IMPLEMENTASI ALGORITMA WINNOWING UNTUK MENDETEKSI KEMIRIPAN PADA DOKUKEN TEKS

Obed Kharisman[1]
obedkharisman@gmail.com

Budi Susanto[2]
budsus@ukdw.ac.id

Sri Suwarno[3]
sswn@ukdw.ac.id

Abstract

Plagiarism is one of the main problems which the academic world must cope with recently. Many student papers and assignments had been found containing lines or sentences directly copied or sourced without sufficient acknowledgments to their original authors.
Based on this fact, this research is conducted by developing software capable of detecting similarity between text documents, using "Winnowing algorithm" which is a document fingerprinting algorithm. The goal of this research is to measure its effectiveness in comparing test documents and reporting their similarity by percentage.

Kata Kunci : *winnowing, similarity, plagiarisme, document fingerprint*

1. Pendahuluan

Semakin mudahnya untuk saling bertukar dan mendapatkan informasi tidak hanya memberikan dampak positif bagi kemajuan teknologi, tetapi juga membawa dampak negatif yang hampir tidak dapat dihindari seperti tindakan *copy paste* yang dapat berujung pada plagiarisme yang sengaja maupun tidak sengaja dilakukan. Tindakan tersebut dapat mematikan kreatifitas seseorang karena sudah terbiasa mengambil sesuatu yang bukan hasil karyanya dengan mudah dan merupakan pelanggaran hak cipta.

Beberapa metode yang dapat digunakan untuk mendeteksi kemiripan dokumen atau menganalis plagiarisme seperti pencocokan *substring,* kesamaan kata kunci dan dokumen *fingprintings*. Algoritma *Winnowing* merupakan algoritma dokumen *fingerprint* yang dapat digunakan untuk mendeteksi kemiripan dokumen teks. *Input* dari algoritma *winnowing* adalah *string* dari sebuah dokumen dan *output* dari dari algoritma tersebut adalah nilai-nilai *hash* yang disebut sebagai *fingerprints* dari dokumen tersebut.

Oleh karena itu perlu dibangun sebuah sistem yang dapat digunakan untuk mendeteksi kemiripan dokumen teks. Dengan mengetahui persentase kemiripan kedua dokumen tersebut dapat dijadikan bahan pertimbangan apakah dokumen yang diuji tersebut merupakan hasil menjiplak karya seseorang atau tidak.

2. Plagiarisme

Plagiarisme dalam Kamus Bahasa Indonesia Kontemporer didefinisikan sebagai perbuatan menjiplak tulisan, ide dan sebagainya milik orang lain; penjiplakan (KBIK, Hal 1172). Sedangkan pada Kamus Besar Bahasa Indonesia menjelaskan bahwa Plagiarisme adalah penjiplakan yang melanggar hak cipta.

Ada beberapa metode untuk mendeteksi atau menganalis plagiarisme (Stein, 2006) seperti Pencocokan Substring (*Substring Matching*) yang mengukur dan membandingkan Substring yang cocok untuk mengidentifikasi jumlah maksimal kecocokan dari dua pasang kalimat yang kemudian digunakan sebagai indikator dari plagiarisme.

---

[1] *Program Studi Teknik Informatika Fakultas Teknologi Informasi Universitas Kristen Duta Wacana*
[2] *Program Studi Teknik Informatika Fakultas Teknologi Informasi Universitas Kristen Duta Wacana.*
[3] *Program Studi Teknik Informatika Fakultas Teknologi Informasi Universitas Kristen Duta Wacana.*

Metode Kesamaan Kata Kunci (*Keyword Similarity*) menggunakan kata kunci (*keyword*) dari setiap dokumen dimana kata kunci tersebut dibandingkan kemudian diukur apakah jumlah atau hasil kemiripan melebihi dari batas yang di tentukan (*Threshold value*).

Dokumen *Fingerprint* (*Fingerprints Analisys*) digunakan untuk mendeteksi keakuratan kesamaan antar dokumen. Prinsip kerja dari metode Analisis *fingerprint* ini dengan menggunakan metode *hashing*. Metode *hashing* adalah sebuah fungsi yang mengubah setiap *string* menjadi bilangan. Beberapa algoritma yang menggunakan metode ini adalah *Winnowing* dan *Rabin-Karp*.

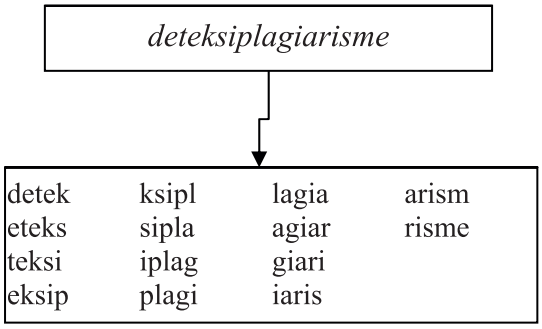
3. Algoritma Winnowing

Algoritma *Winnowing* adalah metode yang meningkatkan efisiensi dari proses perbandingan dokumen *fingerprinting* (Cornic, 2008). *input* dari algoritma ini adalah dokumen teks yang diproses sehingga menghasilkan *output* berupa kumpulan nilai-nilai *hash*, nilai *hash* merupakan nilai numerik yang terbentuk dari perhitungan ASCII dari setiap karakter. Kumpulan-kumpulan nilai *hash* tersebut selanjutnya disebut *fingerprint*. *Fingerprint* inilah yang digunakan untuk membandingkan kemiripan antar dokumen teks.

