# Perancangan Aplikasi Untuk Menghitung Persentase Kemiripan Proposal Dan Isi Skripsi Dengan Algoritma Rabin-Karp

**[1]Ledisma Juliana Purba, [2]Lamhot Sitorus**
[1]Teknik Informatika Unika St. Thomas S.U; Jln. Setia Budi No.479-F Medan, 061-8210161
[2]Teknik Informatika Unika St. Thomas S.U; Jln. Setia Budi No.479-F Medan, 061-8210161
e-mail : [1]ledisjuliana@gmail.com; [2]lamhot68@yahoo.com

**Abstrak**

Kemiripan atau *similarity* merupakan tingkat perbandingan persentase kemiripan antar dokumen yang diuji. *Similarity* ini akan dijadikan acuan dalam menentukan persentase tingkat kemiripan sebuah dokumen yang diuji. Besarnya persentase *similarity* akan dipengaruhi oleh tingkat kemiripan dari dokumen yang diuji, semakin besar persentase *similarity* maka tingkat kemiripan akan semakin dianggap tinggi, oleh karena itu, perlu dirancang aplikasi menghitung persentase kemiripan proposal dan isi skripsi dengan algoritma *Rabin-Karp*.

Algoritma *Rabin-Karp* adalah algoritma pencarian kata yang mencari sebuah pola berupa *substring* dalam sebuah teks dengan menggunakan hashing. Teknik *hashing* merupakan transformasi aritmatika sebuah *string* dari karakter menjadi nilai numerik yang merepresentasikan *string* aslinya. Algoritma ini diimplemetasikan ke dalam sistem berbasis web pada tugas akhir Teknik Informatika . Sistem dikembangkan mulai dari pengumpulan kebutuhan sistem dari pengguna, perancangan dan kemudian implementasi sistem.

Berdasarkan pengujian yang dilakukan, didapat bahwa implementasi algoritma *Rabin-Karp* dan rumus *jaccard Coefficient* telah berhasil menghitung tingkat persentase kemiripan antar proposal dan isi skripsi. Waktu yang diperlukan untuk melakukan proses menghitung kemiripan semakin lama seiring jumlah data yang digunakan.

Kata kunci : *Jaccard Coefficient, Similarity, Rabin-Karp*

***Abstract***

*Similarity is the level of comparison of the percentage of similarity between the documents tested. This similarity will be used as a reference in determining the percentage level of similarity of a document being tested. The amount of similarity percentage will be influenced by the level of similarity of the document being tested, the greater the similarity percentage, the level of similarity will be increasingly considered high, therefore, it is necessary to design an application to calculate the percentage of similarity of proposals and the contents of the thesis with the Rabin-Karp algorithm.*

*Rabin-Karp algorithm is a search algorithm that searches for a substring pattern in a text using hashing. The hashing technique is an arithmetic transformation of a string from a character into a numerical value that represents the original string. This algorithm is implemented into a web-based system in the final project of Informatics Engineering. The system developed from the collection of system requirements from users, design and then system implementation.*

*Based on the test, it is found that the implementation of Rabin-Karp algorithm and jaccard coefficient formula have succeeded in calculating percentage level of similarity between proposal and thesis content. The time needed to do the process of calculating similarity is longer as the amount of data is used.*

*Keywords : Jaccard Coefficient, Similarity, Rabin-Karp*

## 1. PENDAHULUAN

Perkembangan teknologi mempermudah untuk saling bertukar dan mendapatkan informasi sehingga 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. Kemudahan dalam mengakses konten dan informasi, khususnya berupa tulisan, dapat menjadi salah satu faktor bagi manusia untuk melakukan tindakan penggandaan dokumen secara langsung, tanpa memasukkan narasumber. Salah satu contoh yang umum adalah dalam membuat skripsi, hampir seluruh isinya merupakan hasil duplikat dari tugas akhir milik orang lain. Tindakan tersebut dapat mengurangi kreatifitas seseorang dalam membuat hasil karya sendiri dan merupakan pelanggaran hak cipta.

