УДК 004

Реализация алгоритма локального антиплагиата с помощью языка C++

Ленкин Алексей Викторович
Приамурский государственный университет имени Шолом-Алейхема
студент

Кизянов Антон Олегович
Приамурский государственный университет имени Шолом-Алейхема
студент

Лучанинов Дмитрий Васильевич
Приамурский государственный университет имени Шолом-Алейхема
старший преподаватель кафедры информационных систем, математики и методик обучения

Аннотация
В данной статье рассмотрен алгоритм локального определения уникальности текста. Разработанный алгоритм предназначен для сравнения пары документов: эталонного и опытного.
Ключевые слова: Антиплагиат, язык программирования C++, информация, уникальность, алгоритм.

The local anti-plagiarism algorithm implementation with C++ language

Lenkin Aleksei Viktorovich
Sholom-Aleichem Priamursky State University
student

Kizyanov Anton Olegovich
Sholom-Aleichem Priamursky State University
student

Luchaninov Dmitry Vasilyevich
Sholom-Aleichem Priamursky State University
Senior lecturer of the Department of Information Systems, Mathematics and teaching method

Abstract
The article describes an algorithm for local text uniqueness determining. The algorithm is designed for comparing pairs of documents: a reference and experienced.
Keywords: Anti-plagiarism, the programming language C ++, information unique.

В современное время свободный доступ к сети Интернет есть почти у каждого. Как правило, это делает любую информацию доступной и открытой общественности, но также приносит и новое понятие – информационное хищение, когда материалы, созданные одним человеком, используются без его согласия другими. В целях предотвращения данного процесса были разработаны специальные алгоритмы для поиска информационного плагиата, которые, используя специальные формул, проверяют, сколько в проверяемых материалах заимствованных данных и выводят результат, опираясь на который можно делать вывод об уникальности. В данной статье описано исследование процесса поиска плагиата и разработка собственного алгоритма проверки электронных документов.

На данный момент создано уже довольно много программ поиска плагиата в электронных документах, наиболее часто используемая программа принадлежит ЗАО “Анти-Плагиат”. Данный продукт широко используется во всех образовательных организациях высшего образования страны для проверки работ студентов, а также научных работ. Алгоритм проверки плагиата хранится компанией в секрете, но общая методика проверки известна [1]:

1. Система собирает информацию из различных источников: загружает из сети Интернет и обрабатывает сайты, находящиеся в открытом доступе, базы научных статей и рефератов. Загруженные документы проходят процедуру фильтрации, в результате которой отбрасывается бесполезная (с точки зрения потенциального цитирования) информация.

2. На следующем этапе каждый из полученных таким образом текстов определённым образом форматируется и заносится в системную базу данных.

3. Все пользовательские документы, загружаемые для проверки, ставятся в очередь на обработку.

4. После проверки документа, пользователь получает доступ к отчёту, в котором представляются результаты.

Данная система во многом и хороша и используется уже достаточный промежуток времени, но она имеет некоторые недостатки [2]:

1. С каждым улучшением программы студенты придумывают новые способы обмана системы. К примеру, раньше было возможно заменить буквы кириллицы на аналогичные латиницы и программа считала такие слова разными.

2. Системе обязательно нужен выход в сеть Интернет для проверки документа, чтобы сверить его с базой данных.

3. Даже если выход в сеть Интернет есть, проверка электронного документа может затянуться на долгое время, причем дело не в размере проверяемого файла, в обычной (бесплатной) версии на пользователя накладываются некоторые ограничения, а именно: 1 документ на 6 минут и низкий приоритет в очереди на проверку.

На сегодняшний день существует множество алгоритмов для поиска плагиата, которые используются повсеместно. Перечислим некоторые из

них[3]: проверка документа дословным перекрытием текста, анализ «множества слов», цитирование, стилометрия и т.д. В настоящее время наиболее распространённым подходом является Дактилоскопия [3]: Из ряда документов выбирается набор из нескольких подстрок, которые и являются «отпечатками». Рассматриваемый документ будет сравниваться с «отпечатками» для всех документов коллекции. Найденные соответствия с другими документами указывают на общие сегменты текста.

