IMPLEMENTASI ALGORITMA RABIN-KARP UNTUK MENDETEKSI DUGAAN PLAGIARISME BERDASARKAN TINGKAT KEMIRIPAN KATA PADA DOKUMEN TEKS

Pingky Alfa Ray Leo Lede [1], Adriana Fanggidae [2], dan Yulianto T. Polly [3]

[1,2,3] Jurusan Ilmu Komputer, Fakultas Sains dan Teknik, Universitas Nusa Cendana

ABSTRAK

Plagiarisme merupakan tindakan mengutip ataupun menafsirkan pemikiran-pemikiran atau pekerjaan orang lain yang dicakupkan atau dimasukkan ke dalam sebuah karya dengan metode penulisan yang kurang tepat atau tidak sesuai dengan disiplin atau aturan akademik yang berlaku. Cara untuk mengurangi plagiarisme bisa dilakukan dengan mencegah maupun mendeteksinya menggunakan tingkat kemiripan sebagai bahan pertimbangan. Pemanfaat Rabin-Karp sebagai algoritma pencocokan *string* dengan metode pencarian jamak mampu untuk mendeteksi dugaan plagiarisme berdasarkan tingkat kemiripan. Hasil dari penelitan ini adalah pendekatan matematis untuk mendapatkan nilai k-grams dan *modulo* yang ideal. Fungsi *modus* dipilih untuk menghitung nilai k-grams pada proses *parsing* dan pendekatan bilangan prima terbesar untuk menghitung nilai *modulo* pada proses *hashing* sesuai dengan hasil pengujian terhadap 7 *data set* dengan 2 varian dan 3 kali perlakuan baik dengan ataupun tanpa *stemming*. Hasil pengujian menunjukan bahwa fungsi *modus* mampu menghasilkan rata-rata nilai *similarity* 3,544% di atas fungsi *mean* dan 2,859% di atas fungsi *median* pada pengujian tanpa menggunakan *stemming*, dan 2,456% di atas fungsi *mean* serta 0,75% di atas fungsi *median* untuk pengujian yang menggunakan *stemming*, sedangkan pendekatan bilangan prima terbesar memiliki rata-rata selisih waktu yang lebih baik yaitu 2,04 detik terhadap sistem *manual* dengan nilai *modulo* yang kecil tanpa menggunakan *stemming*, dan 0,13 detik terhadap sistem *manual* yang menggunakan *stemming*, sedangkan dengan menggunakan nilai *modulo* yang sama sistem manual memiliki durasi yang lebih baik yaitu 0,07 detik dibandingkan sistem otomatis tanpa menggunakan *stemming* dan 0,06 detik pada pengujian yang menggunakan *stemming*, namun sistem yang menggunakan perhitungan bilangan prima tetap efektif karena mampu mencegah terjadinya *quadratic*.

Kata kunci : plagiarisme, sistem otomatis, k-grams ideal, *modulo* ideal, *parsing*, *hashing*, *stemming*, Rabin-Karp.

ABSTRACT

Plagiarism is an action of quoting or interpreting the thoughts or works of others and include them into one's work using improper writing method which is not in accordance with academic discipline or regulation. Possible methods to reduce plagiarism are by prevention it or by detecting the level of similarity as consideration. Use of Rabin-Karp as string matching algorithm with multiple searching method can detect plagiarism suspect based on level of similarity. The result of this research is a mathematical approach to obtain ideal k-grams and modulo values. Modus function was chosen to calculate the k-grams value during parsing process and largest prime number approach was used to calculate modulo value during hashing process according to the result of test of 7 data sets with 2 variant and 3 treatments with and without stemming. Test result shows that modus function is capable of producing average similarity value of 3.544% above mean function and 2.859% above median function without stemming, and 2.456% above mean function as well as 0.75% above median function in with stemming. Meanwhile, the largest prime number approach yielded a better time-difference average of 2.04% seconds in comparison to manual system and small modulo value without using stemming; 0.13% seconds when using stemming. On the other hand, using the same modulo values, the manual system yielded better duration of 0.07 without stemming and 0.06 seconds with stemming. However, the system which employs prime number calculation, remains effective since it can reduce quadratic.

***Keywords* : Plagiarism, automatic system, ideal k-grams, ideal *modulo*, *parsing*, *hashing*, *stemming*, Rabin-Karp**

ISSN 2337-7631

ISSN 2337-7631

I. PENDAHULUAN

Plagiat merupakan sebuah tindakan pengambilan ide, gagasan, pernyataan, atau karya orang lain dan menjadikannya seolah-olah karangan atau karya sendiri [1]. Menurunnya tingkat kreativitas serta rasa keingintahuan adalah salah satu pengaruh atau akibat dari tindakan tersebut. Plagiat juga merupakan salah satu jenis tindakan pencurian intelektual dengan konsekuensi yang berat bagi para pelakunya, baik yang dengan atau tanpa sengaja melakukannya [2], dan tindakan tersebut sudah banyak terjadi khususnya di dunia pendidikan oleh kalangan-kalangan pelajar; baik siswa maupun mahasiswa dalam penulisan karya ilmiah yang disebabkan oleh kemajuan teknologi untuk memperoleh informasi yang semakin mudah di saat sekarang ini.

Plagiarisme dapat di kelompokkan dalam beberapa bentuk [3], antara lain:

a. Plagiarisme kata per kata; Penyalinan kalimat secara langsung dari sebuah dokumen teks tanpa adanya pengutipan atau perizinan.
b. Plagiarisme authorship; Mengakui hasil karya orang lain sebagai hasil karya sendiri dengan cara mencantumkan nama sendiri menggantikan nama pengarang yang sebenarnya.
c. Plagiarisme ide; Penggunaan ulang suatu gagasan/pemikiran asli dari sebuah sumber teks tanpa bergantung bentuk teks sumber.
d. Plagiarisme sumber sekunder; Perbuatan mengutip kepada sumber asli yang didapat dari sumber sekunder dengan menhiraukan teks asli dari sumber yang sebenarnya.
e. Plagiarisme struktur sumber; Penyalinan/penjiplakan struktur suatu argument dari sebuah sumber.
f. Plagiarisme paraphrase; Penulisan ulang dengan mengubah kata atau sintaksis, tetapi teks aslinya masih dapat dikenali.

Beberapa faktor yang dapat digunakan untuk mengidentifikasi plagiarisme [4] antara lain:

1) Penggunaan kosakata; Menganalisis kosakata yang digunakan dalam suatu tugas terhadap pengguna kosakata sebelumnya dapat membantu menentukan apakah mahasiswa benar-benar telah menulis teks tersebut, dengan menemukan suatu kosakata baru dalam jumlah yang besar (terutama kosakata lanjut) dapat menentukan apakah mahasiswa menulis teks tanpa melakukan plagiarisme.
2) Perubahan kosakata; Apabila penggunaan kosakata berubah secara significant dalam suatu teks, hal ini dapat mengindikasikan plagiarisme dengan cara copy dan paste.
3) Teks yang membingungkan; Apabila alur dari suatu teks tidak halus dan tidak konsisten, hal ini mengindikasikan penulis tidak menulis menggunakan pemikirannya sendiri atau beberapa bagian dari tulisannya bukanlah hasil karyanya.
4) Penggunaan tanda baca; Tidak wajar apabila dua orang penulis menggunakan tanda baca yang persis sama dalam membuat suatu karya tulis.
5) Jumlah kemiripan teks; Pasti ada beberapa kemiripan antara beberapa teks yang menulis dengan topik yang sama seperti nama-nama, istilah-istilah dan sebagainya. Bagaimanapun, tidak wajar bila beberapa teks yang berbeda memiliki kesamaan atau kemiripan teks dalam jumlah yang besar.
6) Kesalahan ejaan yang sama; Merupakan hal yang biasa terjadi bagi seorang penulis dalam membuat suatu karya tulis. Menjadi tidak wajar bila beberapa teks yang berbeda memiliki kesalahan-kesalahan ejaan yang sama dalam pengejaan atau jumlah ejaan salah yang sama.
7) Distribusi kata-kata; Tidak wajar apabila distribusi penggunaan kata dalam teks yang berbeda memiliki kesamaan. Sebagai contoh, suatu teks memiliki parameter yang sama untuk suatu distribusi statistik yang digunakan untuk menjelaskan penggunaan istilah.
8) Struktur sintaksis teks; Hal ini menunjukan plagiarisme mungkin saja telah terjadi jika dua teks secara jelas memiliki kesamaan struktur sintaksis. Hal yang wajar apabila penggunaan struktur sintaksis yang digunakan oleh beberapa penulis akan berbeda.
9) Rangkaian-rangkaian panjang kata yang sama; Tidak wajar apabila suatu teks yang berbeda (bahkan yang menggunakan judul yang sama) memiliki rangkaian/urutan karakter yang sama.
10) Orde kemiripan antar teks; Hal ini bisa mengindikasikan plagiarisme apabila orde kecocokan kata atau frase antar dua teks sama. Meskipun diajarkan untuk menyajikan fakta-fakta dalam suatu

aturan (contohnya pendahuluan, isi, kemudian kesimpulan), kurang wajar jika fakta-fakta yang sama dilaporkan dalam orde yang sama.

11) Ketergantungan pada kata atau frase tertentu; Seorang penulis mungkin memilih penggunaan suatu kata atau frase tertentu. Kekonsistenan penggunaan kata-kata tersebut dalam suatu teks yang ditulis oleh orang lain dengan menggunakan kata yang berbeda dapat mengindikasikan plagiarisme.
12) Frekuensi kata; Tidak wajar apabila kata-kata dari teks yang berbeda digunakan dengan frekuensi yang sama.
13) Keputusan untuk menggunakan kalimat panjang atau kalimat pendek; Tanpa sepengetahuan kita, para penulis tentu memiliki keputusan penggunaan panjang kalimat yang tidak biasa dikombinasikan dengan fitur-fitur lain.
14) Teks yang dapat dibaca; Penggunaan metrik/ukuran seperti index Gunning FOG, Flesch Reading Ease Formula atau SMOG dapat membantu menentukan suatu skor kemampuan. Tidak wajar apabila penulis yang berbeda akan memiliki skor yang sama.
15) Referensi yang tidak jelas; Apabila referensi yang muncul dalam suatu teks tetapi tidak terdapat pada daftar pustaka, hal ini dapat mengindikasikan plagiarisme *cut* and *paste*, dimana penulis tidak menyalin referensinya secara lengkap.

Algoritma Rabin-Karp sendiri merupakan salah satu algoritma yang digunakan untuk mendeteksi maupun mencegah terjadinya plagiat. Algoritma ini menggunakan fungsi *hashing* sebagai karakteristik pencocokan *string*; fungsi *hashing* tersebut merupakan gabungan antara aturan Horner dan *modulus* yang akan menghasilkan sebuah angka unik untuk mewakili kata tersebut dalam proses pencocokan *string*, dan fungsi *hashing* yang baik haruslah menghasilkan angka unik yang tidak sama atau harus berbeda dengan angka yang mewakili kata lainnya sehingga tidak terjadi *quadratic* atau kata yang berbeda tetapi memiliki hasil *hashing* yang sama; angka tersebut bisa didapat dari sisa hasil bagi dengan sebuah bilangan prima yang cukup besar, sehingga dapat meningkatkan kompleksitas waktu dalam pencarian *string*. Selain dari fungsi *hashing*, algoritma Rabin-Karp juga melakukan proses *parsing* pada *preprocessing*-nya, dan *parsing* yang baik harus bisa mem-*parsing* setiap kata yang tersusun dalam sebuah kalimat atau memiliki *n-grams* yang ideal untuk melakukan proses *parsing*.

Penelitian ini bertujuan untuk menghasilkan sistem pendeteksi dugaan plagiarisme menggunakan algoritma Rabin-Karp yang mampu menghasilkan nilai *modulo* serta *grams* secara otomatis dengan kualitas yang lebih baik untuk proses *parsing* dan *hashing* dengan atau tanpa menggunakan *stemming* sebagai *preprocessing*, dimana nilai kecocokan yang dihasilkan bukan hanya mengeluarkan tingkat kemiripan semata tetapi juga tingkat kepercayaan terhadap hasil dari sistem bagi penggunanya dalam menentukan keputusan.

II. MATERI DAN METODE

2. 1. Preprocessing

Preprocessing adalah tahap awal mempersiapkan dokumen teks untuk pengolahan lebih lanjut dengan tujuan meningkatkan performa serta kualitas dari hasil ekstrasi dokumen teks. Pada tahap *preprocessing* terdapat 3 proses utama antara lain:

a. *Case folding*

Tujuan dari tahap ini adalah mengkonversi semua huruf kapital menjadi huruf kecil sehingga bersifat *uncase sensitive*. Contohnya kalimat “SAYA MAKAN NASI“ menjadi “saya makan nasi”.

b. *Filtering*

Filtering merupakan tahap pengambilan kata-kata penting dari token menggunakan algoritma *stoplist* (membuang kata yang kurang penting). *Stoplist/stopword* adalah kata-kata yang tidak deskriptif yang dapat dibuang dalam pendekatan *bag-of-words*. Contohnya kata “yang”, “di”, “dan”, “dari”, dan seterusnya.

ISSN 2337-7631

c. *Stemming*

Stemming adalah proses mengembalikan berbagai macam bentukan kata ke dalam representasi yang sama. Metode *stemming* memerlukan masukan berupa kata yang tepat dalam suatu dokumen, dengan menghasilkan keluaran berupa *root word*. Menurut Jelita Asian berdasarkan aturan morfologi bahasa Indonesia dapat dinyatakan bahwa algoritma Nazief dan Adriani adalah algoritma yang memiliki hasil terbaik.Metode *stemming* yang dipakai adalah *stemming* Nazief dan Adriani, mereka menyimpulkan sebuah kata dasar dapat ditambahkan imbuhan berupa *derivation prefix* (DP) di awal dan/atau diakhiri secara berurutan oleh *derivation suffix* (DS), *possessive pronoun* (PP) dan *particle* (P) yang masing-masing bersifat optional. Keterangan tersebut dapat dirumuskan sebagai berikut:

DP+DP+DP+root word+DS+PP+P

Langkah-langkah yang digunakan algoritma Nazief dan Adriani adalah sebagai berikut:

1. Kata dicari di dalam kamus kata dasar. Bila kata tersebut ditemukan di dalam kamus, maka dapat diasumsikan kata tersebut adalah kata dasar sehingga algoritma dihentikan.
2. Bila kata di dalam langkah pertama tidak ditemukan di dalam kamus, maka diperiksa apakah sufiks tersebut adalah sebuah partikel (“-lah” atau “-kah”). Bila ditemukan maka partikel tersebut dihilangkan.
3. Pemeriksaan dilanjutkan pada kata ganti milik (“-ku”, “-mu”, “-nya”). Bila ditemukan, maka kata ganti tersebut dihilangkan.
4. Memeriksa akhiran (“-i”, “-an”). Bila ditemukan, maka akhiran tersebut dihilangkan.
5. Memeriksa awalan (“se-“, “ke-“, “di-“, “te-“, “be-“, “pe-“, “me-“). Bila ditemukan, maka awalan tersebut dihilangkan. Pemeriksaan dilakukan dengan berulang mengingat adanya kemungkinan *multi-prefix*. Langkah ke-5 ini juga membutuhkan ketelitian untuk memeriksa kemungkinan peluluhan awalan (Tabel 1), perubahan *prefix* yang disesuaikan dengan huruf-awal kata (Tabel 2) dan aturan kombinasi *prefix-suffix* yang diperbolehkan (Tabel 3).
6. Setelah menyelesaikan semua langkah dengan sukses, maka algoritma akan mengembalikan kata dasar yang ditemukan.