Metode yang dapat digunakan untuk mendeteksi kemiripan dokumen atau menganalis kemiripan teks skripsi seperti pencocokan *substring*, kesamaan kata kunci pada dokumen. Algoritma Rabin-Karp lebih berguna pada pencarian *multiple pattern* dari pada pencarian *single pattern*. Karena algoritma ini tidak memperdulikan huruf besar atau kecil, dan tanda baca yang digunakan. Algoritma Rabin-Karp ini menggunakan fungsi *hash*. Fungsi *hash* adalah fungsi yang digunakan untuk mengubah *string* menjadi untaian integer. Pada algoritma ini untaian *string* akan diubah menjadi integer berdasarkan bilangan ASCII nya. Karena menggunakan bilangan ASCII, proses komputasi menjadi lebih "dekat" ke bahasa mesin. Pendekatan utamanya adalah, *string* yang sama akan memiliki nilai *hash* yang sama. Kelebihan dari algoritma Rabin-Karp dibandingkan algoritma pencocokan *string* lainnya adalah kemampuan dalam mencari banyak pola *string Input* dari algoritma *Rabin-karp* adalah *string* dari sebuah dokumen dan *output* dari algoritma tersebut adalah nilai-nilai *hash* dari dokumen tersebut.

Oleh karena itu perlu dirancang sebuah aplikasi yang dapat digunakan untuk menghitung persentase kemiripan proposal dan isi skripsi. Dengan mengetahui persentase kemiripan dokumen-dokumen tersebut dapat dijadikan bahan pertimbangan apakah dokumen yang diuji tersebut merupakan hasil menjiplak karya seseorang atau tidak, (Obed, 2013).

## 2. METODOLOGI PENELITIAN

### *II.1. Similarty Document*

Konsep *similarity* sudah menjadi isu yang sangat penting di hampir setiap bidang ilmu pengetahuan. Menurut Zaka, (2009) dalam disertasinya menjelaskan tiga macam teknik yang dibangun untuk menentukan nilai *similarity* (kemiripan) dokumen.

*Distance-based similarity measure* mengukur tingkat kesamaan dua buah objek dari segi jarak geometris dari variabel-variabel yang tercakup di dalam kedua objek tersebut. Metode *Distance-based similarity* ini meliputi Minkowski Distance, Manhattan/City block distance, Euclidean distance, Jaccard distance, Dice's Coefficient, Cosine similarity, Levenshtein Distance, Hamming distance, dan Soundex distance.

*Feature-based similarity measure* melakukan penghitungan tingkat kemiripan dengan merepresentasikan objek ke dalam bentuk feature-feature yang ingin di perbandingkan. *Feature-based similarity measure* banyak digunakan dalam melakukan pengklasifikasian atau *pattern maching* untuk gambar dan teks.

*Probabilistic-based similarity measure* menghitung tingkat kemiripan dua objek dengan merepresentasikan dua set objek yang dibandingkan dalam bentuk *probability*. *Kullback Leibler Distance dan Posterior Probability* termasuk dalam metode ini.

Menurut Mutiara dalam Surahman, (2013) untuk menentukan jenis kemiripan antara dokumen yang diuji ada 5 jenis penilaian persentase :

1. 0% : Hasil uji 0% berarti kedua dokumen tersebut benar-benar berbeda baik dari segi isi dan kalimat secara keseluruhan atau tidak ada kemiripan.

2. < 15%: Hasil uji 15% berarti kedua dokumen tersebut hanya mempunyai sedikit kemiripan.
3. 15% - 50%: Hasil uji 15% - 50% berarti menandakan dokumen tersebut termasuk mendekati kemiripan.
4. >50% : Hasil uji lebih dari 50% berarti dapat dikatakan bahwa dokumen tersebut Mirip.
5. 100% Hasil uji 100% menandakan bahwa dokumen tersebut adalah sama karena dari awal sampai akhir mempunyai isi yg sama persis.

## *II.2. Jaccard Coefficient*

Metode yang digunakan untuk menghitung adalah metode Jaccard *Coefficient. Jaccard Coeficient* adalah salah satu metode yang dipakai untuk menghitung similarity antara dua objek (*items*). Seperti halnya *cosine distance dan matching coefficient*, secara umum perhitungan metode ini didasarkan pada *vector space similarity measure*. Masing-masing dokumen akan dihitung kata yang sama antara dokumen yang satu dengan dokumen yang lain. Hasil dari perhitungan akan dihasilkan nilai similaritas dokumen. Nilai similaritas dokumen yang tertinggi dapat dianggap bahwa dokumen tersebut paling similar, dengan kata lain memiliki banyak kesamaan. *Jaccard similarity* atau *Jaccard Coefficient* menghitung similarity antara dua objek, A dan B yang dinyatakan dalam dua buah vektor.

