VOLUME 10 - NOMOR 1 - JUNI 2014

ISSN 0216-4280

Jurnal Informatika

ISSN 0216-4280

9 770216 428004

UNIVERSITAS KRISTEN MARANATHA - BANDUNG

j. informatika	Vol. 10	No. 1	Hlm. 1 - 103	Bandung, Juni 2014	ISSN 0216-4280

Jurnal Informatika

Volume 10 Nomor 1 Juni 2014

DAFTAR ISI

Implementasi *Cosine Similarity* dan Algoritma *Smith-Waterman* untuk Mendeteksi Kemiripan Teks

Radiant Victor Imbar[1], Adelia[2], Mewati Ayub[3], Alexander Rehatta[4]
[1,2]Jurusan S1 Sistem Informasi, [3,4]Jurusan S1 Teknik Informatika
Fakultas Teknologi Informasi Universitas Kristen Maranatha
Jl. Prof. Drg. Suria Sumantri No. 65 Bandung 40164
email: [1]radiant.vi@eng.maranatha.edu, [2]rabell_green@yahoo.com,
[3]mewatia@yahoo.com, [4]all_purewhite@yahoo.co.id

Abstract

One's writing originality in academic world becomes more and more questionable along with the increasingly access to others' writings due to files archiving technology development today, especially over the internet. Therefore, a text similarity detection system is required. Based on that problem, this research tries to provide the solution by developing an application with the concept of text mining which implements cosine similarity and Smith-Waterman algorithm to detect text similarity. Cosine similarity serves to measure text similarity based on words occurrence, while Smith-Waterman algorithm's function is to calculate text similarity based on words sequence. Based on this research test result, the developed application successfully detects text similarity from very similar to very dissimilar pair of texts.

Keywords: *cosine similarity, smith-waterman algorithm, similarity value*

1. Pendahuluan

Dalam dunia akademik, karya tulis seseorang dinilai berdasarkan keorisinalannya. Dengan adanya perkembangan pengarsipan berkas, terutama melalui *internet* yang terus berkembang saat ini, akses untuk mendapatkan karya-karya tulis tersebut menjadi semakin mudah. Contoh karya tulis akademis yang tersebar di *internet* adalah jurnal ilmiah dan tugas akademis mahasiswa. Karenanya, kemungkinan untuk menyalin karya-karya tulis tersebut menjadi lebih tinggi, dan keorisinalan karya tulis akademis pun semakin dipertanyakan. Dengan demikian, perlu adanya pendeteksian kemiripan teks untuk karya-karya tulis akademis sehingga keorisinalannya dapat diketahui dengan cepat.

Namun, pendeteksian kemiripan teks menjadi tugas yang sulit dilakukan oleh manusia karena banyak dan besarnya teks untuk dibandingkan serta strukturnya yang tidak konsisten dan kompleks. Pendeteksian kemiripan teks dapat

dilakukan untuk berbagai tujuan, salah satunya adalah untuk mencegah plagiarisme (tindakan menyatakan karya orang lain sebagai karya sendiri tanpa mengacu pada karya asli). Sebuah penelitian mengenai pendeteksian plagiarisme menggunakan algoritma Smith-Waterman telah dilakukan sebelumnya [3].

Untuk menangani teks berukuran besar yang banyak, diperlukan suatu aplikasi yang dapat mengotomatisasi proses pendeteksian kemiripan teks. Sedangkan untuk mengatasi struktur teks yang tidak konsisten dan kompleks, diperlukan konsep *text mining*. Berfokuskan pada teks berbahasa Indonesia dan berkonsepkan *text mining*, penelitian ini bertujuan untuk mengembangkan sebuah aplikasi yang mengimplementasikan *cosine similarity* yang berguna untuk mengukur kesamaan teks berdasarkan kemunculan kata-kata dalam teks tersebut dan algoritma *Smith-Waterman* yang berfungsi untuk menghitung kemiripan teks berdasarkan urutan kata.