3.1. Langkah-Langkah

Langkah awal dalam penerapan algoritma *Winnowing* adalah melakukan *lowercase* pada setiap karakter dan membuang karakter-karakter dari dokumen yang tidak relevan misal tanda baca, spasi dan simbol lain.

Langkah kedua, isi dokumen yang telah dilakukan pembersihan pada langkah pertama, dilanjutkan dengan dibentuk kedalam rangkaian *gram*, dimana nilai $k = 5$. seperti terlihat pada gambar 1.



Gambar 1. *Hasil N-Gram terhadap teks*

Langkah ketiga lakukan proses *Rolling Hash* untuk menghasilkan nilai *hash* dari setiap *gram* yang terbentuk. Dari proses tersebut didapatkan nilai *hash* dari setiap *gram* sebagai berikut :

12281, 12658, 13536, 12532, 13161, 13579, 12895, 13275, 12706, 11988, 12498, 12580, 12334, 13532.

Setelah didapatkan nilai *hash* dari setiap *gram*, langkah keempat membentuk *window*. Proses pembentukan *window* sama seperti proses *k-gram* dari nilai-nilai *hash* yang dihasilkan dengan besar *window* $w = 4$:

{**12281**,12658,13536,12532}, {12658,13536,**12532**,13161},
{13536,12532,13161,13579}, {12532,13161,13579,**12895**},
{13161,13579,12895,13275}, {13579,12895,13275,**12706**},
{12895,13275,12706,**11988**}, {13275,12706,11988,12498},
{12706,11988,12498,12580}, {11988,12498,12580,**12334**},
{12498,12580,12334,13532}

Langkah kelima adalah memilih nilai *hash* terkecil dari setiap *window* untuk dijadikan *fingerprint* dokumen tersebut. Dari *window* diatas didapatkan *fingerprints* :

12281, 12532, 12895, 12706, 11988 dan 12334

Dalam beberapa aplikasi akan sangat berguna jika tidak hanya menyimpan *fingerprints* dari sebuah dokumen, tetapi juga posisi dari *fingerprints* dalam dokumen. Sebagai contoh, kita ingin menampilkan informasi posisi dari sub-*string* yang cocok. Implementasi algoritma *Winnowing* yang efisien juga sebaiknya menyimpan posisi dari *fingerprint* yang didapatkan (Elbegbayan, 2005) dimana posisi pertama dimulai dari 0 dari nilai *hash* pada *gram* yang terbentuk sebelumnya [*fingerprint*, *posisi*].

[12281, 0] [12532, 3], [12895, 6], [12706, 8], [11988, 9], [12334, 12]

3.2. Rolling Hash

Rolling Hash merupakan sebuah fungsi yang digunakan untuk menghasilkan nilai *hash* dari rangkaian *gram*. Pada awalnya metode *Rolling Hash* digunakan pada Algoritma *Rabin Karp* dimana metode ini digunakan untuk membandingkan nilai *hashing* dari semua *k-grams* kedalam sebuah *string* yang panjang. Akan tetapi proses *hashing* pada setiap *string* sepanjang *k* akan menghabiskan waktu komputasi yang lama jika nilai *k* besar (Schleimer, dkk. 2003). Untuk itu *Rabin Karp* menggunakan *Rolling Hash* dimana fungsi *hash* $H_{(c_1 \ldots c_k)}$ didefinisikan seperti pada persamaan (I).

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

Keterangan :

c : Nilai ASCII karakter
b : Basis bilangan prima
k : nilai *k-gram* / banyaknya karakter pada gram

Untuk mendapatkan nilai *hash* gram berikutnya $H_{(c_2 \ldots c_{k+1})}$ dapat dilakukan dengan cara seperti pada Persamaan (II).

$$\boldsymbol{H_{(c_2 \ldots c_{k+1})} = \left(H_{(c_1 \ldots c_k)} - c_1 * b^{(k-1)}\right) * b + c_{k+1}} \qquad \textbf{(2)}$$

Pada perhitungan *hash* dari gram ke-*n*, nilai *hash* gram *n-1* dikurangi dengan nilai karakter pertama dari gram *n-1* kemudian ditambahkan dengan nilai karakter terakhir dari *gram* 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 menghemat waktu komputasi saat menghitung nilai *hash* dari sebuah *gram.*

3.3. Perhitungan Kemiripan

Untuk menghitung prosentase kemiripan (*similarity*) dari dokumen tersebut digunakan *Jaccard's Similarity Coefficient* dengan rumus seperti pada Persamaan III.