Proses *stemming* yang dilakukan sesuai dengan aturan morfologi bahasa Indonesia yang ada. Berikut adalah daftar prefiks yang meluluh, kemungkinan perubahan prefiks yang terjadi serta kombinasi prefiks dan sufiks yang tidak diperbolehkan.

Tabel 1. Daftar Prefiks yang Meluluh

Jenis Prefiks	Huruf	Hasil Peluluhan
pe-/me-	k	-ng-
pe-/me-	p	-m-
pe-/me-	s	-ny-
pe-/me-	t	-n-

Tabel 2. Daftar Kemungkinan Perubahan Prefiks

Prefiks	Perubahan
se-	tidak berubah
ke-	tidak berubah
di-	tidak berubah
be-	ber-
te-	ter-
pe-	per-, pen-, pem-, peng-
me-	men-, mem-, meng-

Tabel 3. Daftar Kombinasi Prefiks dan Sufiks yang Tidak Diperbolehkan

Prefiks	Sufiks yang tidak diperbolehkan
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i, -kan
te-	-an
pe-	-kan

2. 2. Fungsi *Modus*

Algoritma Rabin-Karp menggunakan proses *parsing* dan *hashing* sebagi *preprocessing* sebelum masuk dalam tahap pencocokan *string*. Proses *parsing* merupakan proses pembangkit kata sesuai dengan besarnya nilai *grams*, proses tersebut akan mempengaruhi nilai *similarity* hasil pencocokan *string*; semakin besar nilai *grams* maka hasil dari proses *parsing* akan semakin memperhatikan struktur kata penyusun kalimat dibandingkan kata-kata yang tersusun di dalamnya, sebaliknya semakin kecil nilai *grams* maka hasil proses *parsing* akan semakin memperhatikan kata-kata dalam kalimat dibandingkan struktur kata dari kalimat tersebut, untuk itu diperlukan nilai k-grams yang cukup ideal

dimana nilai *grams* tersebut adalah nilai yang mampu menghasilkan representasi kata-kata terbanyak dalam kalimat dibandingkan dengan nilai *grams* lainnya.

Pendekatan matematis yang dipilih untuk menghitung nilai *grams* adalah fungsi *modus* karena hasil dari *preprocessing* sebelum proses *parsing* adalah himpunan kata-kata di dalam dokumen teks dan nilai *modus*-nya merupakan frekuensi panjang kata terbesar dalam himpunan tersebut.

Berikut adalah *pseudo-code* dari perhitungan nilai *grams*:

```
    -code dari perhitungan nilai grams:
1   Fungsi modus(himpunan_kata: ArrayDin; var
2   jumlah: integer): integer;
3   var
4     temp_jumlah, temp_modus, i, j: integer;
5   mulai fungsi
6     temp_jumlah := 0;
7     temp_modus := 0;
8     jumlah := 0;
9     modus := 0;
10    lakukan perulangan i sebanyak
11    length(himpunan_kata)-1
12    mulai
13      jika i = 0 maka
14      mulai
15        j := -1;
16        temp_modus := length(himpunan_kata[i]);
17        lakukan perulangan
18          inc(j);
19          jika temp_modus =
20          length(himpunan_kata[j]) maka
21            inc(temp_jumlah);
22        sampai j >= length(himpunan_kata) - 1;
23    modus := temp_modus;
24        jumlah := temp_jumlah;
25      akhir baris 13
26    jika tidak (i > 0) dan (temp_modus <>
27      length(himpunan_kata[i])) maka
28      mulai
29        j := -1;
30        temp_modus := length(himpunan_kata[i]);
31        temp_jumlah := 0;
32        lakukan perulangan
33          inc(j);
34          jika temp_modus =
35          length(himpunan_kata[j]) maka
36            inc(temp_jumlah);
37        sampai j = length(himpunan_kata) - 1;
38        jika temp_jumlah > jumlah maka
39        mulai
40          modus := temp_modus;
41          jumlah := temp_jumlah;
42        akhir baris 35;
43      akhir baris 25;
44    akhir baris 11;
45  akhir fungsi;
```

2. 3. Pendekatan Bilangan Prima Terbesar

Tujuan dari pendekatan bilangan prima terbesar adalah menghasilkan representasi angka hasil *hashing* pola kata agar tidak terjadi *quadratic* (hasil *hashing* yang sama dengan pola kata yang berbeda), dengan adanya bilangan prima tersebut maka kompleksitas waktu pencocokan teks menggunakan algoritma Rabin-Karp akan mencapai kompleksitas waktu maksimum yaitu $O(n + k)$.

Berikut adalah *pseudo-code* dari pendekatan bilangan prima terbesar untuk mencari nilai modulo pada proses *hashing*:

```
Fungsi cari_modulo(asli: ArrayDin; gram:
integer): integer;
var
  i, j, n, max, hasil: integer;
  kata: string;
  cek: boolean;
mulai fungsi
  max := 0;
  mulai perulangan i sebanyak
  panjang(asli)lakukan
    hasil := 0;
    kata := asli[i];
    n := length(kata);
    mulai perulangan j sebanyak n lakukan
      hasil := hasil + (ord(kata[j]) *
      pangkat(10, n - j));
    akhir perulangan j;
    jika i = 0 maka max := hasil
    jika tidak maka
     jika i > 0 maka
       jika hasil > max maka max := hasil;
    akhir perulangan i;
  n := 0;
    i := 1;
    cek := false;
    lakukan perulangan
      inc(n);
      jika max mod n = 0 maka inc(i);
      jika n = 9 maka
        jika i > 2 maka
        mulai
          n := 0;
          max := max + 1;
          i := 1;
        akhir baris 28
        kalau tidak jika i = 2 maka
          cek := true;
    sampai (n >= 9) atau cek;
    result := max;
  akhir fungsi;
```

2. 4. K-grams

K-grams adalah rangkaian terms dengan panjang *k*. Kebanyakan yang digunakan sebagai *terms* adalah kata. *K-grams* merupakan sebuah metode yang diaplikasikan untuk pembangkitan kata atau

ISSN 2337-7631

karakter. Metode *k-grams* ini digunakan untuk mengambil potongan-potongan karakter huruf sejumlah *k* dari sebuah kata yang secara kontinuitas dibaca dari teks sumber hingga akhir dari dokumen.
Berikut ini adalah contoh k-grams dengan k=5:

a. Text: Sistem pendeteksi dugaan plagiat
b. Kemudian dilakukan penghilangan spasi:
   sistempendeteksidugaanplagiat
c. Sehingga dihasilkan rangkaian 5-grams yang diturunkan dari text sebagai berikut:
   siste istem stemp tempe empen mpend pende endet ndete detek eteks teksi eksid ksidu sidug iduga dugaa ugaan gaanp aanpl anpla nplag plagi lagia agiat

2. 5. Hashing

Hashing adalah suatu cara untuk mentransformasi sebuah string menjadi suatu nilai yang unik dengan panjang tertentu (*fixed-length*) yang berfungsi sebagai penanda string tersebut. Fungsi untuk menghasilkan nilai ini disebut fungsi *hash*, sedangkan nilai yang dihasilkan disebut nilai hash, contoh sederhana *hashing* adalah:
Jika substring "hi" dan basisnya 101, nilai hashnya menjadi:

$\text{hash("hi")} = (104 \times 101^1) + (105 \times 101^0)$
$\text{hash("hi")} = 10609$
Nilai ASCII dari "h" adalah 104 dan "i" adalah 105.