Sebagai tambahan, penelitian ini menggunakan algoritma Nazief-Adriani untuk mengubah kata-kata berbahasa Indonesia menjadi kata-kata dasarnya (*stemming*) yang nantinya digunakan dalam tahap *preprocessing*. Ruang lingkup penelitian ini adalah sebagai berikut: kata-kata yang dapat di-*stem* hanyalah kata-kata yang berbahasa Indonesia dan aplikasi dapat memroses teks yang berada di dalam berkas berekstensi .doc,.docx, serta .txt.

2. Landasan Teori

Berdasarkan pendahuluan yang telah dijelaskan, landasan teori yang digunakan untuk mendukung penelitian ini dijabarkan sebagai berikut ini.

2.1 *Text Mining*

Text Mining adalah sebuah penerapan yang berasal dari *information retrieval* (IR) dan *natural language processing* (NLP). Definisi *text mining* secara sempit hanya berupa metode yang dapat menemukan informasi baru yang tidak jelas atau mudah diketahui dari sebuah kumpulan dokumen. Sedangkan secara lebih luas, *text mining* mencakup teknik *text-processing* yang lebih umum, seperti pencarian, pengambilan intisari, dan pengkategorian [4].

Permasalahan yang dihadapi pada *text mining* adalah jumlah data yang besar, dimensi yang tinggi, data dan struktur yang terus berubah, serta *data noise*. Sehingga sumber data yang digunakan pada *text mining* adalah kumpulan teks yang memiliki bentuk yang tidak terstruktur atau setidaknya semi terstruktur.

Tujuan dari *text mining* adalah untuk mendapatkan informasi yang berguna dari sekumpulan dokumen dalam bentuk teks [5].

2.2 Langkah-langkah Pendeteksian Kemiripan Teks

Berikut ini adalah langkah-langkah pendeteksian kemiripan teks yang dilaksanakan dalam penelitian ini:

1. *Preprocessing*
   Preprocessing adalah proses pengubahan bentuk data yang terstruktur sembarang menjadi data yang terstruktur sesuai kebutuhan untuk proses dalam *text mining*. Tahap *preprocessing* terdiri dari *case folding*, *tokenizing*, *filtering*, *stemming*, *tagging*, dan *analyzing* [5]. Penelitian ini menggunakan tahap *case folding* hingga *stemming* yang akan dijelaskan sebagai berikut.

   a. *Case Folding* dan *Tokenizing*
      Case folding adalah tahap mengubah semua huruf dalam dokumen menjadi huruf kecil. Selain itu, karakter non-huruf akan dihilangkan. *Tokenizing* adalah tahap pemecahan kalimat berdasarkan tiap kata yang menyusunnya [5].
   b. *Filtering*
      Filtering adalah tahap mengambil kata-kata penting dari hasil tahap *tokenizing*. *Filtering* dapat dilakukan dengan menghilangkan *stoplist*/*stopword* (kata-kata yang tidak deskriptif, seperti kata "yang" dan "dari") [5].
   c. *Stemming*
      Stemming adalah tahap transformasi suatu kata menjadi kata dasarnya (*root word*) dengan menggunakan aturan-aturan tertentu [1]. Dalam penelitian ini, *stemming* akan dilakukan dengan algoritma Nazief-Adriani.
2. Perhitungan kemiripan teks
   Setelah tahap *preprocessing* dilakukan terhadap dua teks masukan, maka perhitungan kemiripan keduanya dapat dilakukan. Dalam penelitian ini, ada dua cara yang akan dilakukan untuk memperhitungkan kemiripan teks, yaitu dengan *cosine similarity* (berdasarkan kemunculan kata) dan algoritma Smith-Waterman (berdasarkan urutan kata).

2.3 Algoritma Nazief-Adriani

Algoritma ini dibuat berdasarkan aturan morfologi dan mengenkapsulasi afiks yang diizinkan dan dilarang, termasuk prefiks, sufiks, infiks (penyisipan) dan konfiks (kombinasi prefiks dan sufiks). Algoritma ini juga mendukung

recoding, sebuah pendekatan untuk mengembalikan sebuah huruf inisial yang telah dihapus dari sebuah akar kata sebelum mengawali sebuah prefiks. Sebagai tambahan, algoritma ini menggunakan sebuah kamus tambahan dari akar kata yang digunakan pada kebanyakan tahap untuk mengecek apakah *stemming* telah mencapai akar kata [2].

