Implementasi Algoritma TF-IDF Pada Pengukuran Kesamaan Dokumen

Adi Ryansyah[1] dan Sri Andayani[2]
Adi.ryan.syah.149@gmail.com[1] dan andayani_s@yahoo.com[2]
Universitas Katolik Musi Charitas

***Abstract*—Documents similarity measure is a time consuming problem. The large amount of documents and the large number of pages per document are causing the similarity measures to becomes a complicated and hard job to do manually. In this research, a system that can automatically measuring similarity between documents is built by implementing TF-IDF. Measurements are carried by first creating a vector representation of documents being compared. This vector representation containing the weight of each term in the documents. After that, the similarity value are calculated using cosine similarity. The finished system can carry out comparison of documents in pdf or word format. Document comparison can be done using all the chapters in the report, or just a few selected chapters that are considered significant. Based on experiment, it can be concluded that TF-IDF needs at least three documents to be available in the document collection being processes. The test of correlation shows that for document in pdf format, there is a significant correlation between the amount of characters in the document with the processing time.**

Keywords – documents similarity measure, TF_IDF, vector, cosine simility

I. PENDAHULUAN

Perkembangan teknologi informasi terutama internet telah mempermudah penyebaran dokumen dalam bentuk digital. Akibatmya jumlah dokumen yang ditemukan di internet sangatlah banyak dengan berbagai topic dan isi. Tingginya jumlah dokumen digital tersebut dapat menimbulkan berbagai macam permasalahan diantaranya adalah dokumen serupa namun beda versinya. Hal ini karena adanya kebiasaan untuk menyimpan dokumen dengan versi yang berbeda-beda bukan hanya versi terbaru. Selain ini, penyalinan dokumen dari satu tempat ke tempat yang lain juga semakin memperbanyak versi dokumen yang ada [1].

Permasalahan berikutnya adalah sulitnya melakukan pengempolkan dokumen. Pengelompokan diperlukan untuk menempatkan dokumen-dokumen ke dalam kategori yang sesuai sehingga memudahkan pencarian. Banyaknya jumlah dokumen mengakibatkan pengelompokan menjadi proses sulit dan sangat menghabiskan waktu [2]. Permasalahan terakhir adalah mudahnya melakukan plagiarisme. Plagiarisme adalah tindakan menjiplak, menyalin bahkan menjadikan karya ilmiah orang lain seolah-olah menjadi miliknya [3]. Seringkali dokumen yang ditemukan diinternet disalin namun tanpa menyertakan referensi secara jelas dan benar sehingga timbul masalah plagiarism.

Salah satu solusi yang digunakan untuk masalah di atas adalah melalui pengukuran kesamaan dokumen [4]. Pengukuran kesamaan dapat digunakan untuk menentukan apakah dokumen yang dibandingkan merupakan dokumen yang sama (hanya berbeda versi) atau sama sekali berbeda. Tingkat kesamaan dokumen juga dapat digunakan untuk mengelompokkan dokumen, di mana dokumen yang memiliki tingkat kesamaan tinggi berada pada satu kelompok. Pada pendeteksian plagiarism, pengukuran kesamaan dilakukan dengan asumsi bahwa semakin tinggi tingkat kesamaan di antara dokumen yang dibandingkan, semakin besar kemungkinan bahwa salah satu dokumen merupakan hasil plagiat [5].

Pengukuran kesamaan dokumen dilakukan dengan membandingkan satu dokumen dengan dokumen yang lainnya tentu sulit dan memakan banyak waktu jika dilakukan secara manual. Karena itu, dibangunlah sebuah aplikai untuk mempermudah yang dapat membandingkan dokumen dan mengukur tingkat kesamaan dokumen secara otomatis. Pengukuran tingkat kesamaan pada aplikasi dengan mengunakan algoritma TF-IDF. Algortima TF-IDF adalah algoritma untuk menentukan bobot dari suatu term (kata), t, pada suatu dokumen, d.

II. TINJAUAN PUSTAKA

A. Pengukuran Kesamaan Dokumen

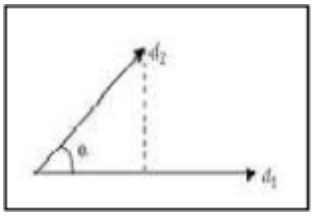
Pengukuran tingkat kesamaan memetakan tingkat kesamaan antara representasi simbolik dua objek ke dalam satu nilai tunggal. Dalam melakukan pengukuran kesamaan, tiap dokumen harus dimodelkan ke dalam suatu bentuk tertentu. Terdapat beberapa cara untuk memodelkan suatu dokumen, salah satunya sebagai bag of words (kantong huruf) yang mengasumsikan bahwa tiap kata muncul secara independen dan urutan kata-kata tersebut tidaklah penting [6].

Pada representasi dokumen dengan bag of words, tiap kata (term) akan dihitung bobotnya. Cara termudah adalah dengan menghitung frekuensi kemunculan *term* dalam dokumen dan mempergunakannya sebagai bobot *term* tersebut. Dokumen tersebut kemudian direpresentasikan sebagai suatu *vector* t_d. Jika tf(d,t) menyatakan frekuensi dari *term* t dalam dokumen d, maka representasi vemtor suatu dokumen d dinyatakan dengan persamaan 1.

$$\vec{td} = (tf\ d, t_1, \ldots, tf(d, t_m)) \quad (1)$$

Jika D={d_1, …, d_n} adalah satu set dokumen, T={t_1, … t_m} satu set term unik yang muncul dalam set dokumen D, dan tf(d,t) frekuensi term t € T dalam dokumen d € D, maka representasi *vector* dokumen d dinyatakan dengan persamaan di atas.

Representasi dokumen sebagai suatu *vector* memungkinkan perhitungan derajat kesamaan dua dokumen sebagai besar kosinus sudut yang terbentuk antara dua *vector* yang merepresentasikan kedua dokumen yang dibandingkan.



