Aplikasi Pendeteksi Penjiplakan pada File Teks dengan Algoritma Winnowing

Putu Yuwono Kusmawan, Umi Laili Yuhana, Diana Purwitasari
Jurusan Teknik Informatika,Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Email: putuyuwono@cs.its.ac.id, yuhana@cs.its.ac.id, diana@if.its.ac.id

Abstrak

Penjiplakan merupakan masalah yang semakin berkembang khususnya dalam dunia pendidikan. Banyak sekali karya-karya tulis mahasiswa, misal: proposal tugas akhir, tugas kuliah dll, yang sebagian kecil atau besar isinya dibuat dengan menjiplak milik orang lain. Berangkat dari permasalahan tersebut dibangunlah sebuah aplikasi pendeteksi penjiplakan dalam file teks yang diharapkan mampu membantu mempermudah pendeteksian penjiplakan dalam file teks meskipun jumlah file yang terlibat cukup banyak.

Aplikasi yang dibuat dalam tugas akhir ini mampu mendeteksi penjiplakan dalam file teks secara one-to-one, one-to-many, bahkan many-to-many. Dalam mendeteksi kesamaan antar file teks, digunakan sebuah algoritma yang disebut Winnowing Algorithm.Algoritma ini berfungsi untuk melakukan proses document fingerprinting, yang mengubah teks menjadi sekumpulan nilai-nilai hash. Aplikasi ini menggunakan database yang berupa file berformat XM,L agar dapat berjalan tanpa memerlukan konfigurasi database yang rumit.

Kata kunci: *Winnowing Algorithm, document fingerprinting,* XML.

1. Pendahuluan

Dalam era yang serba cepat dan instan ini, murid/mahasiswa ingin segera menyelesaikan tugas tanpa perlu bersusah payah untuk mempelajarinya. Cukup dengan menggunakan teknik copy and paste tugas milik teman, tugas-tugas sekolah atau kuliah dapat terselesaikan. Selain itu bagi mahasiswa yang sedang menyusun proposal Tugas Akhir terkadang kerap kali melakukan praktik penjiplakan untuk mengisi bagian pendahuluan, latar belakang, dll, karena ingin proposalnya segera selesai tanpa perlu bersusah payah untuk mengeksplorasi materinya terlebih dahulu.

Meskipun telah berulang kali diberi peringatan, tetap saja praktik penjiplakan dalam tugas bahkan ujian dilakukan oleh murid/mahasiswa. Hal ini menimbulkan masalah dalam tahap evaluasi terhadap hasil pembelajaran murid/mahasiswa. Tidak mudah bagi seorang pengajar untuk memberikan penilaian objektif terhadap tugas atau ujian yang telah diberikan kepada mahasiswa. Akan semakin bertambah parah apabila murid/mahasiswa yang mengikuti perkuliahan tersebut jumlahnya sangat banyak. Kemungkinan terjadinya penjiplakan akan semakin besar dan akan semakin sulit mendeteksinya mengingat begitu banyak file-file jawaban yang terlibat.

Oleh karena itu perlu dibangun sebuah sistem yang dapat mendeteksi penjiplakan pada beberapa file sekaligus, untuk mempermudah pekerjaan guru/dosen. Sistem ini diharapkan mampu mendeteksi penjiplakan dengan sistem one-vs-many comparing (pembandingan satu ke banyak) sehingga dapat digunakan untuk menganalisa file-file jawaban tugas/ujian yang berjumlah banyak.

2. Deteksi Penjiplakan

Penjiplakan menurut Kamus Besar Bahasa Indonesia berarti menggambar atau menulis garis-garis gambaran atau tulisan yg telah tersedia (dengan menempelkan kertas kosong pada gambar atau tulisan yg akan ditiru), mencontoh atau meniru tulisan atau pekerjaan orang lain, mencuri karangan orang lain dan mengakui sebagai karangan sendiri, mengutip karangan orang lain tanpa seizin penulisnya[2].

Banyak cara atau metode yang dapat digunakan untuk mendeteksi penjiplakan dalam file text. Namun ada kebutuhan mendasar yang harus dipenuhi oleh algoritma deteksi penjiplakan[1], yaitu:

Whitespace Insensitivity yang berarti dalam melakukan pencocokan terhadap file teks seharusnya tidak terpengaruh oleh spasi, jenis huruf (kapital atau normal), tanda baca dan sebagainya.

Noise Surpression yang berarti menghindari penemuan kecocokan dengan panjang kata yang terlalu kecil atau kurang relevan, misal: *'the'*. Panjang kata yang ditengarai merupakan penjiplakan harus cukup untuk membuktikan bahwa kata-kata tersebut telah dijiplak dan bukan merupakan kata yang umum digunakan.