Anggap pengelompokan dasar dari penggunaan afiks sebagai pendekatan dasar, dan cara definisi-definisi ini digabungkan untuk membentuk sebuah *framework* untuk mengimplementasikan aturan-aturan tersebut. Skema tersebut mengelompokkan afiks menjadi tiga kategori, yaitu *inflection suffixes*, *derivation suffixes*, dan *derivation prefixes* [2]. Dikutip dari jurnal yang berjudul "Perbandingan Algoritma Stemming Porter dengan Algoritma Nazief & Adriani untuk Stemming Dokumen Teks Bahasa Indonesia" oleh Ledy Agusta, berikut ini adalah tahap-tahap algoritma Nazief-Adriani:

1. Cari kata yang akan di-*stem* dalam kamus. Jika ditemukan maka diasumsikan bahwa kata tersebut adalah *root word*. Maka algoritma berhenti.
2. *Inflection suffixes* ("-lah", "-kah", "-ku", "-mu", atau "-nya") dibuang. Jika berupa *particles* ("-lah", "-kah", "-tah", atau "-pun") maka langkah ini diulangi lagi untuk menghapus *possesive pronouns* ("-ku", "-mu", atau "-nya"), jika ada.
3. Hapus *derivation suffixes* ("-i", "-an", atau "-kan"). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka lakukan langkah 3a.
   a. Jika "-an" telah dihapus dan huruf terakhir dari kata tersebut adalah "-k", maka "-k" juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan, maka lakukan langkah 3b.
   b. Akhiran yang dihapus ("-i", "-an", atau "-kan") dikembalikan, lanjut ke langkah 4.
4. Hapus *derivation prefix*. Jika pada langkah 3 ada sufiks yang dihapus, maka lakukan ke langkah 4a. Jika tidak, lakukan langkah 4b.
   a. Periksa tabel kombinasi awalan-akhiran yang tidak diizinkan. Jika ditemukan maka algoritma berhenti, jika tidak lakukan langkah 4b.
   b. *For i* = 1 *to* 3, tentukan tipe awalan kemudian hapus awalan. Jika *root word* belum juga ditemukan lakukan langkah 5, jika sudah maka algoritma berhenti. Catatan: jika awalan kedua sama dengan awalan pertama algoritma berhenti.
5. Melakukan *recoding*.
6. Jika semua langkah telah selesai tetapi tidak juga berhasil, maka kata awal diasumsikan sebagai kata dasar. Proses selesai. (Agusta, 2009, p.197)

2.4 *Cosine Similarity* (Kesamaan Kosinus)

Cosine similarity adalah ukuran kesamaan yang lebih umum digunakan dalam *information retrieval* dan merupakan ukuran sudut antara vektor dokumen D_a (titik (ax, bx)) dan D_b (titik (ay, by)). Tiap vektor tersebut merepresentasikan setiap kata dalam setiap dokumen (teks) yang dibandingkan dan membentuk sebuah segitiga, sehingga dapat diterapkan hukum kosinus untuk menyatakan bahwa

$$\cos(C) = a^2 + b^2 - c^2/2ab \tag{1}$$

dimana

$$a^2 = {a_x}^2 + {a_y}^2, b^2 = {b_x}^2 + {b_y}^2 \tag{2}$$

Dan

$$c^2 = (b_x - a_x)^2 + (a_y - b_y)^2 \tag{3}$$

Gantikan nilai-nilai tersebut untuk a, b, dan c, sehingga didapatkan:

$$\cos C = \frac{a_x b_x + a_y b_y}{\sqrt{{a_x}^2 + {a_y}^2} \times \sqrt{{b_x}^2 + {b_y}^2}} \tag{4}$$

Ketika dua dokumen identik, sudutnya adalah nol derajat (0°) dan kesamaannya adalah satu (1); dan ketika dua dokumen tidak identik sama sekali, sudutnya adalah 90 derajat (90°) dan kesamaannya adalah nol (0) [4].