Gambar 1. Sudut Antara Dua Dokumen

Pengukuran kesamaan dokumen dengan menggunakan besar sudut tersebut dikenal dengan *cosine similarity*. Misalkan kedua dokumen yang dibandingkan adalah dokumen A dan dokumen B maka tingkat kesamaan dokumen dapat dinyatakan dengan persamaan 2[2]. Hasil perhitungan cosine similarity selalu bernilai positif dan berada pada rentang nol hingga satu [3] dengan nilai yang semakin besar menandakan tingkat kesamaan yang semakin tinggi.

$$sim = \frac{AB}{|A||B|}$$

(2)

Keterangan:

sim = tingkat kesamaan

A = vektor A yang dibentuk dari dokumen A

B = vektor B yang dibentuk dari dokumen B

$|A|$ = panjang vektor A

$|B|$ = panjang vektor B

B. *Term Frequency – Inverse Document Frequency*

Algoritma *Term Frequency – Inverse Document Frequency* (TF-IDF) merupakan algoritma yang berasal dari bidang information retrieval, namun saat ini semakin banyak digunakan dalam perbandingan dokumen [7]. Algoritma ini digunakan untuk menentukan bobot dari suatu term (kata), t, pada suatu dokumen, d, dan dinyatakan pada persamaan 3.

$$tfidf_{t,d} = tf_{t,d} \cdot idf_t$$

(3)

Keterangan:

$tfidf_{t,d}$ = bobot *term*

$tf_{t,d}$ = *term frequency* kata t pada dokumen d

idf_t = *inverse document frequency* kata t

C. *Term Frequency*

Term Frequency (TF) adalah bobot dari suatu kata, t, dalam suatu dokumen, d dan dilambangkan dengan $tf_{t,d}$. Pendekatan paling sederhana dari konsep ini adalah dengan menyatakan bobot suatu kata t sebagai jumlah kemunculannya pada dokumen d. Sebagai contoh, jika dalah suatu dokumen, kata plagiat muncul sebanyak 10 kali maka nilai TF adalah 10.

$$tf_{t,d} = \begin{cases} \log f_{t,d} + 1 & \text{Jika } f_{t,d} > 0 \\ 0 & \text{Jika kata tidak muncul} \end{cases}$$

(4)

Keterangan:

$tf_{t,d}$ = term frequency

$f_{t,d}$ = jumlah kemunculan kata/term t di dalam dokumen d

Konsep *term frequency* memandang suatu dokumen sebagai bag of words (kantong kata) di mana urutan dari kemunculan suatu kata diabaikan dan hanya jumlah kemunculan dari kata itu saja yang penting.

Konsep *term frequency* memiliki kelemahan yaitu semua kata dianggap setara. Hal ini mengakibatkan relevansi suatu kata menjadi sangat tinggi jika kata itu sering muncul dalam suatu kumpulan dokumen. Padahal tingginya frekuensi kemunculan suatu kata tidak selalu menyatakan bahwa kata tersebut penting.

D. *Inverse Document Frequency*

Konsep *inverse document frequency* (IDF) dibuat untuk mengurangi efek dari kata yang frekuensinya terlalu tinggi dalam kumpulan dokumen. Ide dasarnya adalah untuk menurunkan bobot dari kata dengan frekuensi kolektif (frekuensi total kemunculan kata di semua dokumen) yang tinggi. Dengan kata lain, semakin banyak dokumen kata tersebut pada suatu kumpulan dokumen, maka semakin rendah bobotnya.

$$idf_t = \log \frac{N}{N_t}$$

(5)

Keterangan:

idf_t = *inverse document frequency*

N = jumlah keseluruhan dokumen

N_t = jumlah dokumen yang memuat *term* t

E. *PENELITIAN TERDAHULU*

Penelitian yang berjudul *Text Reuse Detection Using a Composition of text Similarity Measures*, membandingkan berbagai metode pengukuran kesamaan seperti metode *String M*etric (Jaro, Jaro-Winkler, Monge and Elkan, dan Levenshtein), IF_TDF dan character n-gram. Penelitian ini berfokus pada penggunaan berbagai metode pengukuran kesamaan teks dengan mempertimbangkan aspek isi, struktur dan gaya penulisan pada dokumen. Pada penelitian ini, berbagai metode digabungkan guna mendapatkan hasil perhitungan kesamaan yang lebih baik [8].

Penelitian yang berjudul Aplikasi Pendeteksi Plagiat

Dengan Menggunakan Metode Latent Semantic Analysis, menggunakan algoritma IF-IDF untuk membandingkan dokumen-dokumen laporan tugas akhir. Aplikasi pendeteksian yang dikembangkan pada penelitian ini merupakan aplikasi berbasis web. Aplikasi ini memberikan dua alternatif penggunaan bagi usernya pertama, user dapat menentukan kedua dokumen yang akan dibandingkan dengan mengunggah dua dokumen yang dipilih sendiri oleh user. Kedua, memungkinkan user untuk mengunggah satu dokumen dan membandingkannya dengan kumpulan dokumen yang telah disimpan di basis data [3].

Penelitian yang berjudul *Textual Similarity*, melakukan penelitian untuk membangun sistem yang dapat mengukur kesamaan antar dokumen. Dokumen untuk pengujian sistem terdiri atas sembilan artikel dengan beberapa artikel berada pada topic yang sama. Penelitian ini terutama berfokus pada perbandingan antara empat algoritma yang dipakai dalam pengukuran kesamaan teks yaitu: *Levenshtein distance*, *textual fuzzy similarity*, IF-TDF dan *ontology based query*. Dari penelitian ini didapatkan bahwa IF-TDF merupakan algoritma dengan kinerja terbaik, diindikasikan dengan running time yang paling rendah serta hasil deteksi kesamaan yang paling baik di antara keempat algoritma [9].

III. ANALISIS PERMASALAHAN

A. Analisis Sistem

Sistem yang dibangun akan mengimplementasikan algoritma *Term Frequency – Inverse Document Frequency* (TF-IDF). Algoritma ini bekerja dengan bobot tiap kata di dalam dokumen yang dibandingkan. Proses pembobotan dengan algoritma ini membutuhkan sekumpulan dokumen agar bobot yang dihasilkan tidak hanya bergantung pada jumlah kemunculan kata. Setelah proses pembobotan, bobot tersebut disusun ke dalam *vector* yang akan diukur tingkat kesamaannya dengan menggunakan *cosine similarity*.

Sistem akan dapat menerima input berupa file-file dokumen yang berformat PDF (.dpf) atau DOC (.doc). Sebelum dilakukan pengukuran kesamaan, file-file akan melalui proses *tokenizing* dan *filtering*. *Tokenizing* merupakan proses penguraian isi dokumen menjadi unit-unit yang disebut dengan token (dalam hal ini token adalah kata). Semua token yang dihasilkan kemudian difilter untuk mendapatkan token-token yang akan dibobot. Proses filter ini akan membuang karakter-karakter yang tidak termasuk ke dalam karakter ACII. Token yang didapat dari kedua proses itu akan dibobot agar kemudian nilai kesamaan dokumen dapat dihitung.

Ekstraksi Dokumen

Sistem akan memanfaatkan Apache PDFBox(http://pdfbox.apache.org/) dan apache POI (http://poi.apache.org/) untuk ekstraksi isi dokumen ke dalam stream yang dapat diproses sistem. Apache PDFBox merupakan library yang digunakan untuk bekerja dengan file berformat .pdf, sedangkan Apache POI digunakan untuk file berformat .doc atau .docx. Keduanya merupakan library yang berbasis java dan bersifat *open-source*.