Position Independence yang berarti penemuan kecocokan / kesamaan harus tidak bergantung pada posisi kata-kata. Meskipun posisnya tidak sama, kecocokan harus dapat ditemukan.

Sebuah karya tulis dikatakan telah menjiplak karya lain apabila memiliki tingkat kesamaan yang melebihi batas toleransi tertentu yang telah ditentukan.

3 Algoritma Winnowing

Winnowing adalah algoritma yang digunakan untuk melakukan proses *document fingerprinting*[1]. Proses ini ditujukan agar dapat mengidentifikasi penjiplakan, termasuk bagian-bagian kecil yang mirip dalam dokumen yang berjumlah banyak.

Input dari proses *document fingerprinting* adalah file teks. Kemudian *output*-nya akan berupa sekumpulan nilai *hash* yang disebut *fingerprint*. *Fingerprint* inilah yang akan dijadikan dasar pembanding antar file-file teks yang telah dimasukkan. Algoritma yang digunakan untuk mencari nilai *hash* dalam *winnowing* adalah *rolling hash*.

Salah satu prasyarat dari algoritma deteksi penjiplakan adalah *whitespace insensitivity*, *Winnowing* telah memenuhi prasyarat tersebut dengan cara membuang seluruh karakter-karakter yang tidak relevan misal: tanda baca, spasi dan juga karakter lain, sehingga nantinya hanya karakter-karakter yang berupa huruf atau angka yang akan diproses lebih lanjut.

Langkah-langkah

Berikut ini adalah langkah-langkah yang harus di lakukan untuk menerapkan algoritma *Winnowing* dalam melakukan proses *document fingerprinting*.

Tahap pertama adalah pembuangan karakter-karakter yang tidak relevan, misal: tanda baca, spasi, dll. Misal : *"nama saya putu"* akan diubah menjadi *namasayaputu*.

Tahap kedua adalah pembentukan rangkaian *gram* dari teks yang telah dibersihkan dengan ukuran 5: namas amasa masay asaya sayap ayapu yaput aputu.

Pada tahap ketiga dilakukan penghitungan nilai-nilai *hash* dari setiap *gram* menggunakan *rolling hash*: 12916, 12115, 12895, 12295, 13426, 12450, 13895, 12399. Lalu dibentuk *window* dari nilai-nilai *hash* dengan ukuran 4: { 12916 12115 12895 12295 }, { 12115 12895 12295 13426 }, { 12895 12295 13426 12450 }, { 12295 13426 12450 13895 }, dan { 13426 12450 13895 12399 }.

Tahap yang terakhir adalah memilih nilai hash terkecil dari setiap window untuk dijadikan sebagai *fingerprint*: [12115,1] [12295,3] [12399,7].

Rolling Hash

Fungsi yang digunakan untuk menghasilkan nilai hash dari rangkaian gram dalam algoritma *Winnowing* adalah *rolling hash*[1]. Fungsi hash $H_{(c_1 \dots c_k)}$ didefinisikan sebagai berikut:

$$c_1 * b^{(k-1)} + c_2 * b^{(k-2)} + \cdots + c_{(k-1)} * b^k + c_k$$

Rumus 1. Formula rolling hash

Keterangan :
c : nilai ascii karakter
b : basis (bilangan prima)
k : banyak karakter

Keuntungan dari rolling hash adalah untuk nilai hash berikutnya $H_{(c_2 \dots c_{k+1})}$ dapat dilakukan dengan cara:

$$H_{(c_2 \dots c_{k+1})} = \left(H_{(c_1 \dots c_k)} - c_1 * b^{(k-1)}\right) * b + c_{(k+1)}$$

Rumus 2. Formula untuk mencari nilai hash ke-2 sampai ke-n

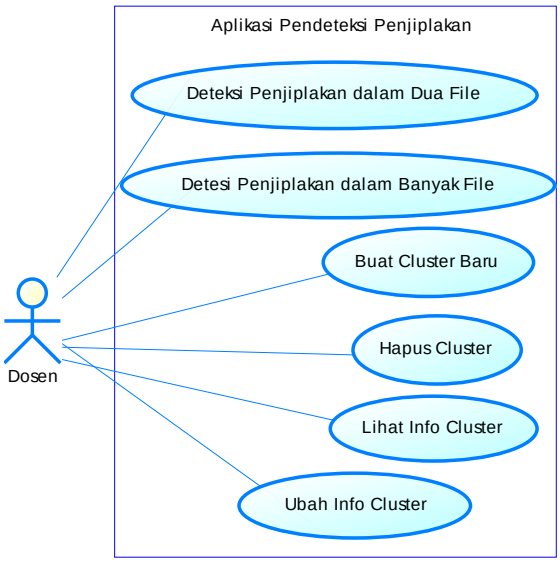
Dengan begitu tidak perlu melakukan iterasi dari indeks pertama sampai terakhir untuk menghitung nilai *hash* untuk *gram* ke-2 sampai terakhir. Hal ini tentu dapat mengehemat biaya komputasi saat menghitung nilai *hash* dari sebuah *gram*