$$\boldsymbol{Jaccard's\ Similarity}\ (\boldsymbol{A},\boldsymbol{B}) = \frac{|(\mathbf{A}\cap\mathbf{B})|}{|(\boldsymbol{A}\cup\boldsymbol{B})|}\mathbf{x\ 100}$$

Ket:
A = Hash dokumen 1 ( $A_1$, $A_2$, $A_3$..... $A_n$)
B = Hash dokumen 2 ( $B_1$, $B_2$, $B_3$..... $A_n$)

Dimana (A,B) adalah nilai similarity, |A∩B| adalah jumlah dari hash yang sama dari dokumen 1 dan 2 dan |A∪B| adalah jumlah hash dari dokumen 1dan dokumen 2. Sebagai contoh A={1,2,4}, b={1,2,4,7,8} maka, |A∩B|={1,2,4} dan |A∪B|={1,2,4,7,8} sehingga:

(A,B)= $\frac{3}{5}x100 = 60$

Dalam kasus menghitung persentase kemiripan proposal dan skripsi rumusnya disesuaikan menjadi seperti pada persamaan :

$$\textbf{\textit{Jaccard's Similarity}}\ \mathbf{(A,B)} = \frac{\textbf{jumlah hash yang sama dari kedua dokumen}}{\textbf{Hash yang unik dari kedua dokumen}}\mathbf{x\ 100}$$

## *II.3. Algoritma Rabin-Karp*

Algoritma Rabin-Karp adalah algoritma pencocokan *string* yang menggunakan fungsi *hash* sebagai pembanding antara *string* yang dicari (m) dengan *substring* pada teks (n). Apabila *hash value* keduanya sama maka akan dilakukan perbandingan sekali lagi terhadap karakter-karakternya. Apabila hasil keduanya tidak sama, maka *substring* akan bergeser ke kanan. Pergeseran dilakukan sebanyak (n-m) kali. Perhitungan nilai *hash* yang efisien pada saat pergeseran akan mempengaruhi performa dari algoritma ini. (Firdaus, 2008 dalam Noprisson, Susilo, dan Ernawati, 2014).

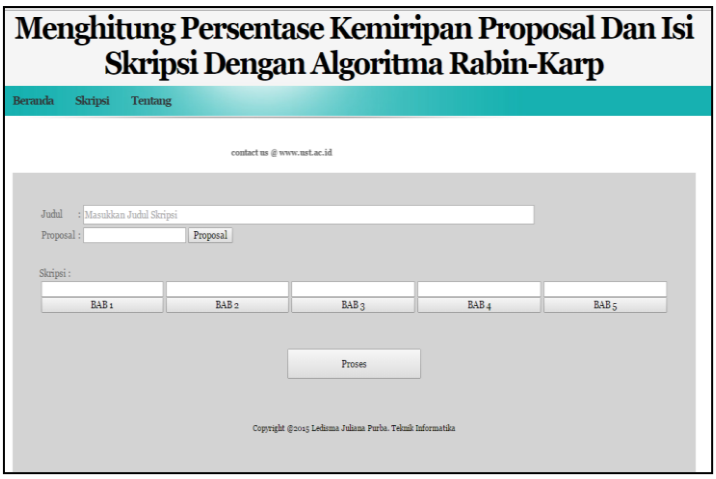
Pada dasarnya, algoritma Rabin-Karp akan membandingkan nilai *hash* dari *string* masukan dan *substring* pada teks. Apabila sama, maka akan dilakukan perbandingan sekali lagi terhadap karakter-karakternya. Apabila tidak sama, maka substring akan bergeser ke kanan. Kunci utama performa algoritma ini adalah perhitungan yang efisien terhadap nilai hash

substring pada saat penggeseran dilakukan. Berikut dijelaskan contoh kerja algoritma Rabin-Karp. (Firdaus, 2008).