Analisis Sistem Terhadap TF-IDF

Pembobotan dengan TF-IDF melibatkan perhitungan dua komponen yaitu *term frequency* dan *inverse document frequency*. Untuk mengilustrasikan pembobotan TF-IDF akan digunakan tiga dokumen (D1, D2 dan D3) dengan isi sebagai berikut:

D1: Password yang baik haruslah tersusun atas kombinasi yang berupa huruf, angka dan simbol
D2: Password generator adalah software yang dapat menyusun password
D3: Password adalah sekumpulan karakter yang melindungi suatu software

Misalkan saja diperlukan nilai kesamaan antara D1 dan D3. Langkah pertama adalah tokenizing ada ketiga dokumen tersebut guna mendapatkan semua token. Semua token ini kemudian akan digunakan untuk menyusun daftar token-token unik (token yang sama hanya akan dimuat satu kali). Pada contoh tersebut, token yang didapatkan adalah sebagai berikut (tiap token dipisahkan dengan tanda |)

Password | yang | baik | haruslah | tersusun | atas | kombinasi | karakter | berupa | huruf | angka | dan | symbol | generator | adalah | software | dapat | menyusun | sekumpulan | melindungi | suatu.

Langkah berikutnya adalah menghitung bobot tiap token tersebut pada masing-masing dokumen D1 dan D3 dengan menggunakan persamaan 3, 4 dan 5. Perhitungan bobot tiap token pada dokumen D1 dapat dilihat pada Tabel 1. Pada tabel terdapat kolom f yang berisikan frekuensi kemunculan masing-masing token di dokumen D1. Nilai f ini akan diperlukan dalam perhitungan tf dengan persamaan 5. Pada perhitungan tf dan idf tersebut, logaritma yang digunakan logaritma berbasis dua.

Tabel 1. Perhitungan Bobot Token Pada D1

Token	$f_{t,D1}$	$tf_{t,D1}$	N_t	$idf_t = log(\frac{N}{N_t})$	$tf.idf_{t,D1}$
Password	1	1	3	0	0
Yang	2	2	3	0	0
Baik	1	1	1	1,58	1,58
Haruslah	1	1	1	1,58	1,58
Tersusun	1	1	1	1,58	1,58
Atas	1	1	1	1,58	1,58
Kombinasi	1	1	1	1,58	1,58
Karakter	1	1	2	0,58	0,58
Berupa	1	1	1	1,58	1,58
Huruf	1	1	1	1,58	1,58
Angka	1	1	1	1,58	1,58
Dan	1	1	1	1,58	1,58
Simbol	1	1	1	1,58	1,58
Generator	0	0	1	1,58	0
Adalah	0	0	2	0,58	0
Software	0	0	2	0,58	0
Dapat	0	0	1	1,58	0
Menyusun	0	0	1	1,58	0
sekumpulan	0	0	1	1,58	0
melindungi	0	0	1	1,58	0
Suatu	0	0	1	1,58	0

Perhitungan bobot token pada D3 dapat dilihat pada Tabel 2. Nilai N_t dan idf tidak bergantung pada satu dokumen namun pada keseluruhan dokumen yang ada sehingga nilainya sama untuk pembobotan pada Tabel 1 dan Tabel 2. Kedua nilai tersebut tetap dicantumkan pada Tabel 2 untuk

memudahkan dalam mengamati perhitungan bobot dokumen D3.

Setelah mendapatkan bobot semua token pada dokumen D1 dan D3, maka langkah selanjutnya adalah menyusun bobot tersebut ke dalam *vector*. Kedua *vector* yang dihasilkan sebagai berikut:

$$\vec{D1} = (0; 0; 1{,}58; 1{,}58; 1{,}58; 1{,}58; 1{,}58; 0{,}58; 1{,}58; 1{,}58; 1{,}58; 1{,}58; 1{,}58; 0; 0; 0; 0; 0; 0; 0; 0; 0)$$

$$\vec{D3} = (0; 0; 0; 0; 0; 0; 0; 0{,}58; 0; 0; 0; 0; 0; 0; 0{,}58; 0{,}58; 0; 0; 1{,}58; 1{,}58; 1{,}58)$$

Tabel II. Perhitungan Bobot Token Pada D3

Token	$f_{t,D1}$	$tf_{t,D1} = log(f)$	N_t	$idf_t = log(\frac{N}{N_t})$	$tf \cdot idf_{t,D1}$
Password	1	1	3	0	0
Yang	1	1	3	0	0
Baik	0	0	1	1.58	0
Haruslah	0	0	1	1.58	0
Tersusun	0	0	1	1.58	0
Atas	0	0	1	1.58	0
Kombinasi	0	0	1	1.58	0
Karakter	1	1	2	0.58	0.58
Berupa	0	0	1	1.58	0
Huruf	0	0	1	1.58	0
Angka	0	0	1	1.58	0
Dan	0	0	1	1.58	0
Simbol	0	0	1	1.58	0
Generator	0	0	1	1.58	0
Adalah	1	1	2	0.58	0.58
Software	1	1	2	0.58	0.58
Dapat	0	0	1	1.58	0
Menyusun	0	0	1	1.58	0
sekumpulan	1	1	1	1.58	1.58
melindungi	1	1	1	1.58	1.58
Suatu	1	1	1	1.58	1.58

Kedua vector digunakan untuk menghitung nilai kesamaan dokumen D1 dan D3. Nilai ini dihitung dengan menggunakan cosine similarity. Sesuai persamaan 2 nilai kesamaan didapat dengan menghitung dot product dari vector D1 dan D3 di dapat dari jumlah hasil kali antara komponen vector. Dengan kata lain, komponen pertama *vector* D1 dikalikan dengan komponen kedua D3, begitu seterusnya hingga komponen terakhir.

$$\begin{aligned}\vec{D1}.\vec{D3} = {} & 0 \times 0 + 0 \times 0 + 1.58 \times 0 + 1.58 \times 0 + 1.58 \times 0 + 1.58 \times 0 \\ & + 1.58 \times 0 + 0.58 \times 0.58 + 1.58 \times 0 + 1.58 \times 0 \\ & + 1.58 \times 0 + 1.58 \times 0 + 1.58 \times 0 + 0 \times 0 + 0 \times 0.58 \\ & + 0 \times 0.58 + 0 \times 0 + 0 \times 0 + 0 \times 1.58 + 0 \times 1.58 \\ & + 0 \times 1.58 \\ = {} & 0.34\end{aligned}$$

Berdasarkan perhitungan didapat bahwa dot product vector D1 dan D3 adalah 0.34. Nilai ini kemudian dibagi dengan hasil kali panjang *vector* D1 dan D3 untuk mendapatkan nilai kesamaan dokumen D1 dan D3.