4. Implementasi Sistem

Aplikasi ini menggunakan database yang disimpan dalam file berfomat XML. XML adalah himpunan aturan, panduan, atau konvensi untuk mendesain penyimpanan data dalam format teks, sedemikian rupa sehingga file tersebut mudah dibuat dan dibaca (oleh komputer) tanpa memerlukan bantuan *software* khusus[3].

Kebutuhan yang akan dipenuhi oleh aplikasi ini adalah: dapat mendeteksi penjiplakan yang terdapat diantara dua file, dapat mendeteksi penjiplakan yang terdapat dalam banyak file, menyediakan fitur untuk membuat cluster baru, menyediakan fitur untuk menghapus cluster, menampilkan informasi detail sebuah cluster, menyediakan fitur untuk mengubah informasi detail sebuah cluster.

Diagram use case aplikasi dapat dilihat pada gambar 1. Diagram tersebut menggambarkan aktivitas yang dapat dilakukan oleh seorang user terhadap aplikasi ini.



Gambar 1. Diagram *Use Case*

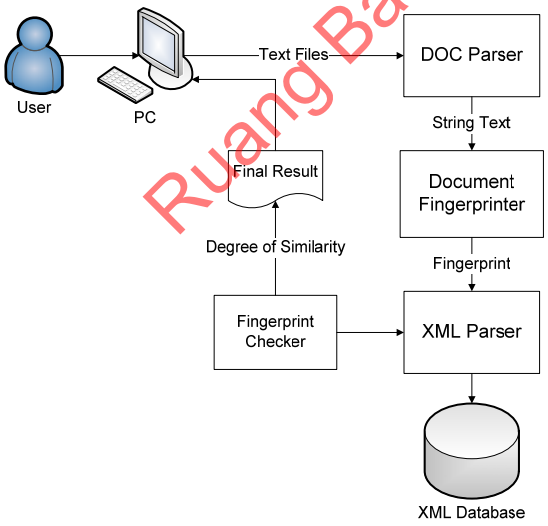
Use case diatas menggambarkan aktivitas- aktivitas yang dapat dilakukan oleh seorang user terhadap aplikasi ini: mendeteksi penjiplakan dalam dua file dan banyak file, membuat dan menghapus cluster, melihat dan mengubah informasi cluster.

Aplikasi pendeteksi penjiplakan ini dibangun dalam lingkungan pemrograman dengan spesifikasi sebagai berikut:

- Microsoft Windows Seven Ultimate sebagai sistem operasi.
- NetBeans 6.7 sebagai tool utama pemrograman dengan JDK versi 1.6.
- Sybase Power Designer 12.5 sebagia desain tool untuk membuat pemodelan UML.
- Microsoft Office Visio 2007 untuk membuat desain antarmuka. 3.4.2 Implementasi Proses.

Arsitektur

Perangkat lunak yang akan dibangun adalah perangkat lunak berbasis desktop. Gambar 6 merupakan ilustrasi arsitektur dari aplikasi pendeteksi penjiplakan yang akan dibangun.



Gambar 2. Arsitektur Aplikasi

Gambar 2. Arsitektur aplikasi

Dari gambar tersebut dapat diketahui bahwa aplikasi deteksi penjiplakan yang akan dibangun terdiri dari:

DOC Parser : bertugas mengekstrak konten / isi dari file yang ingin diperiksan menjadi sebuah *string*.

Document Fingerprinter : bertugas memproses *string* yang merupakan hasil ekstraksi dari Doc Parser menjadi *fingerprint* dari file yang berupa nilai-nilai *hash*.