Firdaus (2008) mengemukakan cara kerja algoritma Rabin-Karp : Diberikan masukan "cab" dan teks "aabbcaba". Fungsi *hash* yang dipakai misalnya akan menambahkan nilai keterurutan setiap huruf dalam alfabet (a = 1, b = 2, dst.) dan melakukan modulo dengan 3. Didapatkan nilai *hash* "cab" adalah 0 dan tiga karakter pertama pada teks yaitu "aab" adalah 1.

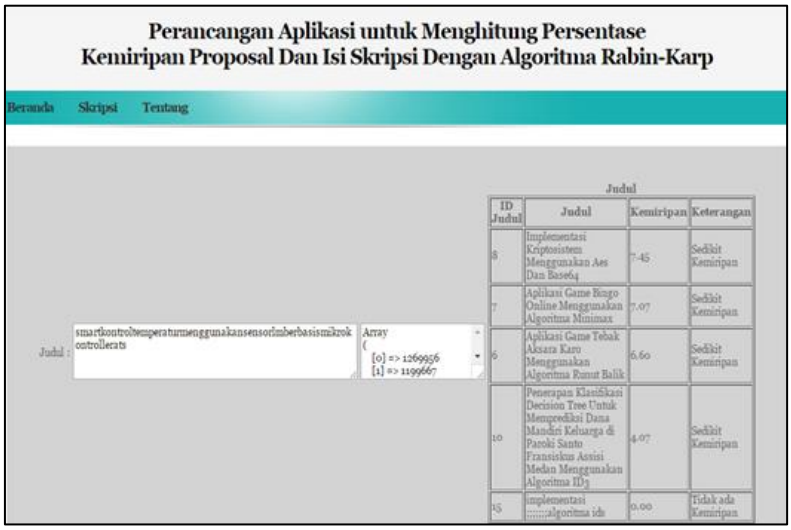
## 3. HASIL DAN PEMBAHASAN

Form Rabin-Karp ini berfungsi untuk melakukan proses pendeteksian kemiripan teks dengan menggunakan algoritma Rabin-Karp, sehingga akan menghasilkan tigkat kemiripan antar dua file teks. Tampilan form Rabin-Karp dapat dilihat pada Gambar 1.




**Gambar 1.** Form Rabin-Karp

Perhitungan tingkat persentase tertinggi dari judul yang diuji dengan yang ada didatabase, Antarmuka halaman beranda dapat dilihat pada Gambar 2.




**Gambar 2.** Persentasi Tingkat Kemiripan

Berdasarkan teori langkah-langkah algoritma Rabin-Karp diatas, maka disediakan contoh data yang akan diuji yaitu:

Kalimat 1 : Aplikasi game binggo online menggunakan algoritma minimax.

Kalimat2: Smart kontrol temperatur menggunakan sensor LM35 berbasis mikrokontroller AT89S52.

Langkah -langkah yang akan dilakukan yaitu:

1. Menghilangkan tanda baca, menghapus spasi dan mengubah teks menjadi kata-kata tanpa huruf kapital.
   Kalimat 1: aplikasigamebinggoonlinemenggunakanalgoritmaminimax
   Kalimat 2 : smartkontroltemperaturmenggunakansensorlmberbasismikrokontrollerats
2. Menentukan k-gram, basis bilangan dan membagi teks ke dalam gram-gram yang sudah ditentukan nilai k-gram nya.
   K-gram = 5 (lima)
   Basis bilangan = Desimal (10)
   **Bentuk gram kalimat 1 :**
   aplik|plika|likas|ikasi|kasig|asiga|sigam|igame|gameb|amebi|mebin|ebing|bingg|inggo|nggoo|ggoon|goonl|oonli|onlin|nline|linem|ineme|nemen|emeng|mengg|enggu|nggun|ggunagunak|unaka|nakan|akana|kanal|analg|nalgo|algor|lgori|gorit|oritm|ritma|itmam|tmami|mamin|amini|minim|inima|nimax|
   **Bentuk gram kalimat 2 :**
   smart|martk|artko|rtkon|tkont|kontr|ontro|ntrol|trolt|rolte|oltem|ltemp|tempe|emper|mpera|perat|eratu|ratur|aturm|turme|urmen|rmeng|mengg|enggu|nggun|gguna|gunak|unaka|nakan|akans|kanse|ansen|nsens|senso|ensor|nsorl|sorlm|orlmb|rlmbe|lmber|mberb|berba|erbas|rbasi|basis|asism|sismi|ismik|smikr|mikro|ikrok|kroko|rokon|okont|kontr|ontro|ntrol|troll|rolle|oller|llera|lerat|erats.
3. Mencari nilai hash dengan fungsi rolling hash dari tiap gram yang terbentuk.
   **Nilai hash kalimat 1:**