$$sim = \frac{0.34}{14.76}$$

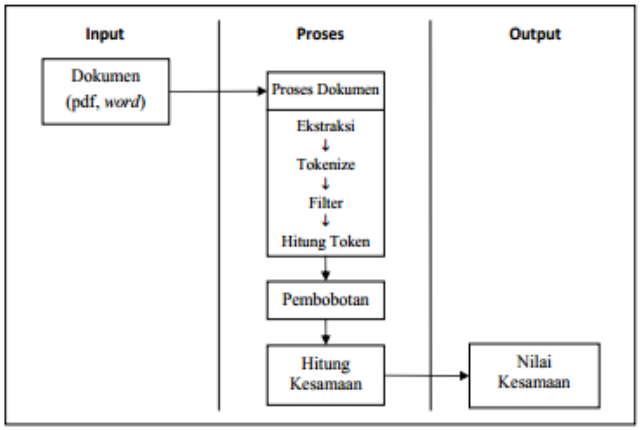
$$= 0.02$$

Berdasarkan hasil perhitungan, didapat bahwa nilai kesamaan antara dokumen D1 dan D3 adalah sebesar 0.02

B. Gambaran Umum Sistem

Sistem yang dibangun dapat menerima input berupa dokumen yang berformat pdf atau doc. Teks pada dokumen akan diekstrak lalu di-tokenize (dipecah ke dalam sekumpulan token) dan difilter, Proses filter dilakukan untuk membuang karakter yang bukan alphabet dan token yang hanya terdiri atas satu karakter. Selanjutnya, frekuensi tiap token akan dihitung dan dibobot dengan menggunakan rumusan tf-idf. Token yang telah dibobot akan dipergunakan untuk perhitungan kesamaan antar dokumen. Proses sistem dapat dilihat pada Gambar 2.

Pengukuran kesamaan dokumen dapat dilakukan antara satu dokumen dengan satu lebih dokumen lain. User dapat menentukan pasangan dokumen yang akan diukur kesamaannya. Selain itu, user dapat menentukan bab yang akan digunakan dalam proses perbandingan.

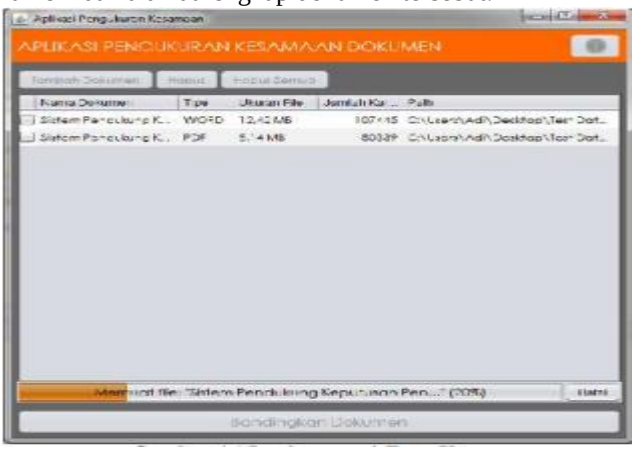


Gambar 2. Gambaran Umum Sistem

IV. IMPLEMENTASI SISTEM

Implementasi form utama dapat dilihat pada Gambar 3. Pada bagian atas form terdapat tombol Tambah Dokumen yang dapat digunakan untuk menambahkan dokumen ke sistem. Jika user mengklik tombol tersebut dan memilih dokumen maka dokumen akan dimuat ke sistem. Hal ini dilakukan dengan mengekstrak teks pada dokumen ke dalam memori. Proses ini membutuhkan waktu yang lama sehingga menampilkan progress bar di bawah tabel untuk menunjukkan sejauh mana proses ekstraksi berlangsung.

Di sebelah kanan progress bar tersebut terdapat tombol Batal yang memungkinkan user untuk membatalkan proses ekstraksi. Dokumen yang telah dimuat akan ditampilkan pada table, beserta nama, ukuran file, jumlah karakter dalam dokumen dan alamat lengkap dokumen tersebut.



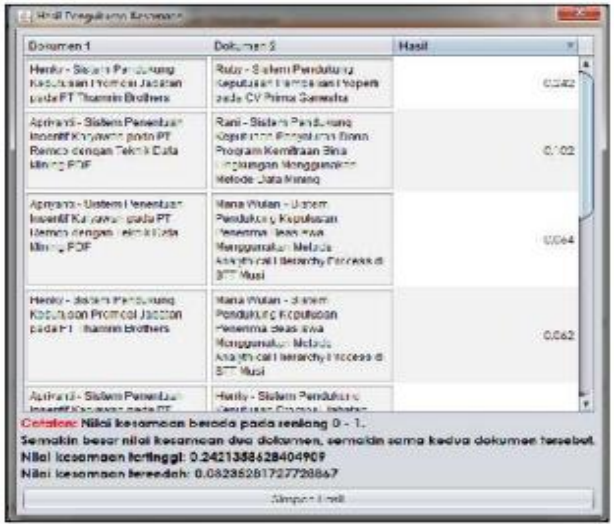
Gambar 3. Implementasi Form Utama

Sedangkan untuk implementasi diaglog pilih perbandingan dapat dilakukan dengan mengklik tombol Bandingkan dokumen pada form utama seperti pada Gambar 4. Gambar menunjukkan tabel yang telah terisi dengan pasangan perbandingan yang dihasilkan aplikasi setelah checkbox Bandingkan Semua diklik. Pasangan dokumen yang akan dibandingkan juga dapat dipilih satu persatu dengan mengklik sel pada tabel. Saat salah satu sel tabel diklik, maka akan tampil combobox berisikan daftar nama dokumen yang dipilih pada form utama.

Gambar 4. Implementasi Dialog Pilih Perbandingan

Jika semua pasangan perbandingan yang diperlukan telah dipilih, maka user dapat mengklik tombol OK untuk memulai proses perhitungan. Pemrosesan dokumen membutuhkan waktu yang cukup lama, oleh karena itu, di bawah table pasangan dokumen akan tampil progress bar yang menunjukkan status pemrosesan dokumen. Adanya progress bar untuk membantu user untuk dapat mengetahui sejauh mana proses telah berlangsung.

Selanjutnya implementasi dialog hasil pada dilihat pada Gambar 5. Dialog ini berisikan tabel yang menampilkan daftar nilai kesamaan tiap pasang dokumen yang dibandingkan dan tombol Simpan Hasil. Jika tombol Simpan Hasil diklik, maka hasil perhitungan akan disimpan untuk dapat dilihat lagi di kemudian waktu.