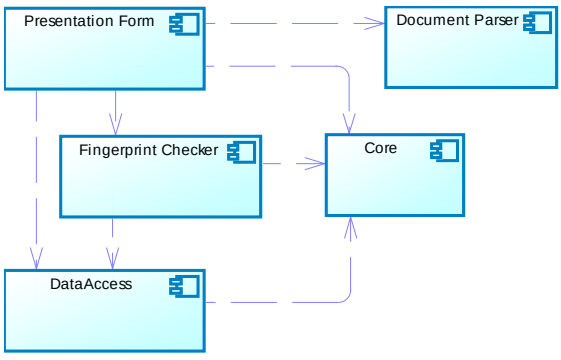
XML Parser : bertugas untuk melakukan proses baca, tulis terhardap file xml yang merupakan database dari aplikasi ini.

Fingerprint Checker : bertugas untuk membandingkan *fingerprint* yang dimiliki oleh file satu dengan yang lainnya.

Final Result : berupa derajat kesamaan antar file yang dibandingkan. Hasil ini didasarkan pada fingerprint yang dimiliki oleh setiap file yang dibandingkan.

Diagram Komponen

Aplikasi ini terdiri atas beberapa komponen – komponen kecil yang saling berhubungan sehinggan dapat membentuk sebuah aplikasi untuk mendeteksi penjiplakan pada file teks.



Gambar 3. Diagram Komponen Aplikasi

Dalam diagram diatas dapat dilihat bahwa terdapat 5 komponen utama dalam aplikasi ini :

1. Core : merupakan komponen yang berisi kelas-kelas utama, yaitu : *Cluster*, *Document*, *Winnowing Algorithm*, dll.

2. Fingerprint Checker : merupakan komponen yang berisi kelas untuk membandingkan *fingerprint* antardokumen dan menghitung tingkat kesamaan dokumen satu dengan yang lainnya.

3. Data Access : merupakan komponen yang berisi kelas untuk memanipulasi dan membaca isi file XML yang digunakan sebagai database.

4. Document Parser : merupakan komponen yang digunakan untuk membaca softcopy file berekstensi *.doc*, *.docx*, dan *.pdf* yang merupakan *input* dari aplikasi.

5. Presentation Form : merupakan komponen yang berisi kelas-kelas *user inferface* yang memudahkan user untuk menggunakan aplikasi.

Implementasi Algoritma Winnowing

Berikut ini adalah implementasi dari algoritma yang digunakan untuk proses *document fingerprinting*. Proses ini meliputi beberapa tahap yaitu:

1. Membersihkan teks dari karakter yang tidak dibuthkan.
2. Membentukan rangkaian *gram*.
3. Melakukan proses *Rolling Hash* untuk mencari nilai *hash* dari setiap *gram*.
4. Memilih *hash* untuk dijadikan *fingerprint*.

Selanjutnya akan diberikan potongan kode dalam java untuk melakukan proses-proses tersebut:

```
private void stringSanitizer(){
        StringBuffer sb = new
StringBuffer();
        char[] chars = text.toCharArray();
        for(Character c : chars){
                if(Character.isDigit(c) ||
Character.isLetter(c)){
                        sb.append(c);
                }
        }
        text = sb.toString();
}
```

Gambar 4. Kode untuk membersihkan karakter selain alfa numerik

```
private void nGramParser(){
        grams = new ArrayList<String>();
        hash = new ArrayList<Long>();
        String sub = "";
        Long hash_value;
        for(int i = 0; i <= text.length() -
ngram_size; i++){
                sub =
text.substring(i, i+ngram_size);
                grams.add(sub);
                hash_value =
this.hashGrams(sub);
                hash.add(hash_value);
        }
}
```

Gambar 5. Kode untuk membentuk rangkaian gram

```
private Long hashGrams(String sub){
        long hash_value = 0;
        int ascii;
        int length = sub.length() - 1;
        if(prev_hash == 0){
                for(int i=0;i<= length;i++){
                        ascii =
sub.charAt(i);
                        hash_value +=
(long)(ascii * Math.pow(basis,length-i));
                }
        }else{
                hash_value = prev_hash -
(long) (c_awal*Math.pow(basis, length));
                hash_value *= basis;
                hash_value +=
sub.charAt(length);
        }
        c_awal = sub.charAt(0);
        prev_hash = hash_value;
        return hash_value;
}
```

Gambar 6. Kode untuk melakukan rolling hash

```
private void chooseHash(){
        long min_hash = Long.MAX_VALUE;
        int min_index = 0;
        if(hash.size()>0){
        for(int i = 0;i<= hash.size()-
window_size;i++){
        for(int j = i; j < window_size +
```

```
i;j++){
        if(j==i){
          min_hash = hash.get(j);
                  min_index = j;
                        }else{
          if(hash.get(j) < min_hash){
            min_hash = hash.get(j);
            min_index = j;
          }
        }
        }
        if(!cekIndexFingerprint(min_index))
{
         fingerprints.add(new Fingerprint
(min_hash, min_index));
         fingerprintHash.add(min_hash);
        }
        }
        }
}
```