$$aplik = Ascii(a)\,97 * 10^{(5-1)} + Ascii\,(p)112 * 10^{(4-1)} + Ascii\,(l)108 * 10^{(3-1)} + Ascii\,(i)105 * 10^{(2-1)} + Ascii\,(k)107 * 10^{(1-1)}$$

$$=97 * 10^4 + 112 * 10^3 + 108 * 10^2 + 105 * 10^1 + 107 * 10^0$$

= 970000 + 112000 + 10800 + 1050 + 107

=1093957

$$plika = 112 * 10^{(5-1)} + 108 * 10^{(4-1)} + 105 * 10^{(3-1)} + 107 * 10^{(2-1)} + 97 * 10^{(1-1)}$$

$$=112 * 10^4 + 108 * 10^3 + 105 * 10^2 + 107 * 10^1 + 97 * 10^0$$

=1120000 + 108000 + 10500 + 1070 + 97

= 1239667

$$likas = 108 * 10^{(5-1)} + 105 * 10^{(4-1)} + 107 * 10^{(3-1)} + 97 * 10^{(2-1)} + 115 * 10^{(1-1)}$$

$$=108 * 10^4 + 105 * 10^3 + 107 * 10^2 + 97 * 10^1 + 115 * 10^0$$

=1080000 + 105000 + 10700 + 970 + 115

= 1196785

$$ikasi = 105 * 10^{(5-1)} + 107 * 10^{(4-1)} + 97 * 10^{(3-1)} + 115 * 10^{(2-1)} + 105 * 10^{(1-1)}$$

$$=105 * 10^4 + 107 * 10^3 + 97 * 10^2 + 115 * 10^1 + 105 * 10^0$$

=1050000 + 107000 + 9700 + 1150 +105

=1167955

$$kasig = 107 * 10^{(5-1)} + 97 * 10^{(4-1)} + 115 * 10^{(3-1)} + 105 * 10^{(2-1)} + 103 * 10^{(1-1)}$$

$$=107 * 10^4 + 97 * 10^3 + 115 * 10^2 + 105 * 10^1 + 103 * 10^0$$

=1070000 + 97000 + 11500 + 1050 +103

=1179653

$$oonli = 111 * 10^{(5-1)} + 111 * 10^{(4-1)} + 110 * 10^{(3-1)} + 108 * 10^{(2-1)} + 105 * 10^{(1-1)}$$

$$=111 * 10^4 + 111 * 10^3 + 110 * 10^2 + 108 * 10^1 + 105 * 10^0$$

=1110000 + 111000 + 11000 +1080 +105

=1233185

$onlin = 111 * 10^{(5-1)} + 110 * 10^{(4-1)} + 108 * 10^{(3-1)} + 105 * 10^{(2-1)} + 110 * 10^{(1-1)}$
$=111 * 10^4 + 110 * 10^3 + 108 * 10^2 + 105 * 10^1 + 110 * 10^0$
=1110000 + 110000 + 10800 +1050 +110
=1231960

$nline = 110 * 10^{(5-1)} + 108 * 10^{(4-1)} + 105 * 10^{(3-1)} + 110 * 10^{(2-1)} + 101 * 10^{(1-1)}$
$=110 * 10^4 + 108 * 10^3 + 105 * 10^2 + 110 * 10^1 + 101 * 10^0$
=1100000 + 108000 + 10500 +1100 +101
=1219701

$linem = 108 * 10^{(5-1)} + 105 * 10^{(4-1)} + 110 * 10^{(3-1)} + 101 * 10^{(2-1)} + 109 * 10^{(1-1)}$
$=108 * 10^4 + 105 * 10^3 + 110 * 10^2 + 101 * 10^1 + 109 * 10^0$
=1080000 + 105000 + 11000 +1010 +109
=1197119

$ineme = 105 * 10^{(5-1)} + 110 * 10^{(4-1)} + 101 * 10^{(3-1)} + 109 * 10^{(2-1)} + 101 * 10^{(1-1)}$
$=105 * 10^4 + 110 * 10^3 + 101 * 10^2 + 109 * 10^1 + 101 * 10^0$
=1050000 + 110000 + 10100 +1090 +101
=1171291