2. 6. Algoritma Rabin-Karp

Konsep algoritam Rabin-Karp pada dasarnya akan membandiingkan nilai *hash* dari *string* masukan dan *substring* pada teks. Apabila sama maka akan dilakukan perbandingan sekali lagi terhadap karakter-karakternya. Sedangkan apabila tidak sama maka *substring* akan bergeser ke kanan. Kunci utama performa dari algoritma Rabin-Karp adalah perhitungan yang efisien terhadap nilai *hash substring* pada saat pergeseran dilakukan.
Berikut adalah contoh cara kerja dari algoritma Rabin-Karp:
Diberikan masukan "makan" dan teks "sayamakannasi"; dengan ordinal masing-masing karakter sesuai dengan urutan huruf dalam alphabet (a=1, b=2..z=26). Fungsi hash yang dipakai menggunakan nilai modulo adalah 3 dengan basis bilangannya adalah 10.
Tahap pertama sebelum memulai pembandingan adalah perhitungan *hash-value* dari setiap *substring* yang akan dilakukan sebagai nilai pembanding.
Perhitungan *hash-value* dapat dilakukan dengan cara:

- $\text{hash("cab")} = (3 \times 10^2 + 1 \times 10^1 + 2 \times 10^0) \bmod 3$
  $\text{hash("cab")} = 0$
- $\text{hash("aab")} = (1 \times 10^2 + 1 \times 10^1 + 2 \times 10^0) \bmod 3$
  $\text{hash("aab")} = 1$

Langkah	Pembandingan									
1	Teks	a	a	b	b	c	a	b	a	ketemu = false
	Masukan	c	a	b						hash("cab") = 0 hash("aab") = 1

Proses pembandingan dilakukan hingga akhir dari teks. Proses tersebut adalah sebagai berikut

Langkah	Pembandingan									
1	Teks	a	a	b	b	c	a	b	a	ketemu = false
	Masukan	c	a	b						hash("cab") = 0 hash("aab") = 1

2	Teks	a	a	b	b	c	a	b	a	ketemu = false
	Masukan		c	a	b					hash("cab") = 0 hash("abb") = 2
3	Teks	a	a	b	b	c	a	b	a	ketemu = false
	Masukan			c	a	b				hash("cab") = 0 hash("bbc") = 1
4	Teks	a	a	b	b	c	a	b	a	ketemu = false
	Masukan				c	a	b			hash("cab") = 0 hash("bca") = 0
	Karena hasil hashing yang sama maka dilakukan pengecekan karakter per karakter (brute force), dan hasilnya tidak sama sehingga ketemu = false									
5	Teks	a	a	b	b	c	a	b	a	ketemu = true
	Masukan					c	a	b		hash("cab") = 0 hash("cab") = 0

2. 7. Pengukuran nilai *similarity*

Inti dari pendekatan k-grams dibagi menjadi dua tahap. Tahap pertama, membagi kata menjadi k-grams. Kedua, mengelompokkan hasil terms dari k-grams yang sama. Kemudian untuk menghitung similarity dari kumpulan kata tersebut maka digunakan Dice's Similarity Coefficient untuk pasangan kata yang digunakan. Nilai similaritas tersebut dapat dihitung dengan menggunakan persamaan 2.5:

$$S = \frac{2 \times C}{(A+B)} \qquad (2.5)$$

Dimana S adalah nilai similarity, A dan B adalah jumlah dari kumpulan k-grams dalam teks 1 dan teks 2. C adalah jumlah dari k-grams yang sama dari teks yang dibandingkan. Berikut ini adalah contoh penghitungan nilai similarity 3 kata dengan $K = 2$.

Kata yang dibandingkan	K-grams yang sama	Similarity
Photography = 10 Photographic = 11	Ph ho ot to og gr ra ap ph = 9	$\frac{2 \times 9}{(10+11)} = 0.86$
Photography = 10 Phonetic = 7	Ph ho = 2	$\frac{2 \times 2}{(10+7)} = 0.24$
Photographic = 11 Phonetic = 7	Ph ho ic = 3	$\frac{2 \times 3}{(11+7)} = 0.33$

2. 8. Diagram alir sistem pendeteksi secara keseluruhan

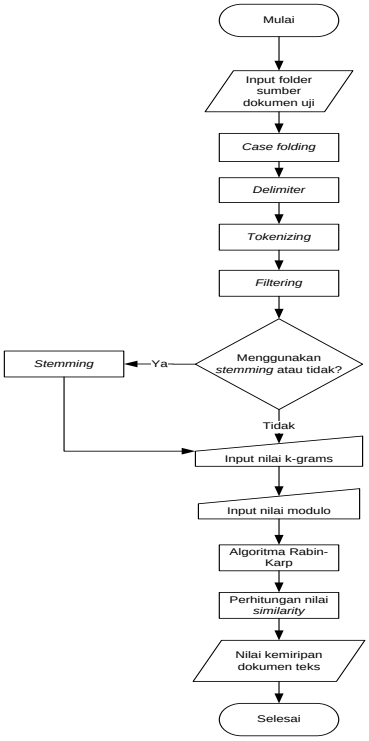
Secara umum sistem pendeteksi plagiarisme ini dimulai dengan menginputkan dokumen-dokumen uji oleh pengguna sistem dari sebuah *source folder*, dokumen-dokumen tersebut akan diproses menggunakan algoritma Rabin-Karp untuk melakukan perbandingan antara satu atau beberapa dokumen dengan dokumen lainnya yang akan menghasilkan nilai kemiripan sebagai bahan pertimbangan bagi pembuat keputusan atau pengguna sistem itu sendiri untuk memutuskan apakah dokumen tersebut merupakan hasil plagiarisme atau tidak.

Cara kerja sistem secara lebih spesifik dimulai setelah pengguna sistem menginputkan dokumen-dokumen uji kedalam sistem pendeteksi, dokumen-dokumen tersebut kemudian diekstraksi dalam tahap *preprocessing* dengan tujuan untuk mengoptimalkan proses perbandingan dokumen menggunakan algoritma Rabin-Karp. *Preprocessing* sendiri terdiri dari 6 prosedur utama antara lain case folding, *delimiter*, *tokenizing*, *filtering*, *stemming*, perhitungan nilai k-grams dan *modulo*. Proses-proses ini akan menghasilkan token-token kata, nilai k-grams serta *modulo* yang akan dipakai pada algoritma Rabin-Karp nantinya.