Gambar 5. Implementasi Dialog Hasil

Implementasi Class SimTool

Dialog Pilih Perbandingan menggunakan Class SimTool untuk melakukan sebagian besar pekerjaannya. Class SimTool memiliki beberapa method yang berperan dalam pemrosesan dan perhitungan kesamaan dokumen. Method pertama adalah method prosesDokumen (File, int[]) yang ditunjukkan pada Gambar 6. Method ini mengakses teks pada objek LaporanSkripsi yang diekstrak dari dokumen.

```
public void prosesDokumen(File aFile, int[] chaps) throws IOException,
        InvalidFormatException, ContentNotFoundException,
        OpenXML4JException, XmlException
{
    String loadedContent = "";
    LaporanSkripsi ls = (LaporanSkripsi)aFile;
    for(int i = 0; i < chaps.length; i++){
        loadedContent += (ls.getContent(chaps[i]) + " ");
    }
    files.add(aFile);
    StringTokenizer rawTokens = tokenize(loadedContent);
    while(rawTokens.hasMoreTokens()){
        String filteredWord = filter(rawTokens.nextToken());
        if(filteredWord != null) hitungToken(filteredWord, aFile);
    }
}
```

Gambar 6. Method prosesDokumen (File, [])

Method tersebut kemudian memecah teks ke dalam token dengan menggunakan method tokenize(String) yang ditampilkan pada Gambar 7. Method tokenize memecah teks ke dalam token sekaligus membuang angka dan karakter-karakter selain alphabet (tanda baca dan karakter lainnya) dari token.

```
public StringTokenizer tokenize(String deretanKata){
    String numbersDelim = "1234567890";
    String symbolDelim = "!@#$%^&*() {[]:;\"<>? -[]\\/',./-' ";
    String nonPrintableDelim = "\t\n\r";
    String delim = numbersDelim + symbolDelim + nonPrintableDelim;
    return new StringTokenizer(deretanKata, delim);
}
```

Gambar 7. Method Tokenize(String)

Setelah dipecah ke dalam token, method prosesDokumen akan memfilter tiap token dengan menggunakan method filter(String) yang ditunjukkan pada Gambar 8. Method ini memeriksa tiap karakter pada token dan membuang karakter yang bukan berupa alphabet yang tidak dapat dibuang oleh method tokenize(String).

```
public String filter(String kata){
    char[] c = kata.toCharArray();
    int numRemovedChar = 0;
    int end = c.length;

    for(int i = 0; i < end; i++){
        if((int)c[i] > 122){
            numRemovedChar++;
            end--;
            for(int j = i; j < c.length - numRemovedChar; j++){
                c[j] = c[j+1];
            }
        }
    }

    String filteredWord;
    int leftOver = c.length - numRemovedChar ;
    if(leftOver > 1){
        filteredWord = new String(c, 0, leftOver);
        return filteredWord.toLowerCase();
    }else{
        return null;
    }
}
```

Gambar 8. Method filter(String)

Setelah memfilter tiap token, method prosesDokumen(File, int[]) akan menghitung kemunculan tiap token pada dokumen yang sedang diproses. Method yang digunakan untuk perhitungan token adalah hitungToken(String, File) dan ditunjukkan pada Gambar 9.

```
public void hitungToken(String word, File aFile){
    Iterator<Token> it = tokens.iterator();
    while(it.hasNext()){
        Token t = it.next();
        if(t.getKata().equals(word)){
            t.tambahSatu(aFile);
            return;
        }
    }

    Token t = new Token(word);
    t.tambahSatu(aFile);
    tokens.add(t);
}
```

Gambar 9. Method hitungToken(String, File)

Class SimTool juga memiliki method hitungKesamaan(File, File) yang digunakan untuk menghitung kesamaan antara dua dokumen. Method ini ditunjukkan pada Gambar 10. Perhitungan kesamaan dilakukan dengan terlebih dahulu membobot semua token. Pembobotan dilakukan dengan method pembobotan(File). Method pembobotan dapat dilihat pada Gambar 11.

```
public void hitungKesamaan(File a, File b){
    double[] tfidfFirst = pembobotan(a);
    double[] tfidfSecond = pembobotan(b);
    double vectorProduct = 0;
    double vectorLength = tfidfFirst.length;
    double sim;
    for(int i = 0; i < vectorLength; i++){
        vectorProduct += (tfidfFirst[i] * tfidfSecond[i]);
    }
    sim = vectorProduct / (vectorLength(tfidfFirst) * vectorLength(tfidfSecond));
    similarities.add(new DocSim(a, b, sim));
}
```

Gambar 11. Method hitungKesamaan(File, File)

Method pembobotan menghitung bobot untuk semua token yang ada pada satu dokumen. Bobot yang dimaksud adalah nilai tfidf (term frequency – inverse document frequency). Proses perhitungan bobot pada method ini dimulai dengan perhitungan nilai term frequency (tf), dilanjutkan dengan nilai inverse document frequency (tdf), kemudian perhitungan bobot (hasil kali antara tf dengan tdf). Perhitungan nilai tf menggunakan persamaan (3), nilai idf menggunakan persamaan (4), sedangkan perhitungan nilai tfidf menggunakan persamaan (2). Hasil perhitungan adalah vector menjadi representasi dokumen tersebut.

```
public double[] pembobotan(File aFile){
    int numTokens = tokens.size();
    double numDoc = files.size();
    double[] tfidf = new double[tokens.size()];

    for(int i = 0; i < numTokens; i++){
        Token currentToken = tokens.get(i);
        double tf, idf;
        int frekuensi = currentToken.getFrekuensi(aFile);

        if(frekuensi > 0) tf = Math.log10(frekuensi) + 1;
        else tf = 0;

        double occurrence = currentToken.getJumlahDoc();
        idf = Math.log10(numDoc / occurrence);
        tfidf[i] = tf * idf;
    }
    return tfidf;
}
```

Gambar 12. Method pembobotan(File)

V. HASIL DAN PEMBAHASAN

Hasil yang didapat dari implementasi sistem kemudian dilakukan pengujian.

Pengujian Tingkat Kesamaan Dokumen

Pengujian ini dilakukan untuk melihat hasil perbandingan antara dokumen-dokumen dengan tingkat kesamaan yang telah diketahui. Hasil pengujian akan digunakan sebagai interpretasi tingkat kesamaan yang dihasilkan sistem. Tingkat kesamaan yang berada pada rentang 0-1 akan dibagi ke beberapa bagian yang tiap bagiannya menyatakan tingkat kesamaan sangat tinggi, tinggi, sedang, rendah dan sangat rendah.