$nemen = 110 * 10^{(5-1)} + 101 * 10^{(4-1)} + 109 * 10^{(3-1)} + 101 * 10^{(2-1)} + 110 * 10^{(1-1)}$
$=110 * 10^4 + 101 * 10^3 + 109 * 10^2 + 101 * 10^1 + 110 * 10^0$
=1100000 + 101000 + 10900 +1010 +110
=1213020

$emeng = 101 * 10^{(5-1)} + 109 * 10^{(4-1)} + 101 * 10^{(3-1)} + 110 * 10^{(2-1)} + 103 * 10^{(1-1)}$
$=101 * 10^4 + 109 * 10^3 + 101 * 10^2 + 110 * 10^1 + 103 * 10^0$
=1010000 + 109000 + 10100 +1100 +103
=1130303

$mengg = 109 * 10^{(5-1)} + 101 * 10^{(4-1)} + 110 * 10^{(3-1)} + 103 * 10^{(2-1)} + 103 * 10^{(1-1)}$
$=109 * 10^4 + 101 * 10^3 + 110 * 10^2 + 103 * 10^1 + 103 * 10^0$
=1090000 + 101000 + 11000 +1030 +103
=1203133

$enggu = 101 * 10^{(5-1)} + 110 * 10^{(4-1)} + 103 * 10^{(3-1)} + 103 * 10^{(2-1)} + 117 * 10^{(1-1)}$
$=101 * 10^4 + 110 * 10^3 + 103 * 10^2 + 103 * 10^1 + 117 * 10^0$
=1010000 + 110000 + 10300 +1030 +117
=1131447

$nggun = 110 * 10^{(5-1)} + 103 * 10^{(4-1)} + 103 * 10^{(3-1)} + 117 * 10^{(2-1)} + 110 * 10^{(1-1)}$
$=110 * 10^4 + 103 * 10^3 + 103 * 10^2 + 117 * 10^1 + 110 * 10^0$
=1100000 + 103000 + 10300 +1170 +110
=1214580

$gguna = 103 * 10^{(5-1)} + 103 * 10^{(4-1)} + 117 * 10^{(3-1)} + 110 * 10^{(2-1)} + 97 * 10^{(1-1)}$
$=103 * 10^4 + 103 * 10^3 + 117 * 10^2 + 110 * 10^1 + 97 * 10^0$
=1030000 + 103000 + 11700 +1100 +97
=1145897

$gunak = 103 * 10^{(5-1)} + 117 * 10^{(4-1)} + 110 * 10^{(3-1)} + 97 * 10^{(2-1)} + 107 * 10^{(1-1)}$
$=103 * 10^4 + 117 * 10^3 + 110 * 10^2 + 97 * 10^1 + 107 * 10^0$
=1030000 + 117000 + 11000 +970 +107
=1159077

$unaka = 117 * 10^{(5-1)} + 110 * 10^{(4-1)} + 97 * 10^{(3-1)} + 107 * 10^{(2-1)} + 97 * 10^{(1-1)}$
$=117 * 10^4 + 110 * 10^3 + 97 * 10^2 + 107 * 10^1 + 97 * 10^0$
=1170000 + 110000 + 9700 +1070 +97
=1290867

$nakan = 110 * 10^{(5-1)} + 97 * 10^{(4-1)} + 107 * 10^{(3-1)} + 97 * 10^{(2-1)} + 110 * 10^{(1-1)}$
$=110 * 10^4 + 97 * 10^3 + 107 * 10^2 + 97 * 10^1 + 110 * 10^0$
=1100000 + 97000 + 10700 +970 +110

=1208780

$akana = 97 * 10^{(5-1)} + 107 * 10^{(4-1)} + 97 * 10^{(3-1)} + 110 * 10^{(2-1)} + 97 * 10^{(1-1)}$

$=97 * 10^4 + 107 * 10^3 + 97 * 10^2 + 110 * 10^1 + 97 * 10^0$

=970000 + 107000 + 9700 +1100 +97

=1087897

$kanal = 107 * 10^{(5-1)} + 97 * 10^{(4-1)} + 110 * 10^{(3-1)} + 97 * 10^{(2-1)} + 108 * 10^{(1-1)}$