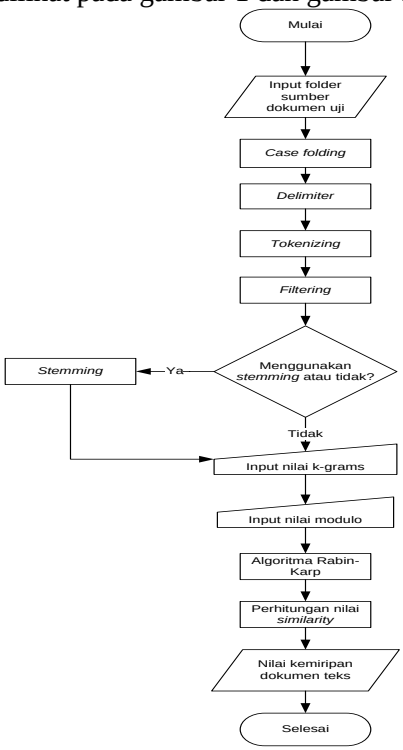
ISSN 2337-7631

Algoritma Rabin-Karp akan melakukan perbandingan terhadap dokumen-dokumen tersebut setelah melalui hasil ekstrasi dan optimasi pada tahap *preprocessing*. Nilai perbandingan menggunakan algoritma Rabin-Karp akan dihitung menggunakan proses *similarity* untuk menghasilkan presentase kemiripan antara dokumen-dokumen tersebut.

Secara keseluruhan, sistem pendeteksi terdiri atas dua bagian besar yaitu sistem pendeteksi secara otomatis dan manual. Sistem pendeteksi secara otomatis adalah sistem yang memperoleh nilai *modulo* serta k-grams melalui proses perhitungan di dalam sistem, sedangkan sistem pendeteksi secara *manual* adalah sistem yang mengharuskan penggunanya untuk memasukkan nilai k-grams serta *modulo*-nya sendiri. *Flowchart* sistem pendeteksi secara keseluruhan dapat dilihat pada gambar 1 dan gambar 2



Gambar 1 *Flowchart* sistem utama pendeteksi secara otomatis



Gambar 2 *Flowchart* sistem utama pendeteksi secara *manual*

Hal utama yang membedakan antara sistem yang secara otomatis dan *manual* adalah pada *preprocessing*; sistem secara otomatis akan melakukan fungsi *modus* untuk menentukan nilai k-grams dan juga fungsi pendekatan bilangan prima untuk menentukan *modulo*.

III. HASIL DAN PEMBAHASAN

Evaluasi performa sistem didapatkan dari perbandingan nilai similaritas atau tingkat kemiripan dokumen teks beserta durasi waktu perbandingan sesuai dengan nilai k-grams serta *modulo* yang dihasilkan otomatis oleh sistem ataupun melalui masukan dari pengguna sistem berdasarkan *preprocessing* yang dilakukan yaitu menggunakan *stemming* atau tidak.

3.1 Hasil pengujian nilai k-grams

Pengujian nilai k-grams bertujuan untuk mengkaji dan menghasilkan grams yang ideal untuk dipakai dalam proses *parsing*. Proses *parsing* sangat berpengaruh terhadap nilai *similarity* atau tingkat kemiripan dokumen teks; semakin kecil nilai grams maka proses pencocokan semakin memperhatikan kata-kata yang menyusun kalimat tersebut dan mengabaikan struktur katanya, dan sebaliknya semakin

besar nilai grams maka proses pencocokan semakin memperhatikan struktur kata dalam kalimat dan mengabaikan kata-kata yang menyusun kalimat tersebut.

Pendekatan matematis yang komperhensif harus dilakukan untuk memperoleh nilai grams yang tidak terlalu kecil karena walaupun nilai *similarity* yang dihasilkannya meningkat namun proses pencocokannya tidak efektif karena cenderung mengabaikan struktur kata yang tersusun dalam kalimat, begitu juga dengan nilai grams yang terlalu besar walaupun memperhatikan struktur kata tetapi cenderung mengabaikan kata-kata penyusun yang ada didalamnya sehingga nilai *similarity* yang dihasilkanpun semakin kecil, dan tujuan dari pengujian ini adalah untuk memperoleh nilai grams yang mampu memperhatikan kata-kata dalam kalimat tanpa mengabaikan struktur kata yang tersusun pada kalimat tersebut, karena hasil dari *preprocessing* merupakan himpunan kata-kata yang tersusun dalam kalimat maka pendekatan matematis yang diuji adalah fungsi *mean*, *median* dan *modus*.

Tabel 4. Nilai *mean*, *median* dan *modus* dari *file text* C

C1.txt					
Menggunakan *stemming*			Tanpa menggunakan *stemming*		
Mean	*Median*	*Modus*	*Mean*	*Median*	*Modus*
5,596 dibulatkan menjadi 6	5	5	6,8 dibulatkan menjadi 7	7	5
C2.txt					
Menggunakan *stemming*			Tanpa menggunakan *stemming*		
Mean	*Median*	*Modus*	*Mean*	*Median*	*Modus*
5,848 dibulatkan menjadi 6	5	5	6,587 dibulatkan menjadi 7	6	5

Tabel 5. Jumlah kata yang ditemukan menggunakan fungsi *mean*, *median* dan *modus*

C1		
Fungsi	Jumlah kata yang ditemukan tanpa *stemming*	Jumlah kata yang ditemukan menggunakan *stemming*
Mean	12	8
Median	12	31
Modus	24	31
C2		
Fungsi	Jumlah kata yang ditemukan tanpa *stemming*	Jumlah kata yang ditemukan menggunakan *stemming*
Mean	9	10
Median	12	10
Modus	28	33

Berdasarkan nilai dari perhitungan fungsi *mean*, *median* dan *modus*, nilai yang dihasilkan fungsi *modus* lebih kecil dari fungsi *mean* dan *median* dikarenakan nilai tersebut didapat dari panjang kata yang paling banyak muncul, hal inilah yang menyebabkan fungsi *modus* lebih efektif untuk mencari nilai grams yang dipakai pada proses parsing dimana nilai grams tersebut mampu menghasilkan representasi kata terbanyak tanpa mengabaikan struktur dari kata yang tersusun dalam kalimat tersebut. Perbandingan nilai *similarity* menggunakan fungsi *mean*, *median* dan *modus* pada proses *parsing* untuk pengujian file C1.txt terhadap C2.txt bisa dilihat pada tabel 6

ISSN 2337-7631

Tabel 6. Perbandingan nilai *similarity* fungsi *mean*, *median* dan *modus* untuk pengujian file C1.txt terhadap C2.txt

Fungsi	Menggunakan *stemming*		Tanpa menggunakan *stemming*	
	Grams	Nilai *similarity* (%)	Grams	Nilai *similarity* (%)
Mean	6	78.4158	7	74
Median	5	82.0158	6	78.7022
Modus	5	82.0158	5	82.8904

Nilai *similarity* yang dihasilkan oleh fungsi *modus* tanpa menggunakan *stemming* lebih besar dikarenakan proses *stemming* menghasilkan kata dasar dari kata-kata berimbuhan tanpa mengubah struktur kata yang tersusun dalam kalimat tersebut sehingga proses pencocokan tetap melihat struktur kata hasil *stemming* dengan jumlah pola kata yang berkurang akibat reduksi kata-kata berimbuhan akibatnya terjadi sedikit penurunan nilai *similarity* pada pengujian yang menggunakan *stemming* sebagai *preprocessing*.

Berdasarkan hasil pengujian tersebut, secara keseluruhan fungsi modus menghasilkan nilai similaritas rata-rata 3,544% di atas fungsi *mean* dan 2,859% di atas fungsi *median* pada pengujian tanpa menggunakan *stemming*, dan 2,456% di atas fungsi *mean* dan 0,75% di atas fungsi *median* pada pengujian menggunakan *stemming*, hal tersebut membuktikan bahwa penggunaan fungsi *modus* lebih efektif dibandingkan penggunakan fungsi *mean* dan *median* dalam menentukan nilai grams yang akan digunakan pada proses *parsing*.