Gambar 7. Kode untuk memilih nilai hash yang akan dijadikan fingerprint

4. Uji Coba dan Pembahasan

Berikut ini adalah hasil uji coba terhadap aplikasi deteksi penjiplakan dengan berbagai konfigurasi/*setting*. Pengujian ini dilakukan guna mendapatkan konfigurasi yang paling baik terkait dengan ukuran *gram*, *window*, basis *hash* dan *threshold* yang digunakan untuk mendeteksi penjiplakan antarfile.

Pengujian I

Pada pengujian ini digunakan konfigurasi *default*, yaitu:

- Ukuran *Window* : 30
- Ukuran *Gram* : 30
- Basis *Hash* : 3
- *Similarity Threshold* : 50 %

Selanjutnya pengujian dilakukan terhadap 8 file tugas kuliah yang dipilih secara acak. Hasilnya adalah sebagai berikut:

Ditemukan 2 file yang tingkat kesamaannya diatas *similarity threshold*.

Similarity	File 1	File 2
99,48%	[doc] Tugas 3 Sostek - 5106100015 Kelas B	[doc] tugas 3 Sostek - 5106100026 Kelas B
98,47%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] Tugas 3 Sostek - 5106100015 Kelas B
22,94%	[doc] Tugas 3 Sostek - 5106100015 Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121
22,92%	[docx] Tugas 3 sostek - 5106100007 kelas B	[doc] Tugas 3 Soset - 5101700098 - Kelas B
22,7%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121
19,24%	[doc] Tugas 3 Soset - 5101700098 - Kelas B	[docx] Tugas 3 sostek - 5106100007 kelas B
15,62%	[docx] Tugas 3 sostek - 5106100007 kelas B	[doc] Tugas 3 Sostek - 5106100020 - kelas B
15,15%	[doc] Tugas 3 Sostek - 5106100020 - kelas B	[docx] Tugas 3 sostek - 5106100007 kelas B
.	.	.
.	.	.
.		.
0,14%	[doc] Tugas 3 Sostek - 5106100024 - Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121

Pengujian II

Pada pengujian ini digunakan konfigurasi yang berbeda dari konfigurasi *default*, yaitu:

- Ukuran *Window* : 30
- Ukuran *Gram* : 30
- Basis *Hash* : 5
- *Similarity Threshold* : 50 %

Pengujian ini digunakan untuk mengetahui seberapa besar dampak dari perubahan nilai basis untuk *hash* terhadap hasil pengecekan kesamaan yang dihasilkan oleh aplikasi. Hasilnya adalah sebagai berikut:
Ditemukan 2 file yang memiliki tingkat kesamaan yang melebihi *threshold*.

Similarity	File 1	File 2
62,54%	[doc] Tugas 3 Sostek - 5106100015 Kelas B	[doc] tugas 3 Sostek - 5106100026 Kelas B
62,33%	[doc] tugas 3 Sostek	[doc] Tugas 3 Sostek -

	- 5106100026 Kelas B	5106100015 Kelas B
0%	[doc] Tugas 3 Soset - 5101700098 - Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121
0%	[doc] Tugas 3 Soset - 5101700098 - Kelas B	[doc] Tugas 3 sosioetika-5107100116 Kelas B
0%	[doc] Tugas 3 Soset - 5101700098 - Kelas B	[docx] Tugas 3 sostek - 5106100007 kelas B
.	.	.
.	.	.
.	.	.
0%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] Tugas 3 Sostek - 5106100024 - Kelas B

Dari tabel diatas dapat dilihat bahwa tingkat kesamaan yang terjadi hanya 62,54% untuk file Tugas 3 Sostek - 5106100015 Kelas B terhadap file tugas 3 Sostek - 5106100026 Kelas B. Cukup jauh berbeda jika dibandingkan dengan pengujian I dimana tingkat kesamaan yang terjadi mencapai 99,48%. Selain itu tingkat kesamaan yang terjadi antar file-file lain menjadi 0%, yang mengindikasikan tidak terdeteksinya kesamaan teks dalam file-file tersebut.