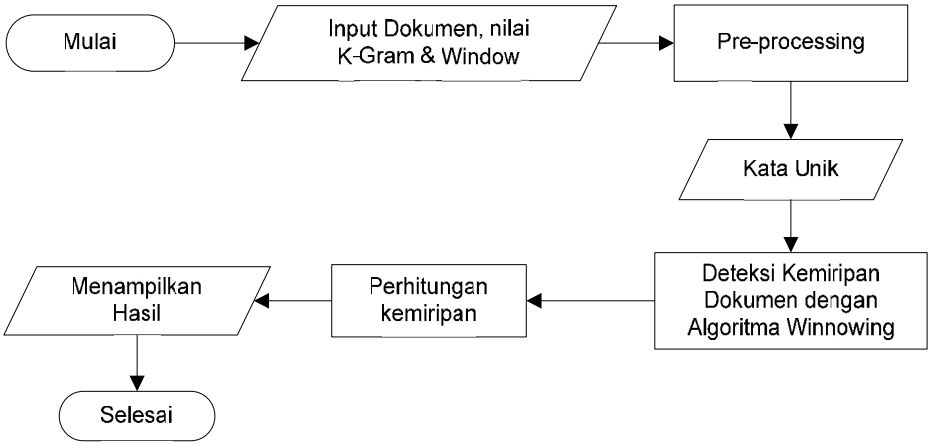
$$D(A,B) = \frac{|A \cap B|}{|A \cup B|} \text{ x } 100 \qquad \textbf{(3)}$$

Dimana D(A,B) adalah nilai *similarity*, |A ∩ B| adalah jumlah dari fingerprints yang sama dari dokumen 1 dan 2 dan |A U B| adalah jumlah *fingerprints* dari dokumen 1 dan dokumen 2. Sebagai contoh a={1,2,4}, b={1,2,4,7,8} maka, |A ∩ B| = {1,2,4} dan |A U B| = {1,2,4,7,8} sehingga

D(a,b) = 3/5 * 100 = 60

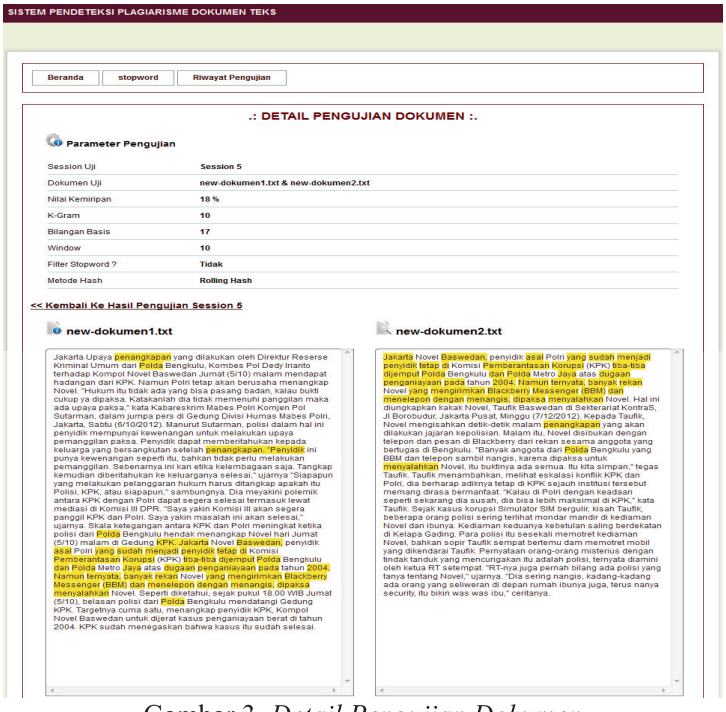
4. Implementasi Sistem

Aplikasi ini berbasis website dan menggunakan database MySQL untuk menyimpan data-data hasil uji coba sistem. Pada Gambar 2 dapat gambaran kerja sistem aplikasi yang akan dibuat.



Gambar 2. *Gambaran Kerja Sistem*.

Aplikasi ini terdiri dari 6 halaman, yaitu halaman Beranda sebagai halaman utama sistem, halaman *Stopword* untuk menampilkan daftar stopword, halaman Riwayat Pengujian untuk menampilkan daftar pengujian yang telah dilakukan, halaman Hasil Pengujian menampilkan hasil dari sebuah sesi pengujian, halaman Detail Dokumen Uji menampilkan secara detail dari dokumen yang akan diuji dan halaman Detail Pengujian Dokumen menampilkan perbandingan hasil pengujian dokumen. Halaman Detail Pengujian Dokumen dapat dilihat pada Gambar 3.