3.2 Hasil pengujian nilai modulo

Tujuan dari bilangan prima terbesar adalah untuk mereduksi terjadinya *quadratic* atau nilai *hashing* yang sama pada saat perbandingan menggunakan algoritma Rabin-Karp, sehingga nilai *modulo* tersebut mampu meningkatkan kompleksitas langkah perbandingan.

Perhitungan panjang kata terbanyak dalam dokumen teks bertujuan untuk mendapatkan nilai k-grams yang bisa merepresentasikan pola-pola kata hasil dari proses *parsing* yang hasilnya mampu mewakili kata serta tidak mengabaikan struktur kata dalam kalimat tersebut pada saat proses pencocokan *string*. Sistem otomatis adalah sistem yang mampu menghasilkan nilai k-grams serta *modulo* yang ideal untuk dipakai pada saat *parsing* dan *hashing* sesuai dengan isi dari dokumen teks yang diujikan. Hasil pengujian sistem secara otomatis bisa dilihat pada tabel 7 dan tabel 8, sedangkan hasil pengujian menggunakan masukan bisa dilihat pada tabel 9 dan tabel 10.

Tabel 7. Hasil Perhitungan Nilai *Modulo* dan K-Grams Secara Otomatis Tanpa Menggunakan *Stemming*

Dok asli	**Dok uji**	**Perbandingan waktu (detik)**			**Rata-rata**	***Modulo***	**K-grams**	***Similarity* (%)**
		Percobaan ke-						
		1	**2**	**3**				
Dokumen A								
A1.txt	A1.txt	6.03	5.87	6.81	**6.24**	**134881**	**4**	**100**
A1.txt	A2.txt	6.33	5.56	5.55	**5.81**	**134881**	**4**	**85.9532**
A1.txt	A3.txt	6.65	6.18	5.4	**6.08**	**134881**	**4**	**79.3215**
A2.txt	A1.txt	5.55	5.4	5.56	**5.50**	**134881**	**4**	**86.9338**
A2.txt	A2.txt	6.02	5.55	5.56	**5.71**	**134881**	**4**	**100**
A2.txt	A3.txt	5.71	5.72	5.56	**5.66**	**134881**	**4**	**79.9677**
A3.txt	A1.txt	5.4	6.34	5.24	**5.66**	**134881**	**4**	**84.1463**
A3.txt	A2.txt	5.25	5.4	5.56	**5.40**	**134881**	**4**	**82.107**
A3.txt	A3.txt	5.56	5.09	5.4	**5.35**	**134881**	**4**	**100**
Dokumen B								
B1.txt	B1.txt	3.69	2.9	3.06	**3.12**	**13489759**	**6**	**100**

B1.txt	B2.txt	3.06	2.75	3.37	**2.91**	**13489759**	**6**	**70.8249**
B2.txt	B1.txt	4.15	3.22	2.59	**3.00**	**13489759**	**6**	**73.8197**
B2.txt	B2.txt	3.84	2.59	2.59	**3.12**	**134897759**	**7**	**100**
Dokumen C								
C1.txt	C1.txt	6.34	6.64	6.81	**6.46**	**132385913**	**7**	**100**
C1.txt	C2.txt	6.03	6.18	6.18	**6.15**	**132385917**	**7**	**74**
C2.txt	C1.txt	6.02	6.34	6.03	**6.24**	**132385917**	**7**	**73.2673**
C2.txt	C2.txt	6.03	6.96	5.87	**6.31**	**132385917**	**7**	**100**
Dokumen D								
D1.txt	D1.txt	7.07	6.92	7.07	**7.04**	**131903087**	**7**	**100**
D1.txt	D2.txt	7.07	6.91	7.22	**7.16**	**131962981**	**7**	**71.8535**
D2.txt	D1.txt	7.85	7.54	7.07	**7.29**	**131962981**	**7**	**70.4698**
D2.txt	D2.txt	7.07	8.32	6.91	**7.26**	**131962981**	**7**	**100**
Dokumen E								
E1.txt	E1.txt	4.31	3.52	3.37	**3.74**	**132403559**	**7**	**100**
E1.txt	E2.txt	4.31	3.68	5.55	**4.31**	**132403559**	**7**	**65.5856**
E2.txt	E1.txt	3.69	4.31	3.99	**4.25**	**132403559**	**7**	**70.9552**
E2.txt	E2.txt	4.31	5.25	4.31	**4.59**	**132403559**	**7**	**100**
Dokumen F								
F1.txt	F1.txt	9.09	8.01	8.01	**8.22**	**132891133**	**7**	**100**
F1.txt	F2.txt	7.69	7.54	7.54	**7.60**	**132891191**	**7**	**65.4709**
F2.txt	F1.txt	7.69	7.84	8.47	**7.91**	**132891191**	**7**	**64.1949**
F2.txt	F2.txt	7.22	7.54	7.23	**7.38**	**132891191**	**7**	**100**
Dokumen G								
G1.txt	G1.txt	9.09	8.94	9.1	**9.03**	**131919653**	**7**	**100**
G1.txt	G2.txt	8.94	9.1	9.1	**9.04**	**131919653**	**7**	**67.3611**
G2.txt	G1.txt	9.41	9.41	9.09	**9.22**	**131919653**	**7**	**68.9252**
G2.txt	G2.txt	8.79	9.1	9.09	**8.97**	**131919653**	**7**	**100**

Tabel 8. Hasil Perhitungan Nilai *Modulo* dan K-Grams Secara Otomatis Menggunakan *Stemming*

Dok asli	**Dok uji**	**Perbandingan waktu (detik)**			**Rata -rata**	***Modulo***	**K-grams**	***Similarity* (%)**
		Percobaan ke-						
		1	**2**	**3**				
Dokumen A								
A1.txt	A1.txt	26.95	23.52	23.37	**24.61**	**1348969**	**5**	**100**
A1.txt	A2.txt	31.01	30.22	23.1	**28.11**	**1348969**	**5**	**78.8732**
A1.txt	A3.txt	23.41	23.41	26.38	**24.40**	**1348969**	**5**	**69.5736**
A2.txt	A1.txt	31.63	30.23	31.17	**31.01**	**1348969**	**5**	**78.7629**
A2.txt	A2.txt	31.84	27	30.28	**29.71**	**1348961**	**5**	**100**
A2.txt	A3.txt	26.58	30.28	30.43	**29.10**	**1348969**	**5**	**71.5116**
A3.txt	A1.txt	24.82	23.1	25.13	**24.35**	**1348969**	**5**	**72.9897**
A3.txt	A2.txt	31.37	31.53	32.61	**31.84**	**1348969**	**5**	**73.0382**
A3.txt	A3.txt	24.4	32.3	25.02	**27.24**	**1348969**	**5**	**100**
Dokumen B								
B1.txt	B1.txt	20.05	26.56	20.68	**22.43**	**1348961**	**5**	**100**
B1.txt	B2.txt	25.98	27.86	26.3	**26.71**	**1348961**	**5**	**71.1443**
B2.txt	B1.txt	24.73	25.36	26.6	**25.56**	**1348961**	**5**	**75.4569**
B2.txt	B2.txt	23.54	24.47	24.88	**24.30**	**1348961**	**5**	**100**
Dokumen C								
C1.txt	C1.txt	22.3	21.52	22.3	**22.04**	**1323841**	**5**	**100**
C1.txt	C2.txt	20.11	19.33	20.11	**19.85**	**1323841**	**5**	**82.0158**
C2.txt	C1.txt	18.87	20.58	19.02	**19.49**	**1323841**	**5**	**83.4971**
C2.txt	C2.txt	16.84	16.37	16.52	**16.58**	**1323841**	**5**	**100**