Pengujian III

Pada pengujian ini digunakan konfigurasi yang berbeda dari konfigurasi *default*, yaitu:

- Ukuran *Window* : 30
- Ukuran *Gram* : 30
- Basis *Hash* : 17
- *Similarity Threshold* : 50 %

Pengujian ini digunakan untuk mengetahui seberapa besar dampak dari perubahan nilai basis untuk *hash* terhadap hasil pengecekan kesamaan yang dihasilkan oleh aplikasi. Hasilnya adalah sebagai berikut:

Ditemukan 2 file yang memiliki tingkat kesamaan yang melebihi *threshold*.

Similarity	File 1	File 2
66,9%	[doc] Tugas 3 Sostek - 5106100015 Kelas B	[doc] tugas 3 Sostek - 5106100026 Kelas B
66,2%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] Tugas 3 Sostek - 5106100015 Kelas B
0%	[doc] Tugas 3 Soset - 5101700098 - Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121
0%	[doc] Tugas 3 Soset - 5101700098 - Kelas B	[doc] Tugas 3 sosioetika-5107100116 Kelas B
0%	[doc] Tugas 3 Soset - 5101700098 - Kelas B	[docx] Tugas 3 sostek - 5106100007 kelas B
.	.	.
.	.	.
.	.	.
0%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] Tugas 3 Sostek - 5106100024 - Kelas B

Dari tabel diatas dapat dilihat bahwa tingkat kesamaan yang terjadi hanya 66,9% untuk file Tugas 3 Sostek - 5106100015 Kelas B terhadap file tugas 3 Sostek - 5106100026 Kelas B. Tidak jauh berbeda jika dibandingkan dengan pengujian II dimana tingkat kesamaan yang terjadi mencapai 62,54%. Selain itu tingkat kesamaan yang terjadi antar file-file lain menjadi 0%, yang mengindikasikan tidak terdeteksinya kesamaan teks dalam file-file tersebut.

Pengujian IV

Pada pengujian ini digunakan konfigurasi yang berbeda dari konfigurasi *default*, yaitu:

- Ukuran *Window* : 10
- Ukuran *Gram* : 30
- Basis *Hash* : 3
- *Similarity Threshold* : 50 %

Pengujian ini digunakan untuk mengetahui seberapa besar dampak dari perubahan ukuran *window* terhadap hasil pengecekan kesamaan yang dihasilkan oleh aplikasi. Hasilnya adalah sebagai berikut:

Ditemukan 2 file yang memiliki tingkat kesamaan yang melebihi *threshold*.

Similarity	File 1	File 2
99,36%	[doc] Tugas 3 Sostek - 5106100015 Kelas B	[doc] tugas 3 Sostek - 5106100026 Kelas B
98,51%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] Tugas 3 Sostek - 5106100015 Kelas B
22,22%	[docx] Tugas 3 sostek - 5106100007 kelas B	[doc] Tugas 3 Soset - 5101700098 - Kelas B
20,19%	[doc] Tugas 3 Sostek - 5106100015 Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121
20,02%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121
.	.	.
.	.	.
.	.	.
0,31%	[doc] Tugas 3 Sostek - 5106100024 - Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121

Dari tabel diatas dapat dilihat bahwa tidak terjadi perubahan yang signifikan terhadap tingkat kesamaan yang dilaporkan oleh aplikasi jika ukuran *window* diubah dari 30 menjadi 10.

Pengujian V

Pada pengujian ini digunakan konfigurasi yang berbeda dari konfigurasi *default*, yaitu:

- Ukuran *Window* : 50
- Ukuran *Gram* : 30
- Basis *Hash* : 3
- *Similarity Threshold* : 50 %

Pengujian ini digunakan untuk mengetahui seberapa besar dampak dari perubahan ukuran *window* terhadap hasil pengecekan kesamaan yang dihasilkan oleh aplikasi. Hasilnya adalah sebagai berikut:

Ditemukan 2 file yang memiliki tingkat kesamaan yang melebihi *threshold*.

Similarity	File 1	File 2
98,65%	[doc] Tugas 3 Sostek - 5106100015 Kelas B	[doc] tugas 3 Sostek - 5106100026 Kelas B
97,78%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] Tugas 3 Sostek - 5106100015 Kelas B
21,65%	[docx] Tugas 3 sostek - 5106100007 kelas B	[doc] Tugas 3 Soset - 5101700098 - Kelas B
18,83%	[doc] Tugas 3 Sostek - 5106100015 Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121
18,67%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121
.	.	.
.	.	.
.	.	.
0%	[doc] Tugas 3 Sostek - 5106100024 - Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121

Dari tabel diatas dapat dilihat bahwa tidak terjadi perubahan yang signifikan terhadap tingkat kesamaan yang dilaporkan oleh aplikasi jika ukuran *window* diubah dari 30 menjadi 50.