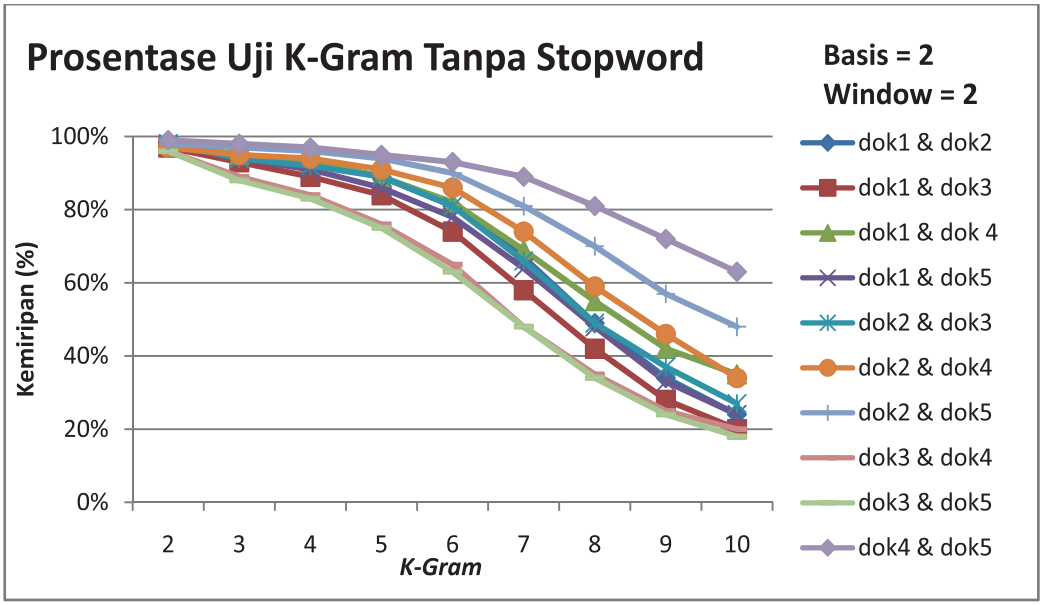
Gambar 3. *Detail Pengujian Dokumen*

5. Analisis Sistem

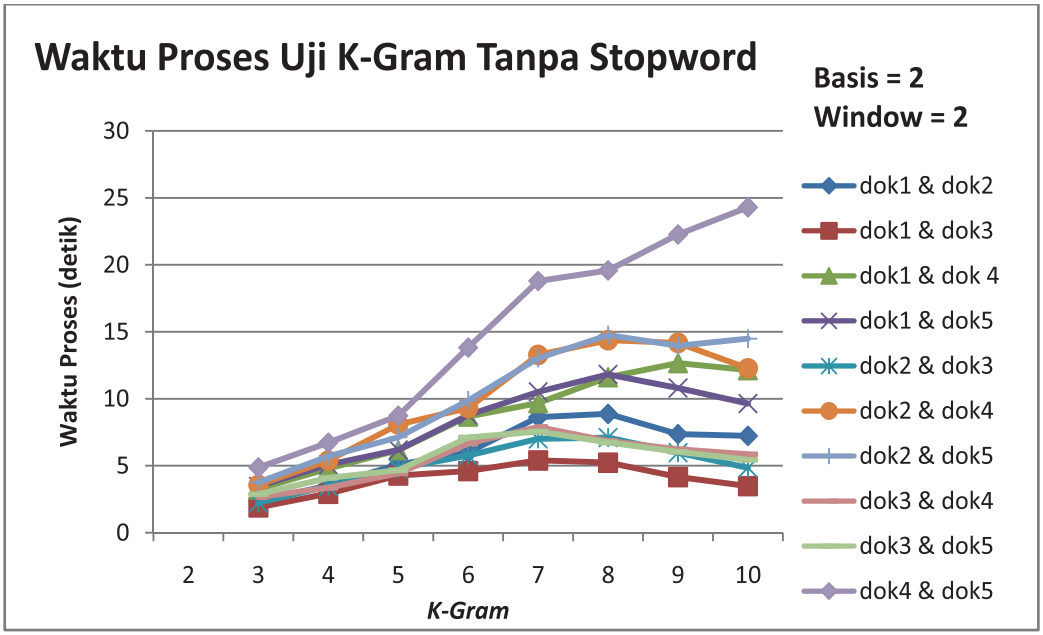
Mekanisme Pengujian dilakukan terhadap kumpulan tugas ORKOM (Organisasi Komputar) Mahasiswa Ilmu Komputer Universitas Nusa Cendana Kupang untuk mendapatkan

besaran nilai parameter yang baik untuk digunakan dengan membandingkan hasil perhitungan sistem dan pengamatan *visual* kemiripan dokumen teks