ISSN 2337-7631

Dokumen D								
D1.txt	D1.txt	17.41	16.95	16.64	**17.00**	**131843**	**4**	**100**
D1.txt	D2.txt	18.34	18.97	17.72	**18.34**	**131851**	**4**	**83.8196**
D2.txt	D1.txt	17.11	17.72	18.35	**17.73**	**131851**	**4**	**83.2905**
D2.txt	D2.txt	20.22	18.5	18.5	**19.07**	**131851**	**4**	**100**
Dokumen E								
E1.txt	E1.txt	25.09	24.16	25.71	**24.99**	**132389**	**4**	**100**
E1.txt	E2.txt	22.64	20.46	20.78	**21.29**	**132389**	**4**	**81.1947**
E2.txt	E1.txt	28.99	21.4	20.93	**23.77**	**132389**	**4**	**87.3536**
E2.txt	E2.txt	24.21	26.39	23.89	**24.83**	**132389**	**4**	**100**
Dokumen F								
F1.txt	F1.txt	23.01	22.55	22.86	**22.81**	**1328899**	**5**	**100**
F1.txt	F2.txt	20.78	21.41	19.85	**20.68**	**1328899**	**5**	**72.0745**
F2.txt	F1.txt	20.16	22.81	20.78	**21.25**	**1328899**	**5**	**71.2468**
F2.txt	F2.txt	16.69	16.99	17.93	**17.20**	**1328899**	**5**	**100**
Dokumen G								
G1.txt	G1.txt	18.39	20.27	19.8	**19.49**	**1319183**	**5**	**100**
G1.txt	G2.txt	18.6	18.6	18.45	**18.55**	**1319183**	**5**	**78.6704**
G2.txt	G1.txt	20.16	18.92	17.35	**18.81**	**1319183**	**5**	**77.1978**
G2.txt	G2.txt	25	25.46	25.15	**25.20**	**1319183**	**5**	**100**

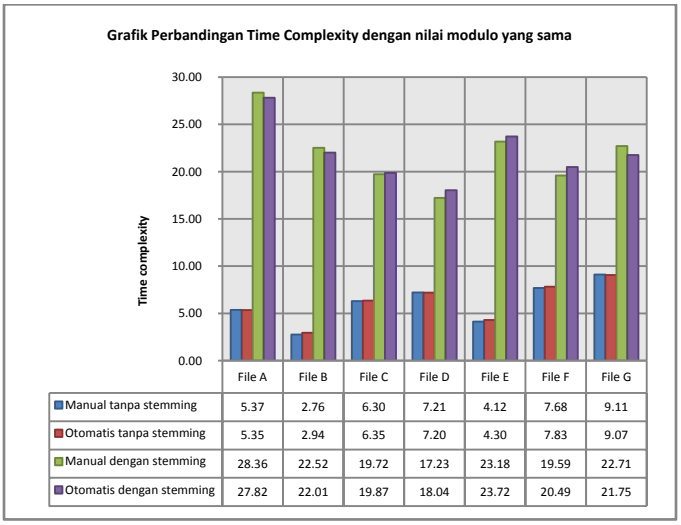
Tabel 9. Hasil Pengujian Menggunakan Masukan Nilai *Modulo* dan K-Grams yang Sama dengan Hasil Perhitungan Secara Otomatis Tanpa Menggunakan *Stemming*

Dok asli	**Dok uji**	**Perbandingan waktu (detik)**			**Rata-rata**	***Modulo***	**K-grams**	***Similarity* (%)**
		Percobaan ke-						
		1	**2**	**3**				
Dokumen A								
A1.txt	A1.txt	5.25	5.25	5.25	**5.25**	**1348969**	**5**	**100**
A1.txt	A2.txt	5.4	5.56	5.24	**5.40**	**1348969**	**5**	**78.8732**
A1.txt	A3.txt	5.09	5.09	5.24	**5.14**	**1348969**	**5**	**69.5736**
A2.txt	A1.txt	5.4	5.56	5.25	**5.40**	**1348969**	**5**	**78.7629**
A2.txt	A2.txt	6.81	5.25	5.4	**5.82**	**1348961**	**5**	**100**
A2.txt	A3.txt	5.09	5.56	5.24	**5.30**	**1348969**	**5**	**71.5116**
A3.txt	A1.txt	5.09	5.4	5.24	**5.24**	**1348969**	**5**	**72.9897**
A3.txt	A2.txt	5.24	6.02	5.4	**5.55**	**1348969**	**5**	**73.0382**
A3.txt	A3.txt	5.09	4.94	5.56	**5.20**	**1348969**	**5**	**100**
Dokumen B								
B1.txt	B1.txt	2.9	2.59	2.91	**2.80**	**1348961**	**5**	**100**
B1.txt	B2.txt	2.75	2.6	2.74	**2.70**	**1348961**	**5**	**76.1044**
B2.txt	B1.txt	2.9	2.59	2.75	**2.75**	**1348961**	**5**	**78.3726**
B2.txt	B2.txt	2.74	2.9	2.74	**2.79**	**1348961**	**5**	**100**
Dokumen C								
C1.txt	C1.txt	6.18	6.33	6.34	**6.28**	**1323841**	**5**	**100**
C1.txt	C2.txt	6.34	6.49	6.8	**6.54**	**1323841**	**5**	**82.8904**
C2.txt	C1.txt	6.34	6.5	6.02	**6.29**	**1323841**	**5**	**81.7434**
C2.txt	C2.txt	6.02	6.18	6.03	**6.08**	**1323841**	**5**	**100**
Dokumen D								
D1.txt	D1.txt	6.91	7.22	7.38	**7.17**	**131903087**	**7**	**100**
D1.txt	D2.txt	7.06	7.07	7.22	**7.12**	**131962981**	**7**	**71.8535**
D2.txt	D1.txt	7.06	7.23	7.07	**7.12**	**1319629**	**7**	**70.4698**

						81		
D2.txt	D2.txt	7.69	7.38	7.22	**7.43**	**1319629 81**	**7**	**100**
Dokumen E								
E1.txt	E1.txt	3.52	3.53	3.68	**3.58**	**1324035 59**	**7**	**100**
E1.txt	E2.txt	4.62	4	3.99	**4.20**	**1324035 59**	**7**	**65.5856**
E2.txt	E1.txt	3.68	4.31	4.3	**4.10**	**1324035 59**	**7**	**70.9552**
E2.txt	E2.txt	4.78	4.46	4.62	**4.62**	**1324035 59**	**7**	**100**
Dokumen F								
F1.txt	F1.txt	8.01	8.01	8.01	**8.01**	**1328911 33**	**7**	**100**
F1.txt	F2.txt	7.69	7.69	7.85	**7.74**	**1328899**	**5**	**74.7768**
F2.txt	F1.txt	7.69	7.54	7.54	**7.59**	**1328899**	**5**	**73.8397**
F2.txt	F2.txt	7.38	7.38	7.38	**7.38**	**1328899**	**5**	**100**
Dokumen G								
G1.txt	G1.txt	8.94	9.09	9.25	**9.09**	**1319183**	**5**	**100**
G1.txt	G2.txt	8.94	8.94	9.09	**8.99**	**1319183**	**5**	**78.1106**
G2.txt	G1.txt	9.87	9.1	9.26	**9.41**	**1319183**	**5**	**78.3721**
G2.txt	G2.txt	8.94	8.78	9.1	**8.94**	**1319183**	**5**	**100**