2.5 Algoritma *Smith-Waterman*

Algoritma *Smith-Waterman* adalah metode klasik untuk membandingkan dua *string* (rangkaian huruf) dengan sebuah pandangan untuk mengidentifikasi bagian yang sangat mirip di dalamnya. Algoritma ini digunakan secara luas dalam pencarian *near-matches* (kecocokan yang dekat) yang baik, atau yang sering disebut dengan *local alignments*, di dalam jajaran/rentetan biologis.

Dasar dari metode ini adalah skema pemrograman dinamis. Untuk selanjutnya, panjang dari *string* X dan Y akan dinotasikan dengan m dan n. Jika dibayangkan sebuah bagian X' dari *string* X disejajarkan dengan sebuah bagian Y' dari *string* Y, dialokasikan skor yang merepresentasikan "*goodness of fit*" antara X' dan Y'.

Misalkan h adalah nilai positif yang ditambahkan oleh simbol "*hit*", d adalah nilai (negatif) yang ditambahkan oleh penambahan atau penghapusan simbol (sebuah "*indel*"), dan r adalah nilai (negatif) yang ditambahkan oleh substitusi satu simbol oleh yang simbol lain (*replacement*). Kemudian digunakan sebuah matriks skor, yang memberi nilai yang cocok untuk semua kemungkinan *hit* dan penggantian. Untuk model ini, nilai relatif dari h, d, dan r tidaklah pasti; pilihan yang paling jelas adalah menentukan $h = d = r = 1$, dan nilai-nilai tersebut telah terbukti efektif pada kenyataannya.

Untuk memformulasikan skema pemrograman dinamis algoritma Smith-Waterman, didefinisikan S_{ij} sebagai nilai maksimum yang dapat didapatkan dengan menyejajarkan sebuah *substring* dari X yang berakhir di posisi i dengan *substring* Y yang berakhir di posisi j.

Relasi perulangan yang standar untuk S_{ij} adalah

$$S_{ij} = \begin{cases} S_{i-1,j-1} + h \text{ jika } X(i) = Y(j) \\ \max(0, S_{i-1,j} - d, S_{i,j-1} - d, S_{i-1,j-1} - r) \end{cases} \quad (5)$$

yang berlaku kepada kondisi awal

$$S_{i,0} = S_{0,j} = 0 \text{ untuk semua } i, j. \quad (6)$$

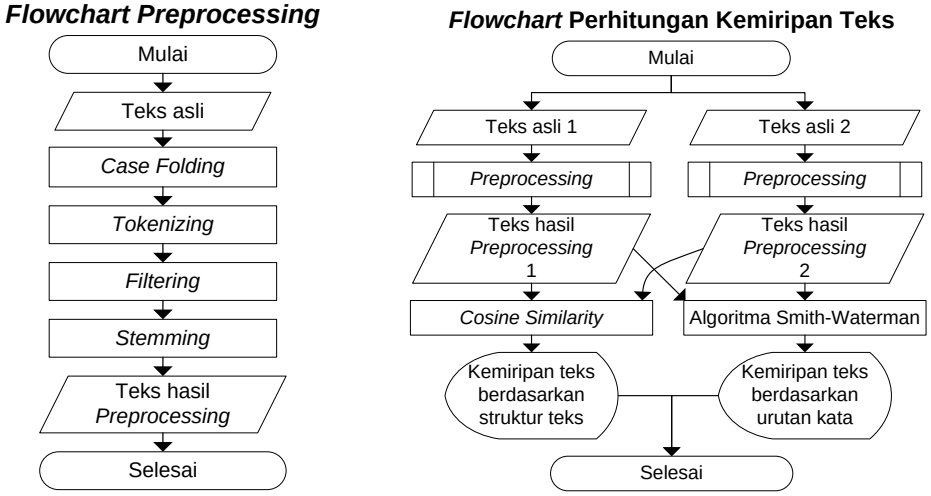
Secara keseluruhan, ukuran kesamaan yang dapat diambil adalah skor kumulatif yang diperoleh dari jumlah *hit*, *indel*, dan *replacement* yang dihitung dari matriks skor yang merupakan nilai terbesar pada matriks skor tersebut [3]. Dua teks atau *string* dapat dikatakan identik seutuhnya jika skor yang diperoleh adalah panjang maksimal dari kedua *string* dikalikan dengan nilai *hit*, dan tidak identik sama sekali jika skor yang diperoleh adalah nol (0). Semakin besar skor, maka semakin mirip kedua teks yang dibandingkan.