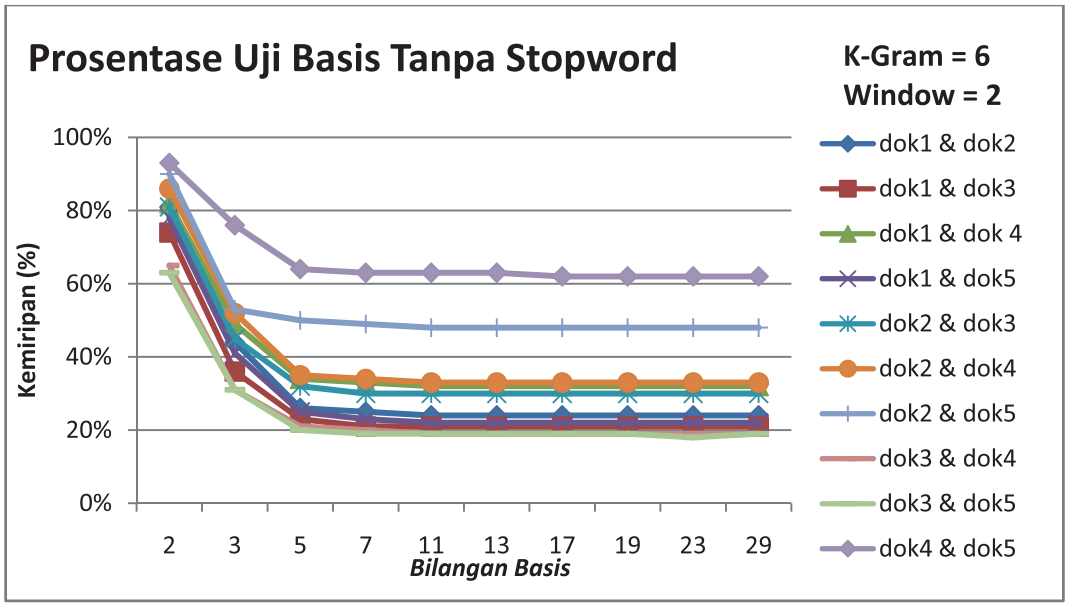
Gambar 4 sampai dengan Gambar 9 adalah grafik-grafik hasil pengujian yang dilakukan terhadap aplikasi dengan memasukan nilai parameter yang berbeda sehingga akan melihat seberapa besar pengaruh perubahan nilai-nilai tersebut terhadap prosentase dan waktu pemrosesan dokumen.



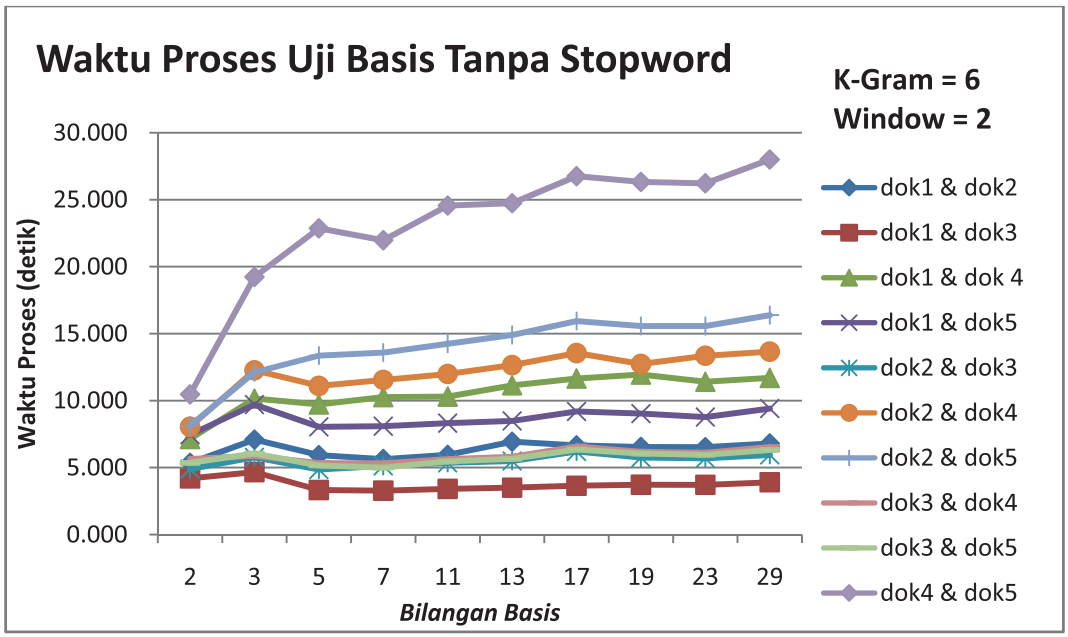
Gambar 4. *Grafik perbandingan prosentase kemiripan dengan pengujian pada nilai k-gram tanpa filtering stopword*



