкам внахлест – странная идея. Нам нужно сократить объем данных для сравнения, а в таком варианте он страшно возрастает? Однако именно так будет гарантия, что не пропускаем ни одной подстроки текста (заданной длины) и, при условии, что удастся придумать устойчивый способ отбирать шинглы, нам удастся очень точно отождествлять документы, имеющие совпадающие части /4/.

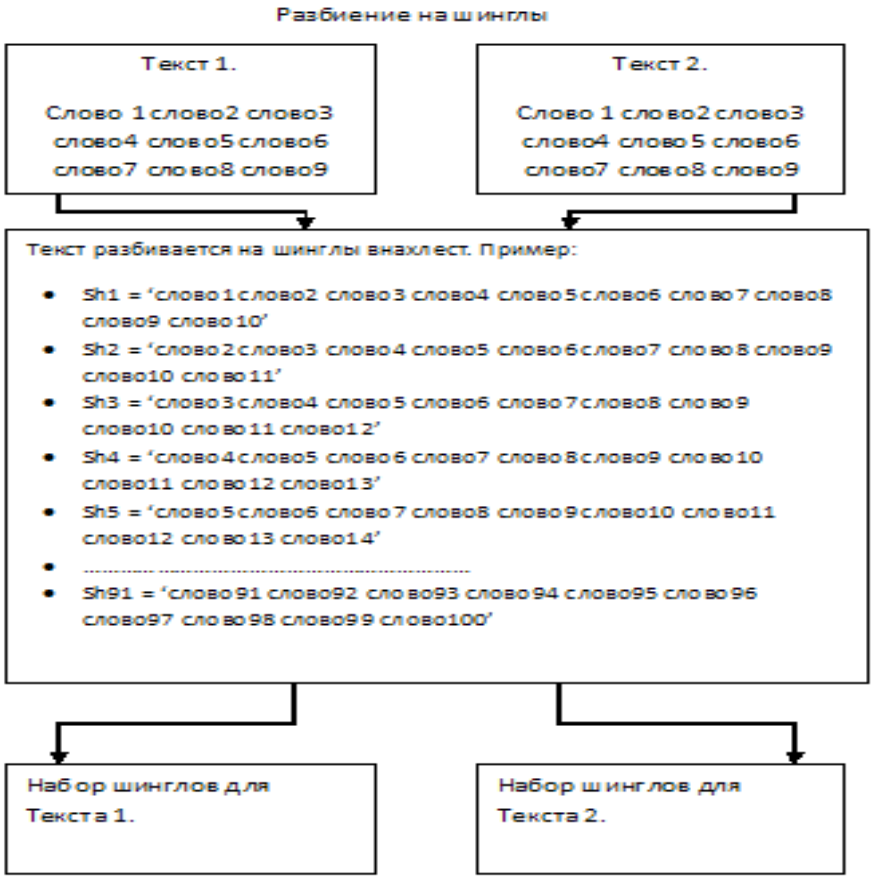

Рис. 2. Разбиение на шинглы

3. Вычисление хэшей шинглов с помощью 84-х статических функций

4.

Принцип алгоритма шинглов заключается в сравнении случайной выборки контрольных сумм шинглов (подпоследовательностей) двух текстов между собой.

Проблема алгоритма заключается в количестве сравнений, ведь это напрямую отражается на производительности. Увеличение количества шинглов для сравнения характеризуется экспоненциальным ростом операций, что критически отразится на производительности.

Предлагается представить текст в виде набора контрольных сумм, рассчитанных через 84 уникальные между собой статические хэш-функции /5/.

Для каждого шингла рассчитывается 84 значения контрольной суммы через разные функции (например, SHA1, MD5, CRC32 и т.д., всего 84 функции). Итак, каждый из текстов будет представлен в виде двумерного массива из 84-х строк, где каждая строка характеризует соответствующую из 84-х функций контрольных сумм.