3.3 Pembahasan

Tujuan dari pengujian ini adalah untuk melihat performa sistem dalam pencocokan *string* baik menggunakan perhitungan otomatis maupun memakai masukan dari pengguna sistem. Pengujian yang pertama ini akan difokuskan pada penggunaan bilangan prima sebagai nilai *modulo* pada proses *hashing* menggunakan algoritma Rabin-Karp; semakin besar bilangan prima yang dipakai maka semakin kecil pula kesempatan terjadinya *quadratic* atau kesamaan nilai *hash* pada pola kata yang berbeda[5], waktu atau durasi pencocokan *string* mewakili *time complexity* dari algoritma Rabin Karp dalam melakukan pencocokan *string* sehingga dari pengujian ini bisa disimpulkan performa sistem dalam mencocokan *string* baik secara otomatis melakukan perhitungan bilangan prima atau tidak.



Gambar 3 perbandingan *time complexity* dengan nilai *modulo* yang sama

ISSN 2337-7631

Pada gambar 3 terlihat bahwa *time complexity* yang dihasilkan antara pengujian sistem manual dengan pengujian otomatis terdapat perbedaan waktu yang tidak terlalu signifikan yaitu sistem manual memiliki estimasi waktu rata-rata 0,07 detik lebih kecil dari sistem otomatis tanpa menggunakan *stemming* dan 0,06 detik lebih kecil dari sistem otomatis dengan menggunakan *stemming*, hal ini menunjukkan bahwa walaupun waktu yang diperlukan oleh sistem yang menggunakan masukan lebih (sistem manual) cepat dibandingkan dengan sistem otomatis, namun karena sistem otomatis mampu menghasilkan bilangan prima yang besar untuk mereduksi terjadinya *quadratic* maka sistem otomatis tetap efektif untuk dipakai; kecilnya perbandingan *time complexity* tersebut sepadan dengan hasil yang diharapkan yaitu sistem mampu menghasilkan nilai *modulo* yang ideal sehingga tidak terjadi *quadratic* atau proses pencocokan *string* mencapai titik pencocokan maksimum.

IV. KESIMPULAN DAN SARAN

4.1 Kesimpulan

Berdasarkan hasil pengujian dan pengamatan terhadap sistem pendeteksi dugaan plagiarisme menggunakan algoritma Rabin-Karp yang dibuat dapat disimpulkan sebagai berikut:

1. Makin besar nilai grams maka hasil dari proses parsing akan semakin memperhatikan struktur kata penyusun kalimat dibandingkan kata-kata yang tersusun di dalamnya, sebaliknya semakin kecil nilai grams maka hasil proses parsing akan semakin memperhatikan kata-kata dalam kalimat dibandingkan struktur kata dari kalimat tersebut.
2. Nilai grams yang ideal adalah nilai grams yang mampu menghasilkan representasi kata-kata terbanyak dalam kalimat dibandingkan dengan nilai grams lainnya.
3. Fungsi modus sebagai pendekatan matematis untuk mendapatkan nilai grams terbukti lebih baik dibandingkan fungsi mean dan median karena memiliki rata-rata nilai similarity 3,544% di atas fungsi mean dan 2,859% di atas fungsi median pada pengujian tanpa menggunakan stemming, dan 2,456% di atas fungsi mean serta 0,75% di atas fungsi median untuk pengujian yang menggunakan stemming.
4. Nilai similarity yang dihasilkan dengan nilai k-grams yang sama tanpa menggunakan stemming sebagai *preprocessing* lebih baik dibandingkan menggunakan stemming karena hasil stemming yang tak merubah struktur kata dan mengurangi jumlah pola yang dihasilkan proses parsing.
5. Pengujian terhadap 7 data set dengan 2 varian dan 3 kali perlakukan membuktikan pendekatan bilangan prima terbesar efektif untuk menghasilkan bilangan prima atau nilai modulo ideal yang bisa dipakai pada proses hashing dan mampu menghilangkan terjadinya quadratic pada saat pencocokan string menggunakan algoritma Rabin-Karp.
6. Durasi pencocokan string yang dihasilkan menggunakan perhitungan bilangan prima terbesar menunjukkan durasi waktu pencocokan yang lebih baik dengan sistem manual yaitu rata-rata durasi pengujian 2,04 detik lebih cepat dibandingkan sistem manual yang memiliki nilai modulo yang kecil pada pengujian yang menggunakan stemming dan 0,13 detik pada pengujian tanpa menggunakan stemming.
7. Rata-rata durasi pencocokan yang dihasilkan menggunakan nilai modulo yang sama terhadap 7 buah data set menunjukkan sistem manual memiliki durasi yang lebih baik yaitu 0,07 detik dibandingkan sistem otomatis tanpa menggunakan stemming dan 0,06 detik pada pengujian yang menggunakan stemming. Walaupun dengan nilai modulo yang sama durasi pengujian sistem secara manual memiliki waktu yang lebih baik, namun sistem yang menggunakan perhitungan bilangan prima tetap efektif karena mampu menghasilkan nilai modulo yang ideal.
8. Telah dibuat suatu sistem otomatis yang mampu menghasilkan nilai modulo dan k-grams yang ideal untuk algoritma Rabin-Karp dan dapat digunakan untuk mendeteksi dugaan plagiarisme terhadap dokumen teks.

4.2 Saran

Saran Peneliti bagi pengembangan penelitian selanjutnya :

1. Untuk penelitian yang lebih lanjut menggunakan algoritma Rabin-Karp dalam mengembangkan sistem pendeteksi dugaan plagiat maka saran yang dapat diberikan sebagai berikut:
2. *Preprocessing* yang digunakan sebelum proses pencocokan string harus bisa memperhatikan semantic dari sebuah kata ataupun kalimat dengan memperhatikan aturan morfologi bahasa Indonesia.
3. Pengujian sebaiknya dilakukan dengan data latih yang bervariasi seperti pengubahan kata maupun bentuk kalimat dengan definisi yang sama untuk menghasilkan *preprocessing* yang lebih akurat.
4. Sistem juga mampu menguji tak hanya dokumen plainteks saja tetapi juga dokumen lain yang berisi gambar, grafik atau objek lainnya.
5. Sistem juga mampu berbasis online untuk mendapatkan database *preprocessing* yang memadai serta pengujian dokumen secara online..

DAFTAR PUSTAKA

.

[1] Tim Penyusun Kamus Pusat Bahasa. 2008. *Kamus Bahasa Indonesia*. Pusat Bahasa Departemen Pendidikan Nasional. Jakarta.

[2] Choy, William. 2008. *Talk on Plagiarism and Referencing*. Policy and Leadership Academic Group.

[3] Iyer, Parvati dan Singh, Abhipsita. 2005. *Document similarity analysis for a plagiatrism detection system*. In Proceedings of the 2nd Indian.

[4] Clough, Paul. 2000. *Plagiarism in Natural and Programming Languages: an Overview of Current Tools and Technologies*. Department of Computer Science, University of Sheffield.

[5] Chillar, Rajender Singh; Kochar, Barjesh. 2008. *RB-Matcher: String Matching Technique*. World Academy of Science, Engineering and Technology.