$=107 * 10^4 + 97 * 10^3 + 110 * 10^2 + 97 * 10^1 + 108 * 10^0$

=1070000 + 97000 + 11000 +970 +108

=1179078

$analg = 97 * 10^{(5-1)} + 110 * 10^{(4-1)} + 97 * 10^{(3-1)} + 108 * 10^{(2-1)} + 103 * 10^{(1-1)}$

$=97 * 10^4 + 110 * 10^3 + 97 * 10^2 + 108 * 10^1 + 103 * 10^0$

=970000 + 110000 + 9700 +1080 +103

=1090883

$nalgo = 110 * 10^{(5-1)} + 97 * 10^{(4-1)} + 108 * 10^{(3-1)} + 103 * 10^{(2-1)} + 111 * 10^{(1-1)}$

$=110 * 10^4 + 97 * 10^3 + 108 * 10^2 + 103 * 10^1 + 111 * 10^0$

=1100000 + 97000 + 10800 +1030 +111

=1208941

$algor = 97 * 10^{(5-1)} + 108 * 10^{(4-1)} + 103 * 10^{(3-1)} + 111 * 10^{(2-1)} + 114 * 10^{(1-1)}$

$=97 * 10^4 + 108 * 10^3 + 103 * 10^2 + 111 * 10^1 + 114 * 10^0$

=970000 + 108000 + 10300 +1110 +114

=1089524

$lgori = 108 * 10^{(5-1)} + 103 * 10^{(4-1)} + 111 * 10^{(3-1)} + 114 * 10^{(2-1)} + 105 * 10^{(1-1)}$

$=108 * 10^4 + 103 * 10^3 + 111 * 10^2 + 114 * 10^1 + 105 * 10^0$

=1080000 + 103000 + 11100 +1140 +105

=1195345

$gorit = 103 * 10^{(5-1)} + 111 * 10^{(4-1)} + 114 * 10^{(3-1)} + 105 * 10^{(2-1)} + 116 * 10^{(1-1)}$

$=103 * 10^4 + 111 * 10^3 + 114 * 10^2 + 105 * 10^1 + 116 * 10^0$

=1030000 + 111000 + 11400 +1050 +116

=1153566

$oritm = 111 * 10^{(5-1)} + 114 * 10^{(4-1)} + 105 * 10^{(3-1)} + 116 * 10^{(2-1)} + 109 * 10^{(1-1)}$

$=111 * 10^4 + 114 * 10^3 + 105 * 10^2 + 116 * 10^1 + 109 * 10^0$

=1110000 + 114000 + 10500 +1160 +109

=1235769

$ritma = 114 * 10^{(5-1)} + 105 * 10^{(4-1)} + 116 * 10^{(3-1)} + 109 * 10^{(2-1)} + 97 * 10^{(1-1)}$

$=114 * 10^4 + 105 * 10^3 + 116 * 10^2 + 109 * 10^1 + 97 * 10^0$

=1140000 + 105000 + 11600 +1090 +97

=1257787

$itmam = 105 * 10^{(5-1)} + 116 * 10^{(4-1)} + 109 * 10^{(3-1)} + 97 * 10^{(2-1)} + 109 * 10^{(1-1)}$

$=105 * 10^4 + 116 * 10^3 + 109 * 10^2 + 97 * 10^1 + 109 * 10^0$

=1050000 + 116000 + 10900 +970 +109

=1177979

$tmami = 116 * 10^{(5-1)} + 109 * 10^{(4-1)} + 97 * 10^{(3-1)} + 109 * 10^{(2-1)} + 105 * 10^{(1-1)}$

$=116 * 10^4 + 109 * 10^3 + 97 * 10^2 + 109 * 10^1 + 105 * 10^0$

=1160000 + 109000 + 9700 +1090 +105

=1279895

$mamin = 109 * 10^{(5-1)} + 97 * 10^{(4-1)} + 109 * 10^{(3-1)} + 105 * 10^{(2-1)} + 110 * 10^{(1-1)}$

$=109 * 10^4 + 97 * 10^3 + 109 * 10^2 + 105 * 10^1 + 110 * 10^0$

=1090000 + 97000 + 10900 +1050 +110

=1199060

$amini = 97 * 10^{(5-1)} + 109 * 10^{(4-1)} + 105 * 10^{(3-1)} + 110 * 10^{(2-1)} + 105 * 10^{(1-1)}$