Pengujian dilakukan dengan membandingkan satu dokumen dengan salinan dari dokumen itu sendiri. Tiap dokumen memiliki lima salinan dengan tingkat kesamaan yang telah ditentukan. Tingkat kesamaan yang ditentukan tersebut adalah tingkat kesamaan antara dokumen salinan dengan dokumen asli, Untuk mendapatkan tingkat kesamaan tersebut, tiap salinan telah mengalami pengurangan jumlah halaman sesuai dengan tingkat kesamaannya. Total halaman yang dikurangi berjumlah 10% dari dokumen asli untuk salinan dengan tingkat kesamaan sangat tinggi, 30% untuk tingkat kesamaan tinggi, 50% untuk kesamaan sedang, 70% untuk tingkat kesamaan rendah dan 90% untuk tingkat

kesamaan sangat rendah. Sebagai contoh, jika terdapat dokumen A dengan jumlah halaman 100 maka salinan dokumen tersebut yang tingkat kesamaannya tinggi akan mengalami pengurangan halaman sebanyak sepuluh halaman. Pengurangan halaman pada setiap dokumen hasil salinan dilakukan secara acak.

Pengujian dilakukan dengan menggunakan sepuluh dokumen seperti terlihat pada Tabel III.

Tabel III. Daftar Dokumen

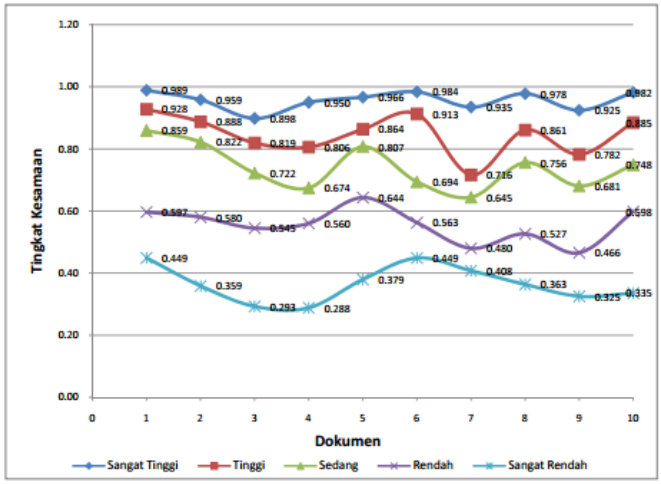
No	Tipe File	Ukuran File
1	PDF	2,56 MB
2	PDF	5.13 MB
3	PDF	3.00 MB
4	DOC	2.88 MB
5	PDF	737 KB
6	DOC	3.53 MB
7	PDF	13.5 MB
8	DOC	2.75 MB
9	DOC	12.4 MB
10	PDF	10.1 MB

Untuk dapat melihat nilai kesamaan yang didapat antara dokumen asli dengan tiap salinannya seperti pada Tabel IV.

Tabel IV. Tingkat Kesamaan Dokumen Asli dan Salinan

No	Tingkat Kesamaan				
	Sangat Tinggi	Tinggi	Sedang	Rendah	Sangat Rendah
1	0.989	0.928	0.859	0.597	0.449
2	0.959	0.888	0.822	0.580	0.359
3	0.898	0.819	0.722	0.545	0.293
4	0.950	0.806	0..674	0.560	0.288
5	0.966	0.864	0.807	0.644	0.379
6	0.984	0.913	0.694	0.563	0.449
7	0.935	0.716	0.645	0.480	0.408
8	0.978	0.861	0.756	0.527	0.363
9	0.925	0.782	0.681	0.466	0.325
10	0.982	0.855	0.748	0.598	0.335

Berdasarkan data pada Tabel IV didapatkan tingkat kesamaan sangat tinggi pada rentang 0.92 – 1, sedangkan tingkat kesamaan tinggi berada pada rentang 0.80 – 0.91, tingkat kesamaan sedang berada antara 0.64 – 0.79, tingkat kesamaan rendah berada pada rentang 0.45 – 0.63 dan tingkat kesamaan sangat rendah pada rentang 0 – 0.44. Dapat dilihat pada Gambar 13.



Gambar 13. Grafik Tingkat Kesamaan Dokumen Asli dan Salinan

Pengujian Pengaruh Jumlah Dokumen Terhadap Hasil Pengukuran

Pengukuran ini dilakukan untuk melihat pengaruh jumlah dokumen yang dipilih terhadap hasil pengukuran kesamaan. Pengukuran dilakukan secara berulang-ulang dengan jumlah dokumen yang berbeda. Pengukuran pertama dilakukan dengan dua dokumen, pengukuran berikutnya tiga dokumen begitu seterusnya hingga sepuluh dokumen digunakan dalam pengukuran.

Pengukuran dilakukan secara berulang-ulang dengan jumlah dokumen yang berbeda. Pengukuran pertama dilakukan dengan dua dokumen, pengukuran berikutnya tiga dokumen dan seterusnya. Tabel V menampilkan hasil pengukuran kesamaan antara pasangan dokumen yang dapat disusun sesuai dengan jumlah dokumen yang dipilih. Sebagai contoh, jika terdapat dua dokumen yang dipilih maka hanya terdapat satu perbandingan file nomor 1 dan nomor 2 yang ditampilkan pada kolom berlabel 2. Jika terdapat tiga dokumen maka terdapat tiga pasangan perbandingan yang ditampilkan pada kolom berlabel 3 dan seterusnya.

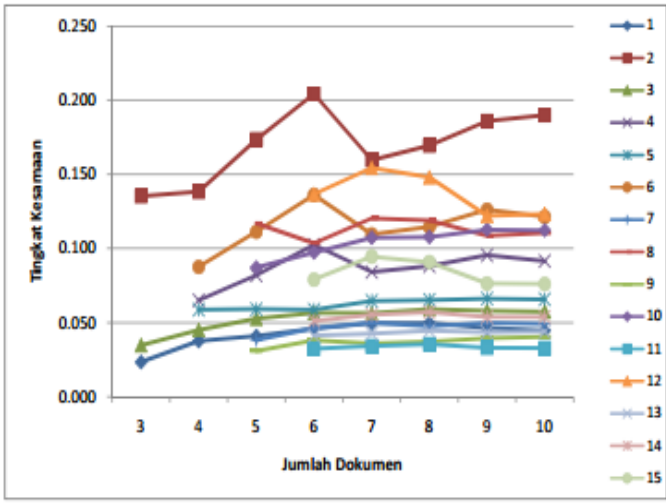
Tabel V. Hasil Pengukuran Kesamaan Berdasarkan Jumlah Dokumen