Мной был написан ещё один метод (программно реализованный в предыдущих исследованиях [4]), который отчасти является улучшенной версией линейного алгоритма [5] (сравнение по одному слову из двух документов от начала до конца). Алгоритм написан на языке C++ и включает в себя две основные функции: стандартизация текста (подготовка к проверке) и проверка двух документов на процент заимствования.

Итак, подробней о каждой функции:

Стандартизация текста:

1. Получает на входе документ и записывает его в строковый массив.

2. Ищет в массиве от начала до конца знаки препинания, цифры, специальные символы и удаляет их.

3. Проходи массив строк ещё раз и переводит все буквы в нижний регистр.

4. Сохраняет полученный массив как текстовый файл возле оригинала.

Код, отвечающий за предварительную стандартизацию текста:

```
        setlocale(0, «»);
        string tmp=n;
        tmp+=«.txt»;
        ifstream x(tmp.c_str());
  char bad_chars[] = «\\\%0123456789,.?!:;\»\(\)-+=-_…[]»;
  int c=0;
  string q;
  string A = «АБВГДЕЁЗЖИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ»;
  string a = «абвгдеёзжийклмнопрстуфхцчшщъыьэюя»;
  while (!x.eof())
  {  string buf;
     getline(x, buf);
     q.append(buf); }
        x.close();
        x.clear();
  for (unsigned i = 0; i < q.size(); ++i)
  {  int pos;
     if ((pos = q.find_first_of(bad_chars)) != string::npos)
     {    q.erase(pos, 1);
          c++;      }
     if (q[i]==' '&&q[i-1]==' ')
     {    q.erase(i, 1);
          c++;      }
     if (A.find(q[i]) !=-1)
     {    int b = A.find(q[i]);
          q[i] = a[b];   }
     if (c > 0)
```

```
    {     c = 0;
          i--; }}
        string tmr=n;
        tmr+=«1.txt»;
ofstream out;
        out.open(tmr.c_str());
out << q << endl;
        tmp.clear();
        q.clear();
        tmr.clear();
out.close();
        out.clear();
```

Проверка двух документов на процент заимствования:

1. Полученные из предыдущей функции стандартизированные файлы записываются в два строковых массива.

2. Массивы помещаются в цикл, в котором берётся слово из первого файла и сверятся со вторым, если слова различные, то берётся следующее. Действие проходит до тех пор, пока не достигнет конца документа (тогда пункт 2 повторяется ещё раз, но теперь берётся следующее слово из первого документа) или не обнаружена схожесть (тогда переходит к пункту 3).

3. Если найдено похожее слово, то проходит проверка строки уже из двух слов в том же месте, количество слов в строке на проверку увеличивается до тех пор, пока они не станут различными.

4. Если количество слов в сформировавшейся в 3 пункте строке более 5, то число этих слов прибавляется к общему числу найденных повторов. После пункт 2 повторятся, но уже с места “Последнее повторившееся слово” +1. Проверка проходит, пока не достигнут конец обоих документов.

5. Высчитывается процент заимствования по формуле «(количество найденных повторов/число слов в документе)*100».

Код, чтения проверяемого файла:

```
string s;
int a=0;
ifstream in(n);
while (!in.eof())
{  in >> s;
   a++; }
in.clear();
in.seekg(0,ios::beg);
string *g = new string [a];
int b = 0;
while (!in.eof())
{  in >> g[b];
   b++; }
in.close();
        in.clear();
```

Код чтения эталонного файла:

```
a = 0;
ifstream ideal(f);
while (!ideal.eof())
{  ideal >> s;
   a++; }
ideal.clear();
ideal.seekg(0, ios::beg);
string *t = new string[a];
int c = 0;
while (!ideal.eof())
{  ideal >> t[c];
   c++; }
ideal.close();
      ideal.clear();
```