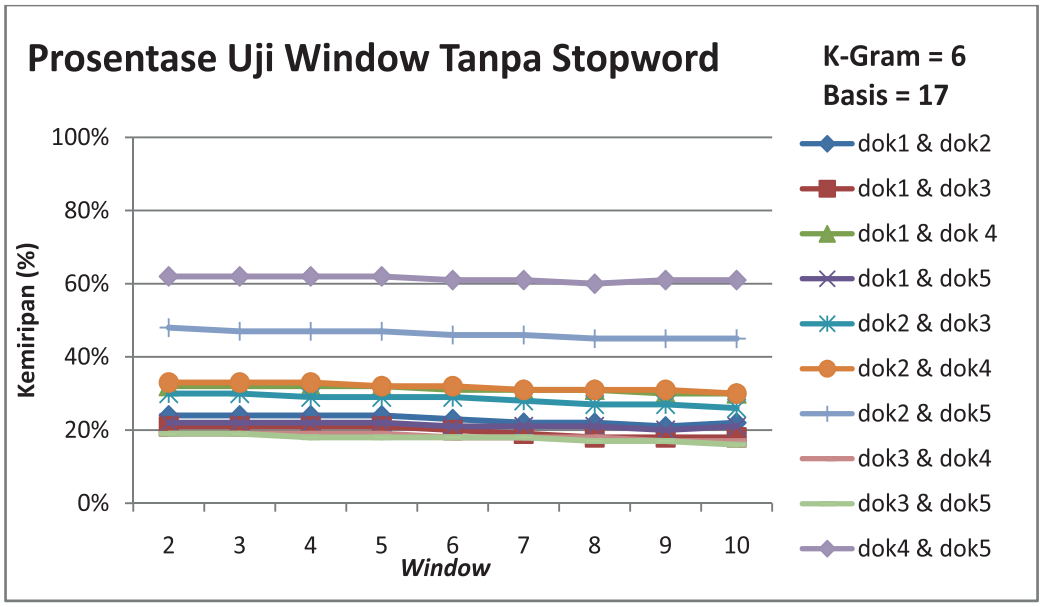
Gambar 5. *Grafik perbandingan waktu proses dengan pengujian pada nilai k-gram tanpa filtering stopword*



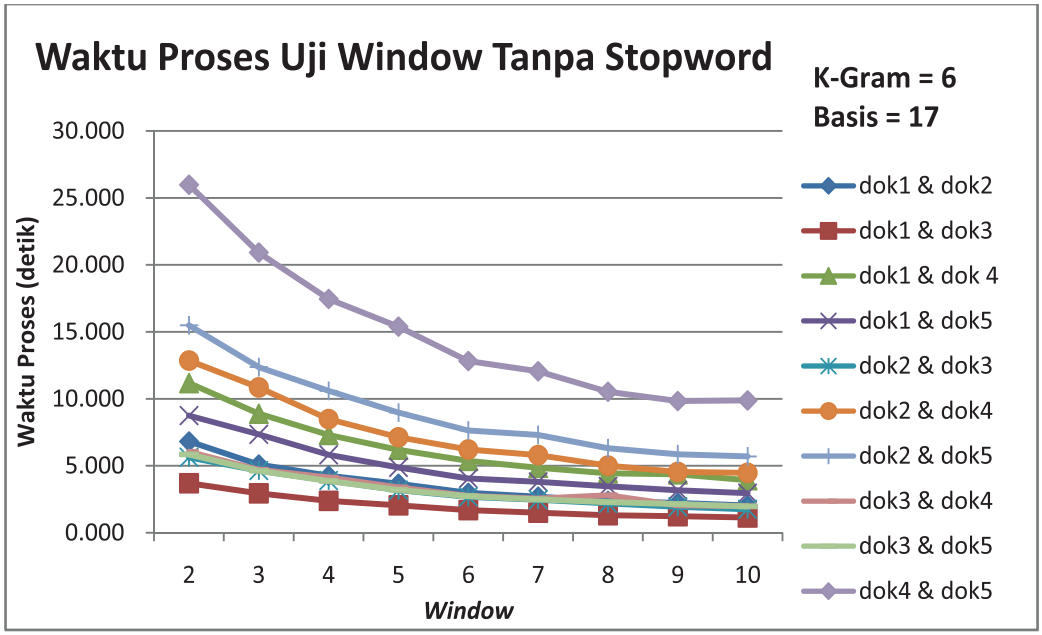
Gambar 6. *Grafik perbandingan prosentase kemiripan dengan pengujian pada nilai basis tanpa filtering stopword*



Gambar 7. *Grafik perbandingan waktu proses dengan pengujian pada nilai basis tanpa filtering stopword*