Из полученных наборов будут случайным образом отобраны 84 значения для каждого из текстов и сравнены между собой в соответствии с функцией контрольной суммы, через которую каждый из них был рассчитан. Таким образом, для сравнения будет необходимо выполнить всего 84 операции.

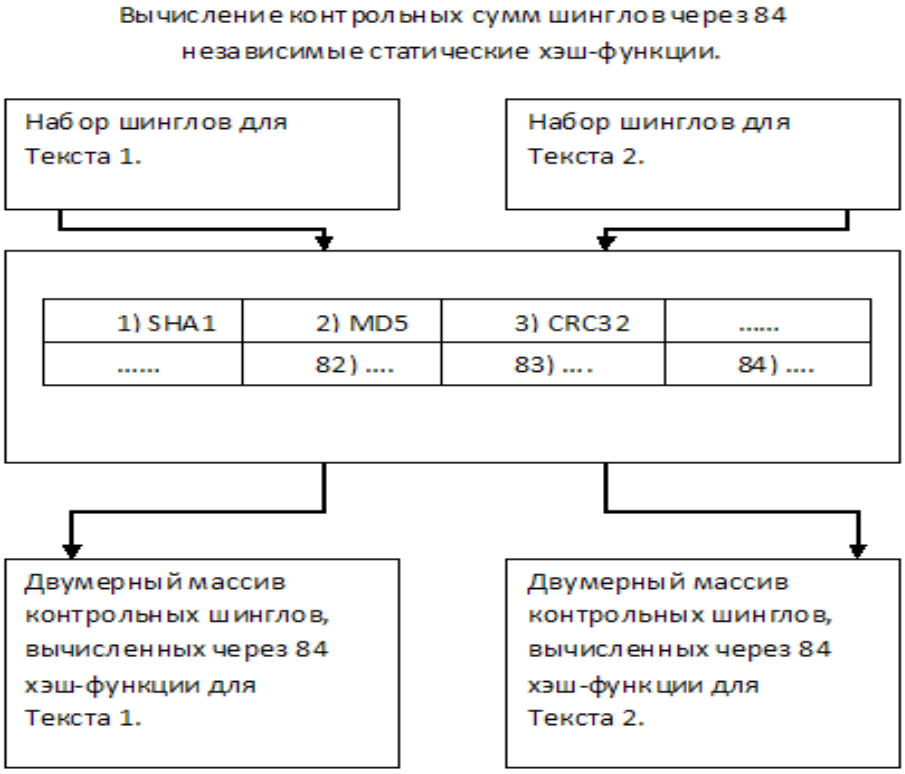

Рис. 3. Вычисление контрольных сумм

5. Случайная выборка 84 значений контрольных сумм

Сравнивать элементы каждого из 84-х массивов между собой – ресурсоемко. Для увеличения производительности выполним случайную выборку контрольных сумм для каждой из 84-х строк двумерного массива для обоих текстов. Например, будем выбирать самое минимальное значение из каждой строки. Итак, на выходе имеем набор из минимальных значений контрольных сумм шинглов для каждой из хэш-функций /6/.