No.	File 1	File 2	Hasil Pengukuran								
			2	3	4	5	6	7	8	9	10
1	1	2	0	0.023	0.038	0.041	0.046	0.049	0.050	0.046	0.045
2	1	3		0.135	0.138	0.174	0.204	0.160	0.169	0.186	0.190
3	2	3		0.035	0.045	0.053	0.056	0.056	0.059	0.058	0.057
4	1	4			0.065	0.082	0.102	0.084	0.088	0.095	0.091
5	2	4			0.059	0.059	0.059	0.065	0.065	0.066	0.066
6	3	4			0.088	0.111	0.136	0.109	0.115	0.126	0.121
7	1	5				0.038	0.046	0.050	0.048	0.050	0.049
8	2	5				0.117	0.104	0.120	0.119	0.109	0.111
9	3	5				0.031	0.038	0.036	0.037	0.039	0.040
10	4	5				0.087	0.097	0.107	0.108	0.113	0.112
11	1	6					0.032	0.034	0.035	0.033	0.033
12	2	6					0.137	0.154	0.148	0.122	0.123
13	3	6					0.041	0.042	0.045	0.043	0.044
14	4	6					0.051	0.055	0.057	0.054	0.054
15	5	6					0.079	0.095	0.091	0.076	0.076
16	1	7						0.165	0.176	0.192	0.195
17	2	7						0.034	0.036	0.038	0.035
18	3	7						0.316	0.328	0.348	0.354
19	4	7						0.125	0.132	0.143	0.137
20	5	7						0.040	0.040	0.044	0.043
21	6	7						0.035	0.037	0.037	0.038
22	1	8							0.062	0.065	0.063
23	2	8							0.104	0.097	0.096
24	3	8							0.057	0.062	0.060
25	4	8							0.074	0.077	0.075
26	5	8							0.105	0.105	0.102
27	6	8							0.083	0.072	0.071
28	7	8							0.048	0.055	0.056
29	1	9								0.048	0.045

No.	File 1	File 2	Hasil Pengukuran								
			2	3	4	5	6	7	8	9	10
30	2	9								0.178	0.180
31	3	9								0.041	0.039
32	4	9								0.052	0.049
33	5	9								0.084	0.084
34	6	9								0.112	0.113
35	7	9								0.042	0.041
36	8	9								0.075	0.071
37	1	10									0.083
38	2	10									0.065
39	3	10									0.081
40	4	10									0.095
41	5	10									0.085
42	6	10									0.066
43	7	10									0.076
44	8	10									0.105
45	9	10									0.071

Pada Tabel V terlihat bahwa saat dokumen yang dipilih dua maka tingkat kesamaan dokumen yang dihasilkan adalah nol. Hal ini dikarenakan saat satu kata hanya muncul di dokumen pertama maka frekuensi kata itu di dokumen kedua adalah nol sehingga nilai tf-nya adalah nol dan bobotnya (hasil kali tf dan idf) adalah nol. Sedangkan jika satu kata muncul di kedua dokumen maka nilai idf-nya adalah nol sehingga hasil kali tf dan idf juga nol.

Pada saat jumlah dokumen ditambahkan menjadi tiga, nilai kesamaannya sudah meningkat. Karena itu, perhitungan kesamaan dengan metode tf-idf membutuhkan paling tidak tiga dokumen. Pada saat dokumen ditambahkan satu demi satu, nilai kesamaan antar pasangan dokumen mengalami perubahan. Perubahan ini dapat dilihat pada grafik pada Gambar 14.



Gambar 14. Grafik Perubahan Nilai Kesamaan Berdasarkan Jumlah Dokumen

Pengujian Waktu Eksekusi

Dilakukan pengukuran waktu pemrosesan dokumen guna melihat hubungan antara ukuran dan format file terhadap waktu pemrosesan. Pengukuran dilakukan dengan meproses semua dokumen tersebut sebanyak seratus kali dan mengukur waktu yang dibutuhkan pada setiap pemprosesan. Semua hasil pengukuran tersebut kemudian dirata-rata untuk mendapat waktu pemprosesna rata-rata per dokumen. Pengujian waktu eksekusi dapat dilihat pada Tabel VI.

Tabel VI. Rata-Rata Waktu Pemprosesan Dokumen

No	Ukuran File (MB)			Jumlah Karakter	Waktu Pemprosesan (milidetik)		
	PDF	DOCX	DOC		PDF	DOCX	DOC
1	2.56	2.001	3.403	62212	1235.32	582.22	661.84
2	5.13	1.671	4.427	80339	1942.63	672.6	722.62
3	3.00	3.691	4.175	108140	1810.12	983.15	3389.89
4	1.882	1.91	2.883	72758	1884.96	639.65	627.5
5	0.737	0.226	1.182	60498	1409.28	980.34	908.57
6	1.058	2.369	3.515	90144	642.58	771.42	741.8
7	13.537	3.533	3.865	88547	1833.72	814.62	801.66
8	1.611	1.18	2.649	124096	3176.49	5394.75	1137.02
9	2.079	12.42	14.03	80912	1286.16	1109.53	1330.87
10	10.109	2.635	4.553	97088	1102.54	865.94	885.4

Waktu pemprosesan pada Tabel VI adalah total waktu yang diperlukan untuk memuat dan memproses file tersebut. Waktu pemprosesan dinyatakan dalam milidetik. Pemrosesan dokumen dilakukan dengan format yang berbeda-beda yaitu pdf, doc dan docx.

Selanjutnya dilakukan uji statistic berupa uji korelasi untuk melihat tingkat korelasi antara jumlah karakter pada dokumen dengan lama waktu pemrosesan. Uji statistic dilakukan dengan menggunakan SPSS. Tingkat siginifikansi adalah 0.01.

Gambar 15. Menunjukkan hasil uji korelasi antara waktu pemrosesan dokumen berformat PDF dengan jumlah karakter pada dokumen. Variabel *length* menyatakan jumlah karakter dalam file sedangkan variable pdf menyatakan waktu pemrosesan. Nilail Pearson Correlationnya sebesar 0.642 dengan nilai signifikansi 0,00 (lebih kecil dari tingkat signifikansi yang ditetapkan yaitu 0,01) yang berarti terdapat korelasi yang siginifikan antara jumlah karakter dengan waktu pemrosesan dokumen berformat PDF. Arah korelasi adalah positif yang berarti semakin banyak jumlah karakter, semakin lama waktu pemrosesan.

Correlations

		length	pdf
length	Pearson Correlation	1	.642**
	Sig. (2-tailed)		.000
	N	28	28
pdf	Pearson Correlation	.642**	1
	Sig. (2-tailed)	.000	
	N	28	28

**. Correlation is significant at the 0.01 level (2-tailed).

Gambar 15. Hasil Uji Korelasi Waktu Pemprosesan Dokumen Berformat PDF dengan Jumlah Karakter Pada Dokumen

Sedangkan nilai person correlation untuk dokumen berformat DOCX sebesar 0,207 dan nilail signifikansi sebesar 2.91 (lebih besar dari tingkat signifikansi yang ditetapkan) yang berarti tidak ada korelasi signifikansi

antara jumlah karakter dengan waktu pemprosesan dokumen berformat DOCX. Seperti ditunjukkan pada Gambar 16.