Pengujian VI

Pada pengujian ini digunakan konfigurasi yang berbeda dari konfigurasi *default*, yaitu:

- Ukuran *Window* : 30
- Ukuran *Gram* : 10
- Basis *Hash* : 3
- *Similarity Threshold* : 50 %

Pengujian ini digunakan untuk mengetahui seberapa besar dampak dari perubahan ukuran *gram* terhadap hasil pengecekan kesamaan yang dihasilkan oleh aplikasi. Hasilnya adalah sebagai berikut:

Ditemukan 2 file yang memiliki tingkat kesamaan yang melebihi *threshold*.

Similarity	File 1	File 2
99,74%	[doc] Tugas 3 Sostek - 5106100015 Kelas B	[doc] tugas 3 Sostek - 5106100026 Kelas B
98,73%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] Tugas 3 Sostek - 5106100015 Kelas B
46,41%	[doc] Tugas 3 Sostek - 5106100015 Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121
45,94%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121
40,08%	[doc] Tugas 3 sosioetika-5107100116 Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121
.	.	.
.	.	.
.	.	.
6,13%	[doc] Tugas 3 Sostek - 5106100024 - Kelas B	[doc] Tugas 3 sosioetika-5107100116 Kelas B

Dari tabel diatas dapat dilihat bahwa tidak terjadi perubahan yang signifikan terhadap tingkat kesamaan yang dilaporkan oleh aplikasi jika ukuran *gram* diubah dari 30 menjadi 10.

Namun bila dilihat laporan tingkat kesamaan file-file lain yang berada dibawah *similarity threshold* terdapat perubahan yang signifikan. Berikut ini adalah laporan 2 pasang file yang memiliki tingkat kesamaan tertinggi namun masih berada dibawah *threshold*.

Similarity	File 1	File 2
46,41%	[doc] Tugas 3 Sostek - 5106100015 Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121
45,94%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121

Jika digunakan konfigurasi *default* maka laporannya adalah sebagai berikut.

Similarity	File 1	File 2
22,94%	[doc] Tugas 3 Sostek - 5106100015 Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121
22,7%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121

Dari kedua tabel diatas dapat dilihat perbedaan yang cukup signifikan yaitu 46,41% - 22,94% dan 45,94%-22,7%. Jika ditelusuri lebih jauh, kesamaan yang muncul bisa saja terjadi secara kebetulan dan tidak disengaja karena panjang karakter yang menjadi *noise threshold* hanya 10 karakter.

Pengujian VII

Pada pengujian ini digunakan konfigurasi yang berbeda dari konfigurasi *default*, yaitu:

- Ukuran *Window* : 30
- Ukuran *Gram* : 50
- Basis *Hash* : 3
- *Similarity Threshold* : 50 %

Pengujian ini digunakan untuk mengetahui seberapa besar dampak dari perubahan ukuran *gram* terhadap hasil pengecekan kesamaan yang dihasilkan oleh aplikasi. Hasilnya adalah sebagai berikut:

Ditemukan 2 file yang memiliki tingkat kesamaan yang melebihi *threshold*.

Similarity	File 1	File 2
65,44%	[doc] Tugas 3 Sostek - 5106100015 Kelas B	[doc] tugas 3 Sostek - 5106100026 Kelas B
62,9%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] Tugas 3 Sostek - 5106100015 Kelas B
0%	[doc] Tugas 3 Soset - 5101700098 - Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121
0%	[doc] Tugas 3 Soset - 5101700098 - Kelas B	[doc] Tugas 3 sosioetika-5107100116 Kelas B
0%	[doc] Tugas 3 Soset - 5101700098 -	[docx] Tugas 3 sostek - 5106100007 kelas B

	Kelas B	
.	.	.
.	.	.
.	.	.
0%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] Tugas 3 Sostek - 5106100024 - Kelas B

Dari tabel diatas dapat dilihat bahwa terjadi perubahan yang cukup signifikan terhadap laporan tingkat kesamaan antar file, yaitu: pada file Tugas 3 Sostek - 5106100015 Kelas B dengan file tugas 3 Sostek - 5106100026 Kelas B dari 99,48% menjadi 65,44% dan sebaliknya pada file tugas 3 Sostek - 5106100026 Kelas B dengan file Tugas 3 Sostek - 5106100015 Kelas B dari 98,47% menjadi 62,9%.