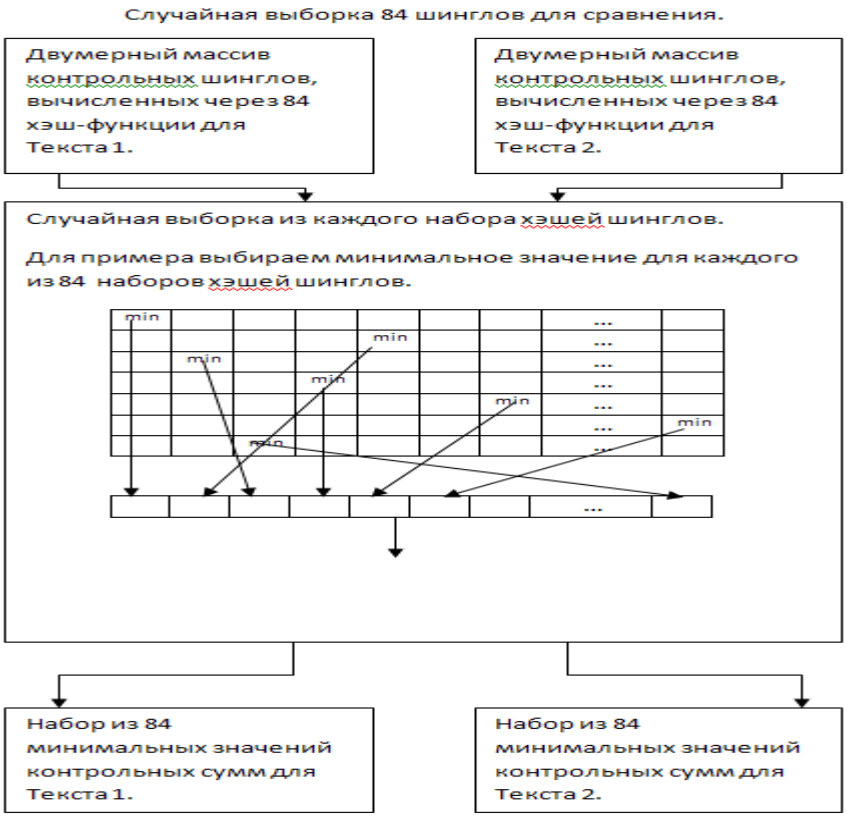

Рис. 4. Случайная выборка 84 шинглов для сравнения

6. Сравнение, определение результата

И последний этап – сравнение. Сравниваем между собой 84 элемента первого массива с соответствующими 84-мя элементами второго массива, считаем отношение одинаковых значений, из этого получаем результат.

Проект

Для решения поставленной задачи был разработан проект «Плагиат». В данной работе применен алгоритм шинглов в простой реализации, реализованный на Visual Studio 2005 на языке C#.

1) канонизация текстов;
2) разбиение текста на шинглы;
3) нахождение контрольных сумм;
4) поиск одинаковых значений контрольных сумм.

Разработана база на СУБД MySql Server 5.1. Она состоит из четырех таблиц:

- Таблица «**objects**» содержит данные по предметной области.
- Таблица «**writers**» содержит данные об авторе текста или ресурсе.
- Таблица «**texts**» содержит данные о тексте и немодифицированную версию текста:
- Таблица «**crc32doc**» заполнена значениями контрольных сумм текста.

При разработке проекта из-за частой работы с файлами был создан класс **WorkFile,** который содержит следующие функции:

1. **public string FileRead(string path)** – чтение текстов различных форматов.
2. **public string CreateDirectory(string path)** – создание папки с уникальным именем.
3. **public void FileCopy(string path1, string path2)** – копирование файла из path1 в path2 при условии, что path1 существует.

4. **public void DirectoryCopy(string path1, string path2)** – копирование всех файлов из папки path1 в папку path2.
5. **public void WriteFile(string put, string text, string textname)** –запись данных в файл /6/.

1. Канонизация текстов

Для решения данной задачи был создан класс **`WorkString`,** который содержит:

1. **public string RemoveHTML(string str)** –удаляет все html теги при загрузке текста с веб-страницы.
2. **public string RegexReplace(string text, string regextext, string replacetext) и public string RegexReplace(string text)** – функция для замены в тексте строки – регулярного выражения. В качестве параметров в первой передается текст, в котором нужно искать text (regextext – строка поиска, replacetext – заменить), а во второй передается текст, в котором нужно удалить конкретные символы.
3. **public int textsum(string text)** – подсчет количества слов в тексте**.**
4. **public string StopWordsClear(string text)** – удаление слов из текста из словаря dictionary.dic с использованием регулярных выражений**.** Отрывок из словаря:

```
----------
был(а|и|о)
быстр(ая|ее|о|ого|ое|ой|ому|ый|ым)
быть
в
важн(ая|о|ого|ое|ой|ом|ому|ые|ый|ее)
ваш(а|ей|ему|его|у)
введение
ввиду
------
```

Данные слова удаляются из текста, при этом сокращается размер обрабатываемого текста (например, 4000 слов в документе, после обработки около 3000-2500 слов).