3. Analisis dan Desain Aplikasi

Untuk mendeteksi kemiripan teks seperti yang telah dijabarkan pada pendahuluan dan didukung oleh landasan teori, analisis dan desain aplikasi telah dibuat dan dijelaskan sebagai berikut ini.

3.1 Analisis Aplikasi

Analisis aplikasi pada penelitian ini digambarkan dengan *flowchart* langkah-langkah pendeteksian kemiripan teks yang terdiri dari *flowchart preprocessing* dan *flowchart* perhitungan kemiripan teks seperti yang sudah dijelaskan pada landasan teori, ditunjukkan pada gambar 1 di bawah ini.



Gambar 3 *Flowchart* Langkah-langkah Pendeteksian Kemiripan Teks

Flowchart preprocessing pada gambar 1(a) menggambarkan proses *case folding* hingga *stemming* pada sebuah teks. Sedangkan *flowchart* perhitungan kemiripan teks pada gambar 1(b) menggambarkan dua teks asli yang diproses sedemikian rupa hingga aplikasi menghasilkan kemiripan teks berdasarkan kemunculan kata-kata di dalam teks tersebut dan urutan kata-kata yang membentuk kedua teks itu.

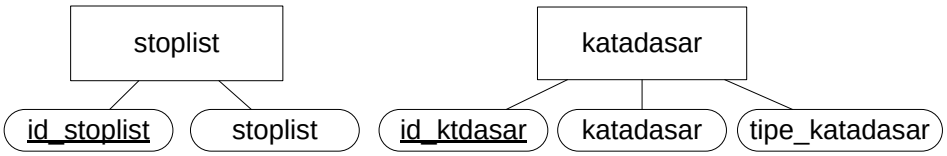
3.2 Desain Aplikasi

Untuk mengembangkan aplikasi pada penelitian ini, desain aplikasi yang berupa *entity relationship diagram* dan *class diagram* aplikasi telah dibuat berdasarkan analisis aplikasi, akan dijelaskan berikut ini.

3.2.1 *Entity Relationship Diagram* (ERD)

ERD aplikasi pada penelitian ini diperlukan untuk mendukung tahap *filtering* dan *stemming* pada langkah satu pendeteksian kemiripan teks seperti yang

sudah digambarkan pada *flowchart preprocessing* yang ditunjukkan oleh gambar 1(a). ERD aplikasi akan ditunjukkan pada gambar 2 di bawah ini.



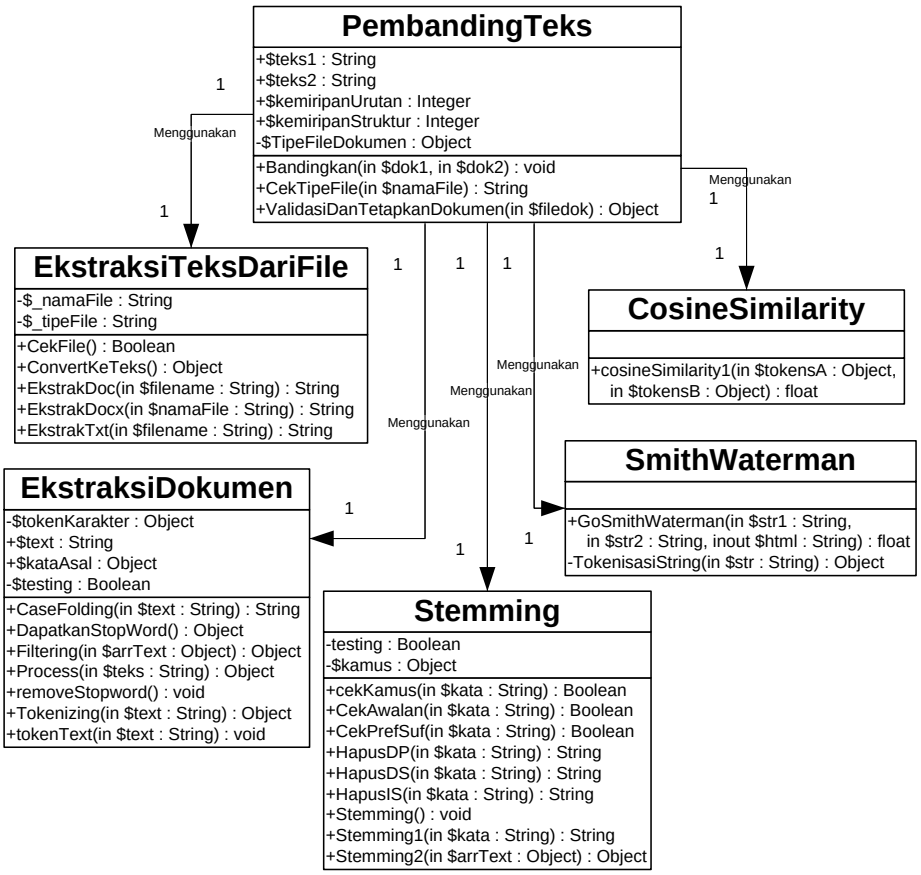
Gambar 4 *Entity Relationship Diagram* Aplikasi