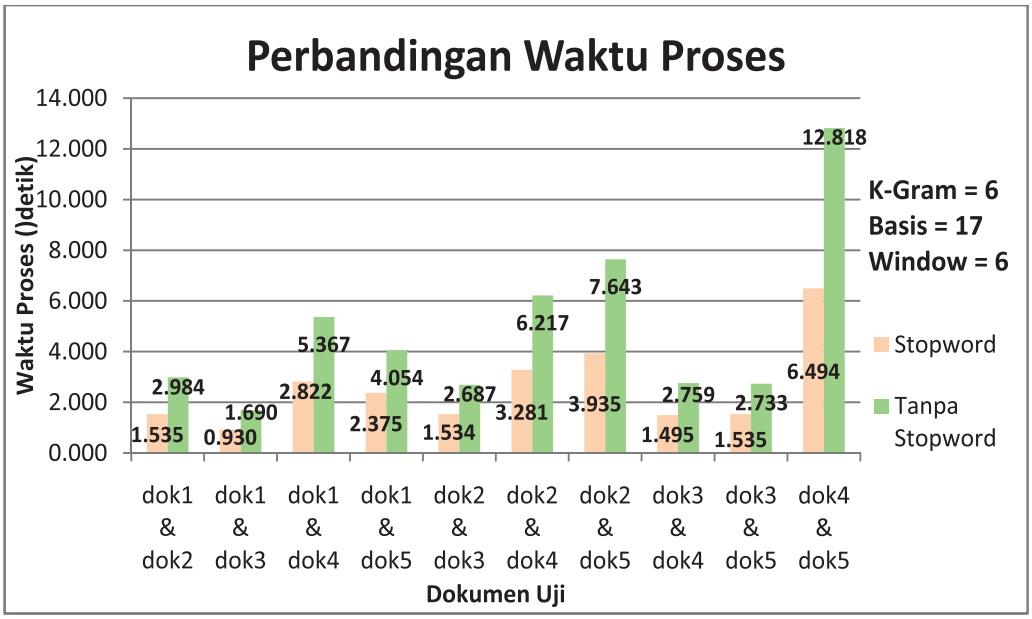
Gambar 8. *Grafik perbandingan prosentase kemiripan dengan pengujian pada nilai window tanpa filtering stopword*



Gambar 9. *Grafik perbandingan waktu proses dengan pengujian pada nilai window tanpa filtering stopword*

Berdasarkan grafik-grafik pengujian yang dilakukan, perubahan nilai *k-gram* dan nilai basis cenderung lebih mempengaruhi prosentase kemiripan antar dokumen, dimana nilai *k-gram* lebih dari 2 dan nilai basis yang lebih besar akan menghasilkan prosentase yang sesuai jika membandingkan kemiripan dokumen tersebut secara *visual.*

Penggunaan nilai *k-gram* yang kecil memungkinan sistem untuk memukannya *gram* yang sama pada dokumen lain dengan sumber kata yang berbeda sehingga akan mempengaruhi prosentase nilai kemiripan yang besar antar dokumen yang diuji. Pada penggunaan nilai basis yang semakin besar, akan mempengaruhi pada proses pembentukan *hashing*, dimana nilai basis yang besar akan menghasilkan nilai *hashing* yang lebih besar sehingga kemungkinan untuk terjadinya tabrakan nilai *hash* (*hash collision*) semakin kecil.



Gambar 10. *Grafik Perbandingan Waktu Proses dengan K-Gram = 6, Basis = 17 dan Window 6.*

Perubahan nilai *window* dan penggunaan *filtering stopword* cenderung mempengaruhi waktu proses yang digunakan untuk membandingkan dokumen yang diuji. Penggunaan nilai *window* yang semakin besar akan mempengaruhi waktu proses dalam pengujian, karena jika semakin besar nilai *window*, maka waktu proses pembetukan *window* dari nilai-nilai *hashing* dokumen akan semakin kecil. Penggunaan filtering *stopword* pada pengujian dokumem memberikan waktu proses yang semakin kecil, hal ini dikarenakan sistem akan mengabaikan kata-kata yang berada pada daftar *stopword*, sehingga sistem hanya mengolah kata-kata penting dalam dokumen untuk dijadikan sebagai acuan pembuatan *fingerprint* dari masing-masing dokumen uji.

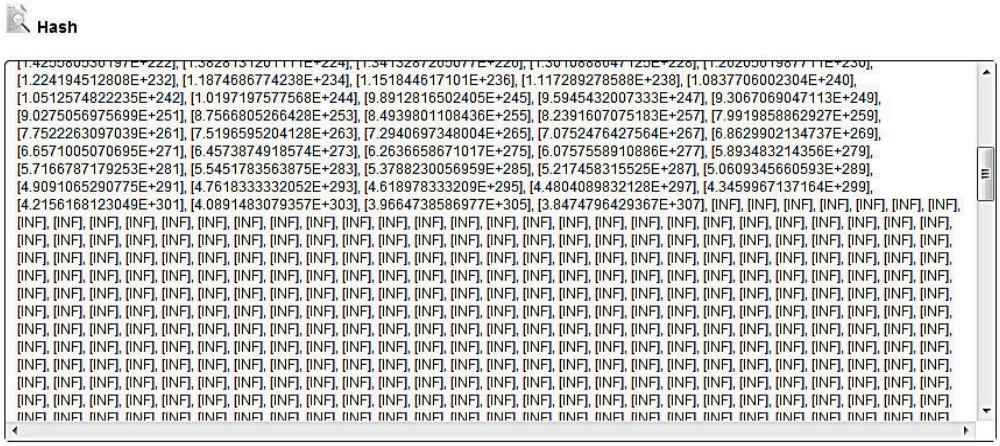
6. Kesimpulan

Penggunaan nilai *k-gram* yang besar akan memberikan prosentase kemiripan yang sesuai jika dilihat secara *visual* dan penggunaan nilai basis yang lebih besar akan memberikan pengaruh terhadap nilai hashing yang dihasilkan sehingga akan berdampak pada nilai kemiripan yang yang didapatkan oleh sistem.