5. **public string ReturnConfig()** – возвращает путь к системной папке программы.
6. **public string Remowe(string text) –** непечатных символов.

2. Разбиение текста на шинглы

Текст, очищенный на предыдущем этапе, разбивается на слова и размещается в массиве. Далее в двойном цикле берется один элемент из массива, прибавляются еще 9 элементов и записываются. При разбивке конца текста, если не хватает слов для составления шингла, то они отбрасываются (рис.1, 2).

Данный цикл из переменной массива shingl, который содержит все слова из текста, присваивает значения textshingl, потом прибавляет девять значений shingl.

3. Нахождение контрольных сумм и поиск одинаковых значений контрольных сумм

Для реализации алгоритмов создан класс **`Algorithm`**

1. **public int FindSubstring(string input, string pattern, int index) и для нее private int[] GetPrefixFunction(string pattern) –** нахождение в тексте начиная с индекса подстроки.
2. **public uint CalculateCRC(System.IO.Stream stream)** – из потока рассчитываем значение контрольной суммы по алгоритму CRC32, данный алгоритм выбран по его надежности и высокой скорости с полиномом 0xEDB88320 (зеркальное отображение полинома 0x04C11DB7) /7/.

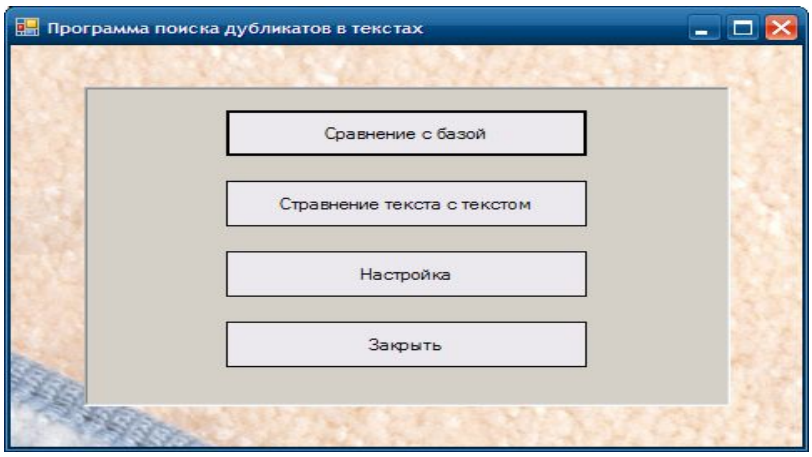

Рис. 5. Главная форма программы

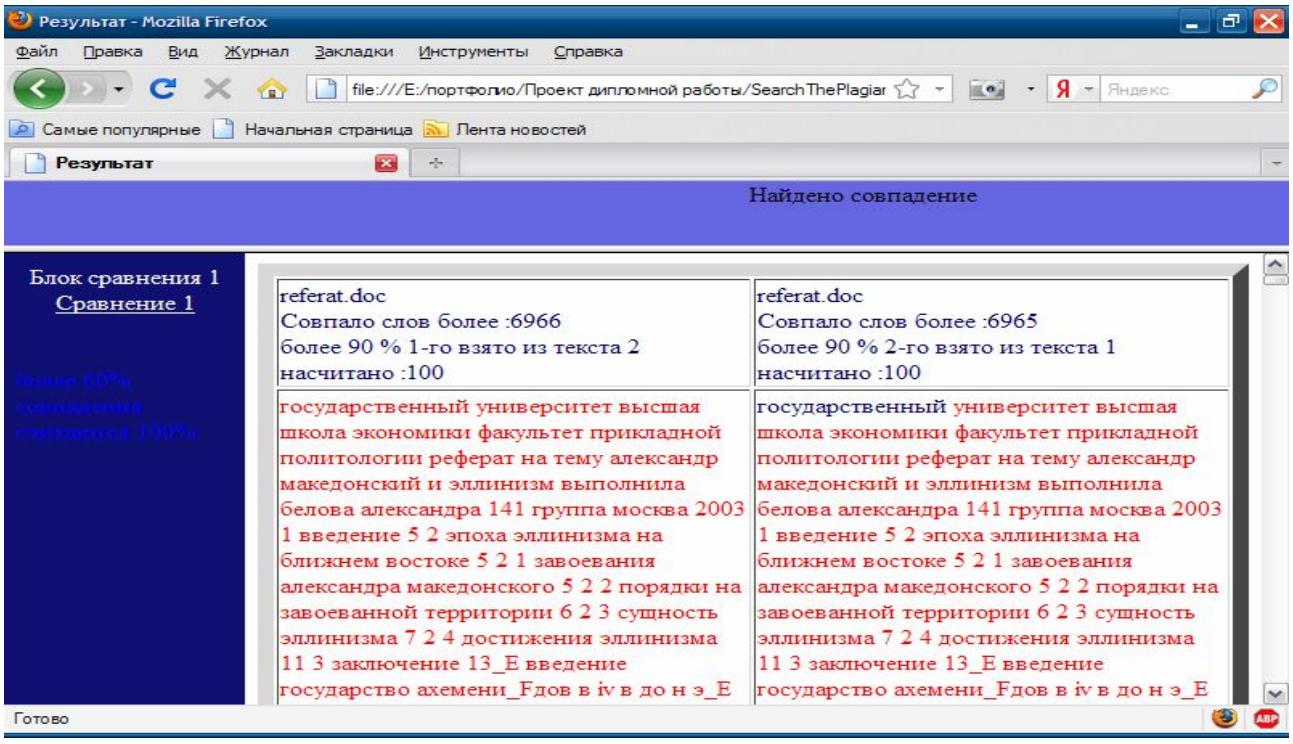

Рис. 6. Результат проверки текста с базой