Penjelasan ERD aplikasi pada gambar 2 dijelaskan di bawah ini:

1. Entitas **stoplist** digunakan untuk menyimpan semua kata berbahasa Indonesia yang kurang penting yang akan digunakan dalam tahap *filtering*.
2. Entitas **katadasar** digunakan untuk menyimpan semua kata dasar berbahasa Indonesia yang akan digunakan dalam tahap *stemming*.

3.2.2 *Class Diagram*

Class diagram aplikasi pada penelitian ini dibuat berdasarkan analisis aplikasi dan digunakan sebagai penggambaran *class-class* yang diimplementasikan pada aplikasi untuk menjalankan langkah-langkah pendeteksian kemiripan teks secara keseluruhan. *Class diagram* aplikasi ditunjukkan pada gambar 3 di bawah ini.



Gambar 5 *Class Diagram* Aplikasi

Penjelasan fungsi tiap *class* di dalam *class diagram* aplikasi pada gambar 3 secara garis besar dijabarkan sebagai berikut ini:

1. *Class* **PembandingTeks** digunakan untuk mendeteksi kemiripan teks secara keseluruhan. *Class* ini menginstanisasi *class-class* lainnya untuk menjalankan fungsinya.
2. *Class* **EkstraksiTeksDariFile** digunakan untuk mengekstrak teks dari dalam berkas.txt, .doc, dan .docx.
3. *Class* **EkstraksiDokumen** digunakan untuk menjalankan tahap *preprocessing* (tidak termasuk langkah *stemming*).
4. *Class* **Stemming** digunakan untuk melakukan *stemming*.
5. *Class* **CosineSimilarity** digunakan untuk memperhitungkan kemiripan dua teks berdasarkan kemunculan kata dalam teks tersebut.

6. *Class* **SmithWaterman** digunakan untuk memperhitungkan kemiripan dua teks dengan algoritma Smith-Waterman (berdasarkan urutan kata dalam teks).

4. Hasil Penelitian

Penelitian ini telah melalui pengujian-pengujian yang dilakukan pada aplikasi yang dikembangkan. Berikut ini adalah pengujian yang telah dilakukan dengan menggunakan empat teks contoh di bawah ini:

Teks 1: Pada zaman sekarang dengan dibantunya teknologi yang membantu mempermudah hidup manusia, zaman menjadi berkembang dengan pesat. Tak heran jika hampir semua pekerjaan yang ada telah banyak dibantu oleh komputer.

Teks 2: Pada zaman sekarang, teknologi sangat membantu manusia dan zaman pun berkembang pesat. Tidak heran jika hampir semua pekerjaan yang ada sekarang ini, sudah banyak dibantu teknologi yang bernama komputer.

Teks 3: Pendidikan merupakan sesuatu yang sangat diperlukan oleh setiap orang, dan sudah menjadi kebutuhan utama. Pada era informasi sekarang ini, perkembangan institusi pendidikan tergantung pada kemampuan untuk mengikuti perkembangan teknologi dan kemampuan mengakses serta menyajikan informasi.