Perubahan nilai *window* dan penggunaan *filtering stopword* cenderung hanya mempengaruhi waktu proses yang digunakan dalam pengujian dokumen. Penggunaan *filtering stopword* juga berfungsi untuk mengurangi waktu proses yang lama.

Kelebihan dari aplikasi ini adalah; aplikasi dapat menampilkan hasil kemiripan kedua dokumen secara visualisasi, sehingga *user* dapat melihat letak kemiripan dokumen yang dibandingkan dan aplikasi ini menampilkan statistik informasi yang jelas mulai dari proses *preprocessing* dokumen hingga proses perbandingan dokumen teks.

Kelemahan dari aplikasi ini terletak pada proses pembuatan *Rolling Hash*, dimana jika kita menggunakan *formula Rolling Hash* I dan *Rolling Hash* II dan memberikan nilai basis yang besar (lebih dari 40), maka akan memberikan hasil INF (*infinity*) pada proses pembentukan nilai *Hash*. Gambar 10 dapat dilihat hasil INF yg dihasilkan oleh *Rolling Hash* untuk nilai basis lebih dari 40.



Gambar 11. *Hasil Rolling Hash nilai INF*

Sistem menghasilkan nilai INF karena terjadi proses pengurangan dan pengkalian pada *formula Rolling Hash II* dan bahasa pemrograman PHP tidak dapat memproses nilai *integer* yang sangat besar.

$$\boldsymbol{H_{(c_2 \ldots c_{k+1})} = \left(H_{(c_1 \ldots c_k)} - c_1 * b^{(k-1)}\right) * b + c_{k+1}} \qquad \textbf{(2)}$$

Untuk menghindari sistem menghasilkan nilai *hash* INF, maka cukup digunakan *formula Rolling Hash I* untuk pembentukan nilai hash pada masing-masing *k-gram*. Selain menggunakan *formula Rolling Hash I*, sistem juga diberi batasan untuk menggunakan nilai basis 2 – 30.

Daftar Pustaka

Ceska, Z. & Fox, C. (2009). The Influence of Text Pre-processing on Plagiarism Detection. In Angelova, Galia and Bontcheva, Kalina and Mitkov, Ruslan and Nicolov, Nicolas and Nikolov, Nikolai, (Eds.) *International Conference on Recent Advances in Natural Language Processing 2009* (pp. 55-59). Borovets, Bulgaria: Association for Computational Linguistics.

Cornic, P. (2008). *Software Plagiarism Detection Using Model-Driven Software Development in Eclipse Platform,* (Thesis, Computer Science, Faculty Engineering and Physical Sciences, University of Manchester). Diakses dari http://studentnet.cs.manchester.ac.uk/resources/library/thesis_abstracts/MSc08/Abstracts/CornicPierre-fulltext.pdf.

Elbegbayan, N. (2005). *Winnowing, a Document Fingerprinting Algorithm*, (TDDC03 Projects, Spring 2005). Linkoping University, Swedia.

Penjiplakan [Def.] (2005). *Kamus Besar Bahasa Indonesia* (Edisi ketiga). Jakarta: Balai Pustaka.

Nugroho, E. (2011). *Perancangan Sistem Deteksi Plagiarisme Dokumen Teks Dengan Menggunakan Algoritma Rabin-Karp* (Skripsi S1, Program Studi Ilmu Komputer, Jurusan Matematika, Fakultas Matematikan dan Ilmu Pengetahuan Alam, Universitas Brawijaya). Diakses dari http://blog.ub.ac.id/ecoorner/files/2011/03/Bab12345.pdf.

Peraturan Menteri Pendidikan Nasional Republik Indonesia Nomor 17 Tahun 2010. (2010). *Pencegahan dan Penanggulangan Plagiat di Perguruan Tinggi*.

Sastroasmoro, S. (Agustus, 2007). Beberapa Catatan Tentang Plagiarisme. *Majalah Kedokteran Indonesia*, 57(8), 239-244.

Scheleimer, S, & Wilkerson, D.S., & Aiken, A. (2003). Winnowing : Local Algorithms for Document Fingerprinting. Dalam *Proceedings of the 2003 ACM SIGMOD international conference on Management of data* (SIGMOD '03). ACM, New York, NY, USA, 76-85.

Soelistyo, H. (2011). *Plagiarisme : Pelanggaran Hak Cipta dan Etika*. Yogyakarta: Penerbit Kanisius.

Stein, B., & Eissen, S. M.Z. (2006). Near Similarity Search and Plagiarism Analysis. Dalam Spiliopoulou et.al. (Eds). *Data and Information Analysis to Knowledge Engineering Selected Papers from the 295th Annual Conference of German Classification Society (GFKI)*, pp. 430-437. Magdeburg : Springer.