Заключение

Рассмотренный проект еще не совершенен. Считается, что идеальная поисковая система должна отвечать следующим требованиям:

- простота в использовании;
- быстрый поиск в базе данных большого объема данных и быстрое реагирование;
- надежность и точность результатов поиска.

Используемый алгоритм показывает результаты, но является немного ресурсоемким. Для усовершенствования данного продукта необходимо улучшить этап канонизации текстов. В программный продукт необходимо включить анализатор смысла текста.

Список литературы

1. Обнаружение нечетких дубликатов документов. Алгоритмы http://www.seo-buster.ru/article/index13.html Беломестных Виктор Борисович.
2. Гайдамакин Н.А. Автоматизированные информационные системы, базы и банки данных. – М.: Гелиос, 2002.
3. Корнеев В.В., Гарев А.Ф., Васюшин СВ., Райх В.В. Базы данных. Интеллектуальная обработка информации. – М.: Нолидж, 2000.
4. Руководство http://www.opennet.ru/docs/RUS/mysqlserver_guide/ Установка и конфигурирование сервера MySQL
5. Проверка контента на плагиат – Сервисы, программы, алгоритмы - Блог SEO негоцианта.htm http://blog.negotiant.org/proverka-kontenta-na-plagiat-servisy-programmy-algoritmy/
6. Как обмануть антиплагиат. htm http://www.zavtrasessiya.com/articles/article10.html
7. Отлавливать студенческий плагиат будет специальная программа www.newsru.com/russia/31may2006/anti.html