Percobaan VIII

Pada pengujian ini digunakan konfigurasi yang berbeda dari konfigurasi *default*, yaitu:

- Ukuran *Window* : 30
- Ukuran *Gram* : 50
- Basis *Hash* : 3
- *Similarity Threshold* : 20 %

Pengujian ini digunakan untuk mengetahui seberapa besar dampak dari perubahan *similarity threshold* terhadap hasil pengecekan kesamaan yang dihasilkan oleh aplikasi. Hasilnya adalah sebagai berikut:

Ditemukan 5 file yang memiliki tingkat kesamaan yang melebihi *threshold*.

Similarity	File 1	File 2
99,48%	[doc] Tugas 3 Sostek - 5106100015 Kelas B	[doc] tugas 3 Sostek - 5106100026 Kelas B
98,47%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] Tugas 3 Sostek - 5106100015 Kelas B
22,94%	[doc] Tugas 3 Sostek - 5106100015 Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121
22,92%	[docx] Tugas 3 sostek - 5106100007 kelas B	[doc] Tugas 3 Soset - 5101700098 - Kelas B
22,7%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121

Perubahan *threshold* ini hanya akan berdampak pada banyaknya file yang dianggap telah melakukkan penjiplakan karena memiliki tingkat kesamaan yang melebihi batas toleransi.

Kesimpulan Hasil Uji Coba

Setelah dilakukan serangkaian dengan berbagai kofigurasi/*setting*. Penulis menyarankan konfigurasi yang paling baik untuk mendeteksi penjiplakan, yaitu:

- Ukuran *Window* : 30
- Ukuran *Gram* : 30
- Basis *Hash* : 3
- Similarity Threshold : 50%

Konfigurasi diatas dianjurkan agar hasil yang didapat lebih maksimal. Berikut ini adalah hasil yang akan didapat jika konfigurasi tersebut digunakan untuk mendeteksi penjiplakan yang terjadi dalam 8 file tugas kuliah mahasiswa:

Similarity	File 1	File 2
99,48%	[doc] Tugas 3 Sostek - 5106100015 Kelas B	[doc] tugas 3 Sostek - 5106100026 Kelas B
98,47%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] Tugas 3 Sostek - 5106100015 Kelas B
22,94%	[doc] Tugas 3 Sostek - 5106100015 Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121
22,92%	[docx] Tugas 3 sostek - 5106100007 kelas B	[doc] Tugas 3 Soset - 5101700098 - Kelas B
22,7%	[doc] tugas 3 Sostek - 5106100026 Kelas B	[doc] TUGAS 3 SOSIO ETIKA B_5107100121
.	.	.
.	.	.
.	.	.
0,14%	[doc] Tugas 3 Sostek	[doc] TUGAS 3 SOSIO

	- 5106100024 - Kelas B	ETIKA B_5107100121

Dari tabel diatas dapat dilihat bahwa seluruh kesamaan yang terjadi antar file dapat dilacak apabila menggunakan konfigurasi sesuai yang disarankan oleh penulis.

5. Simpulan

Dari hasil pengamatan mulai tahap analisis, perancangan, implementasi dan uji coba, penulis mengambil kesimpulan sebagai berikut:

Penjiplakan merupakan tindakan mencontoh atau meniru tulisan atau pekerjaan tanpa izin dari pemiliknya dan mengakui sebagai hasil karya pribadi.

Winnowing adalah algoritma untuk proses document fingerprinting, yang meliputi beberapa tahap, yaitu : pembersihah teks, pembuatan rangkaian *gram*, pencarian nilai *hash*, dan pemilihan nilai *hash* untuk dijadikan *fingerprint*.

Winnowing telah memenuhi persyaratan sebagai algoritma pendeteksi penjiplakan, yaitu : *whitespace insensitivity*, *noise suppression*, dan *positiom independence*.

Adanya cluster dapat membantu pengguna untuk mengelompokkan file-file yang sekirannya memiliki kesamaan topik sehingga besar kemungkinan telah terjadi penjiplakan.

Daftar Pustaka

[1] Schleimer, S., Wilkerson, D., dan Aiken, A. 2003. Winnowing: Local algorithms for document fingerprinting. In *Proceedings of the ACM SIGMOD international conference on management of data*. pp 76–85.

[2] ____,____, Kamus Besar Bahasa Indonesia [Online], (http://www.pusatbahasa.diknas.go.id/kbbi, diakses tanggal 1 November 2009).

[3] Kuspriyanto dan Listiana, E.,____, Transformasi XML ke Relational Database, Departemen Teknik Elektro, Institut Teknologi Bandung, Jalan Ganesha 10 Bandung 40132.

[4] Smitri W., dan Kurniawati A., 2008, Perbandingan Pendekatan Deteksi Plagiarism Dokumen dalam Bahasa Inggris, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Gunadarma.