Код определения оригинальности группы слов:

```
s.clear();
int d=0;
int k,j,i;
i = 0;
j = 0;
k = 0;
while (i<b)
{ for (j = 0; j < c;j++)
   { if (strcmp(g[i].c_str(), t[j].c_str())==0)
        { d++; i++; }
        else { if (d != 0)
             { k += d; if (d < 4)
                  { i -= d; k -= d; }
                  d = 0;
   }    }    }
   i++; }
```

Код вычисления оригинальности:

```
float e, r;
e = k;
r = b;
float h = (e / r) * 100;
return h;
```

В итоге, в совокупности эти две функции могут стать частью программного обеспечения для поиска плагиата в локальном режиме [3] (см. рис. 1).

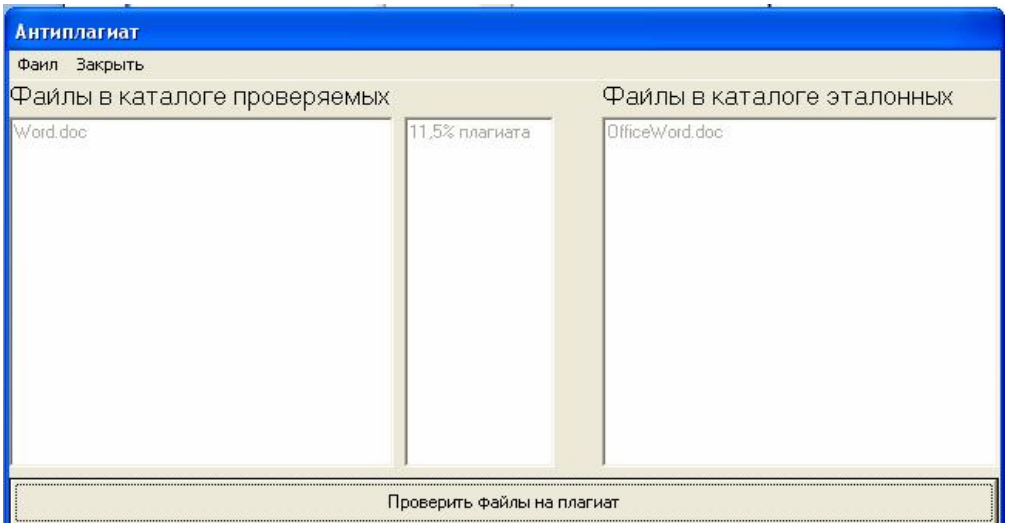

Рисунок 1. Программа «Локальный антиплагиат»

Алгоритм достаточно прост, но имеет в себе несколько небольших недостатков:

1. Время выполнения сильно зависит от размера проверяемых документов. К примеру, можно потерять много времени, если он попытается найти слово в документе из 10000 слов, где такого слова нет.

2. Возможны ложные срабатывания на общие фразы из 5 и более слов. К примеру, «Для меня эта тема является ключом к пониманию того, что…», если эта фраза будет найдена в двух документах, то будет засчитана как плагиат, хотя таковой не является.

Библиографический список

1. Против рейдерства в науке [Электронный ресурс]. URL: http://www.chaskor.ru/article/protiv_rejderstva_v_nauke_32043 (дата обращения 24.01.2016).
2. Система Антиплагиат [Электронный ресурс]. URL: http://www.antiplagiat.ru/ (дата обращения 24.01.2016).
3. Ленкин А.В. Колесников А.А. Лучанинов Д.В. Реализация локального антиплагиата с помощью объектного приложения языка C++ // Постулат. 2015. №2. [Электронный ресурс] URL: http://e-postulat.ru/index.php/Postulat/article/view/21 (дата обращения 24.01.16)
4. Дягилев В.В., Цхай А. А., Бутаков С. В. Архитектура сервиса определения плагиата, исключающая возможность нарушения авторских прав (рус.) // Вестник НГУ. Серия: Информационные технологии. 2011. Т.9. №3. С.23-29
5. Алгоритмы поиска [Электронный ресурс]. URL: http://studlab.com/news/algoritmy_poiska/2011-05-31-110 (дата обращения 24.01.2016).