Teks 4: Teknologi informasi saat ini berkembang dengan pesat. Dahulu hanya perusahaan besar yang menggunakan teknologi informasi namun dengan berkembangnya kebutuhan dan tuntutan akan data informasi yang semakin komplek maka teknologi informasi mulai memasuki dunia pendidikan.

Keempat teks di atas telah dibandingkan antara satu dengan yang lainnya menggunakan aplikasi ini dan hasil perbandingannya ditunjukkan pada tabel 1.

Tabel 3 Hasil Perbandingan Empat Teks Contoh antara Satu dengan yang Lainnya Berdasarkan Kemunculan dan Urutan Katanya

	Teks 1		Teks 2		Teks 3		Teks 4	
	K*	U*	K*	U*	K*	U*	K*	U*
Teks 1	-		X	X	X	X	X	X
Teks 2	86,86	65.00	-		X	X	X	X
Teks 3	16,37	08,33	23,33	10,00	-		X	X
Teks 4	24,66	16,67	39,54	20,00	60,99	13,33	-	

Keterangan tabel 1:

1. K* = Nilai kemiripan berdasarkan kemunculan kata.
2. U* = Nilai kemiripan berdasarkan urutan kata.
3. - = Tidak diisi, karena kedua teks yang dibandingkan sama (selalu 100%).
4. X = Tidak diisi, karena nilainya sudah ditampilkan sebelumnya. (Contoh: nilai kemiripan teks 1 dengan teks 2 sama dengan nilai kemiripan teks 2 dengan teks 1).
5. Semua nilai kemiripan dibulatkan ke atas sebanyak dua angka di belakang koma dan dalam bentuk %.

Aplikasi hasil penelitian ini juga sudah diimplementasikan dalam sistem informasi *online* pengajuan proposal kerja praktek yang sudah digunakan oleh Fakultas Teknologi Informasi di Universitas Kristen Maranatha.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan hasil penelitian ini, kesimpulan-kesimpulan yang diperoleh adalah:

1. Penelitian telah berhasil mengembangkan sebuah aplikasi yang mengimplementasikan *cosine similarity* dan algoritma Smith-Waterman untuk mendeteksi kemiripan teks.
2. Aplikasi hasil penelitian ini dapat mendeteksi tingkat kemiripan teks dari sangat mirip hingga sangat tidak mirip berdasarkan kemunculan kata di dalamnya dengan menggunakan *cosine similarity*.
3. Aplikasi hasil penelitian ini dapat mendeteksi tingkat kemiripan teks dari sangat mirip hingga sangat tidak mirip berdasarkan urutan kata pembentuknya dengan menggunakan algoritma Smith-Waterman.

5.2 Saran

Saran-saran yang telah diperoleh mengenai pengembangan penelitian selanjutnya adalah mengimplementasikan konsep *singular value decomposition* (SVD) pada aplikasi yang telah dikembangkan.

Daftar Pustaka

[1] Agusta, L. (2009). Perbandingan Algoritma Stemming Porter dengan Algoritma Nazief & Adriani untuk Stemming Dokumen Teks Bahasa Indonesia. *Konferensi Nasional Sistem dan Informatika 2009*; Bali, November 14, 2009.

[2] Asian, J., Williams, H. E. & Tahaghoghi, S. M. M. (2005). Stemming Indonesian. *Proc. Twenty-Eighth Australasian Computer Science Conference (ACSC2005), Newcastle, Australia. CRPIT*, 38, 308.

[3] Irving, R. W. (2004). Plagiarism dan Collusion Detection using the Smith-Waterman Algorithm. *DCS Technical Report*. Diakses terakhir tanggal 22 Maret 2013, dari http://www.dcs.gla.ac.uk/publications/PAPERS/7444/TR-2004-164.pdf

[4] Konchady, M. (2006). *Text Mining Application Programming*. Boston: Charles River Media.

[5] Triawati, C. (2009, Mei). *Metode Pembobotan Statistical Concept Based untuk Klastering dan Kategorisasi Dokumen Berbahasa Indonesia*. Diakses terakhir tanggal 14 Maret 2013, dari http://digilib.ittelkom.ac.id/index.php?option=com_content&view=article&id=590:text-mining&catid=20:informatika&Itemid= 14