Correlations

		length	docx
length	Pearson Correlation	1	.207
	Sig. (2-tailed)		.291
	N	28	28
docx	Pearson Correlation	.207	1
	Sig. (2-tailed)	.291	
	N	28	28

Gambar 16. Hasil Uji Korelasi Waktu Pemprosesan Dokumen Berformat DOCX dengan Jumlah Karakter Pada Dokumen

Gambar 17 menunjukkan hasil uji korelasi antara ukuran file dengan waktu pemprosesan dokumen bertipe DOC. Nilai signifikansi adalah 0.304 lebih besar dari tingkat signifikansi yang ditetapkan sehingga tidak terdapat korelasi signifikansi antara jumlah karakter dengan waktu pemprosesan dokumen berformat DOC.

Correlations

		length	doc
length	Pearson Correlation	1	.187
	Sig. (2-tailed)		.340
	N	28	28
doc	Pearson Correlation	.187	1
	Sig. (2-tailed)	.340	
	N	28	28

Gambar 17. Hasil Uji Korelasi Waktu Pemprosesan Dokumen Berformat DOCX dengan Jumlah Karakter pada Dokumen

Pengujian Perbandingan Data Sampel

Data sampel yang diambil adalah data yang mempunyai topic yang sama. Daftar data sampel sesuai dengan Tabel III. Hasil perbandingan dokumen dapat dilihat pada Tabel VII. Data pada tabel telah diurutkan berdasarkan tingkat kesamaan dokumen, dimulai dari tingkat kesamaan tertinggi hingga yang paling rendah. Tiap dokumen diwakilkan dengan angka yang merupakan nomor urut dokumen yang dibandingkan.

Berdasarkan tabel tersebut, nilai kesamaan yang tertinggi adalah 0.34292705 yaitu kesamaan antara dokumen nomor 4 dan 5 pada Tabel III karena mempunyai topic dan metode yang digunakan sama.

Tabel VII. Hasil Perbandingan Data Sampel

No	No Dokumen 1	No Dokumen 2	Nilai Kesamaan
1	4	5	0.34292705
2	4	9	0.18476728
3	1	2	0.17887911
4	5	9	0.17320938
5	4	6	0.13977831
6	2	8	0.12448809
7	5	6	0.1198659
8	1	8	0.11567855
9	6	10	0.11084461
10	2	10	0.11029629
11	3	7	0.104325
12	7	10	0.10260571
13	2	7	0.09389999
14	3	6	0.09376539
15	6	9	0.09364074
16	3	10	0.08560643
17	1	10	0.08399657
18	8	10	0.07833856
19	3	5	0.07807229
20	3	4	0.07634604
21	3	9	0.07431591
22	6	7	0.07431283
23	7	8	0.07008661
24	1	7	0.06981982
25	1	3	0.06899071
26	3	8	0.06644366
27	2	6	0.06402687
28	2	3	0.06363066
29	7	9	0.06188543
30	5	7	0.05928796
31	2	5	0.0572615
32	9	10	0.05512373
33	6	8	0.05388379
34	4	7	0.05387936
35	1	6	0.04745377
36	1	9	0.04481552
37	5	8	0.04474773
38	2	9	0.043277
39	4	10	0.04216459
40	1	4	0.04204607
41	4	8	0.03944676
42	5	10	0.03797045
43	1	5	0.03797045
44	2	4	0.03689451
45	8	9	0.03645244

VI. KESIMPULAN

Algoritma TF-IDF dapat diimplementasikan dalam sistem pengukuruan kesamaan dokumen. Hasil pengujian tingkat kesamaan menunjukkan bahwa algoritma ini membutuhkan paling sedikit tiga dokumen dalam kumpulan dokumen yang digunakan. Hasil uji korelasi menunjukkan bahwa untuk dokumen berformat pdf terdapat hubungan yang erat antara jumlah karakter pada dokumen dengan lama waktu pemprosesan.

REFERENSI

[1] Hoad, Timothy C dan Zobel, Justin. (2003). Methods for Identifying Versioned and Plagiarized Documents. Journal of the American Society for Information Science and Technology. 54(3): 203-215

[2] Kusniawati, Ana dan Wicaksana, I Wayan Simri. (2008). Perbandingan Pendekatan Deteksi Plagiarism Dokumen dalam Bahasa Inggris. *Proceeding Seminar Nasional Komputer dan Sistem Intelijen* (KOMMIT 2008). 284-291

[3] Khairunnisa, Nova. SS, Syarif Dadang dan Wibowo, Ardianto. (2012). Aplikasi Pendeteksian Plagiat dengan Menggunakan Metode Latent Semantic Analysis (Studi Kasus: Laporan TA PCR), e-Journal Teknik Informatika Vol.1 No.95 (http://journal.pcr.ac.id/i-journal/page/read_pdf.php?name=paper.pdf&id=95)

[4] Hatzivassiloglou, Vasileios. Klavans, Judith L. Eskin, Eleazar. (1999). Detecting Text Similarity over Short Passages: Exploring Linguistic Feature Combinations via Machine Learning. (http://www1.cs.columbia.edu/vh/papers/1999/SimFinder-EMNLP.pdf)

[5] Clough, Paul. (2003). Old and New Challenges in Automatic Plagiarism Detection. (http://ir.shef.ac.uk/cloughie/papers/pas_plagiarism.pdf)

[6] Huang, Ana. (2008). Similarity Measures for Text Document Clustering. (http://www.milanmirkovic.com/wp-content/uploads/2012/10/pg049_Similarity_Measures_for_Text_DocumeDo_Clustering.pdf)

[7] Micol, Daniel. Ferrandez. Oscar. Llopis, Fernando dan Munoz, Rafael. (2010). A Textual-Based Similarity Approach for Efficient and Scalable External Plagiarism Analysis. (http://www.uni-weimar.de/medien/webis/research/events/pan-10/pan10-papers-final/pan10-palgiarism-detection/micol10-a-textual-based-similarity-approach-for-efficient-and-scalable-external-plagiarism-analysis.pdf)

[8] Bar, Daniel. Zesch, Torsten dan Gurevych, Iryna. (2012). Text Reuse Detection Using a Composition of Text Similarity Measures, Proceedings of COLING 2012: Technical Papers, pages 167-184. COLING 2012. Mumbai. December.

[9] Van Beusekom, Johan dan Poulsen, Peter Gammelgaard. Textual Similarity. Skripsi. Informatics and Mathematical Modelling Technical University of Denmark. Denmark. 2012.