$=97 * 10^4 + 109 * 10^3 + 105 * 10^2 + 110 * 10^1 + 105 * 10^0$

=970000 + 109000 + 10500 +1100 +105
=1090705

$$minim = 109 * 10^{(5-1)} + 105 * 10^{(4-1)} + 110 * 10^{(3-1)} + 105 * 10^{(2-1)} + 109 * 10^{(1-1)}$$
$$=109 * 10^4 + 105 * 10^3 + 110 * 10^2 + 105 * 10^1 + 109 * 10^0$$
=1090000 + 105000 + 11000 +1050 +109
=1207159

$$inima = 105 * 10^{(5-1)} + 110 * 10^{(4-1)} + 105 * 10^{(3-1)} + 109 * 10^{(2-1)} + 97 * 10^{(1-1)}$$
$$=105 * 10^4 + 110 * 10^3 + 105 * 10^2 + 109 * 10^1 + 97 * 10^0$$
=1050000 + 110000 + 10500 +1090 +97
=1171687

$$nimax = 110 * 10^{(5-1)} + 105 * 10^{(4-1)} + 109 * 10^{(3-1)} + 97 * 10^{(2-1)} + 120 * 10^{(1-1)}$$
$$=110 * 10^4 + 105 * 10^3 + 109 * 10^2 + 97 * 10^1 + 120 * 10^0$$
=1100000 + 105000 + 10900 +970 +120
=1216990

## 4. KESIMPULAN

Berdasarkan hasil analisis dan pengujian terhadap aplikasi menghitung persentase kemiripan proposal dan isi skripsi dengan algoritma Rabin-Karp ini maka dapat disimpulkan bahwa:

1. Sistem dalam membandingkan *file* memberikan hasil berupa presentase *similarity*. Karena hasil kemiripan dinyatakan dalam bentuk persentase, maka jika persentase tinggi kecurigaan adanya kemiripan juga semakin tinggi. Ada kalanya sebuah kasus yang jarang terjadi, dua buah file yang tidak mirip namun memiliki *token* yang sama, oleh karena itu aplikasi ini digunakan hanya sebagai pertimbangan dan masih diperlukan koreksi manual oleh Kepala program studi.
2. Faktor yang mempengaruhi performa algoritma Rabin-Karp, diantaranya banyak konten sebuah *file* akan memperpanjang waktu prosesnya (*running time*).
3. Tahap preprocessing membuat waktu proses cenderung lebih lama tetapi tingkat akurasi *similarity*nya lebih tinggi.
4. Tingkat persentase *similarity* dipengaruhi oleh nilai k-gram yang digunakan dalam proses pengujian, semakin besar nilai *k-gram* yang digunakan maka akan semakin kecil persentase *similarity* yang didapat dari proses pengujian *file*.

## 5. SARAN

Penelitian lebih lanjut diharapkan sistem dapat menghitung kemiripan teks yang bersinonim, jika masih menggunakan algoritma Rabin-Karp lebih baik dibenahi tahap stemming agar hasil pencariannya lebih cepat. Diharapkan sistem dapat menguji dokumen teks dengan ekstensi lain seperti .doc .odt .txt . untuk kemudahan pengguna.

## DAFTAR PUSTAKA

[1] Afdhal, Chalis,T., and Gani,T.A., 2014, Analisa Perbandingan Aplikasi Pendeteksi Plagiat Terhadap Karya Ilmiah, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Syiah Kuala, Aceh, pp. 193-199.

[2] Ariyani,N.H., Sutardi, and Ramadhan,R., 2016, *Aplikasi Pendeteksi Kemiripan Isi Teks Dokumen Menggunakan Metode Levenshtein Distance*, Program Studi Teknik Informatika, Fakultas Teknik, Universitas Halu Oleo, Kendari, Vol.2, No.1, pp. 279-286, ISSN : 2502-8928.

[3] Firdaus,A., Ernawati dan Vatresia,A., 2014, Aplikasi Pendeteksi Kemiripan Pada Dokumen Teks Menggunakan Algoritma Nazief & Adriani Dan Metode Cosine Similarity. Program Studi Teknik Informatika, Fakultas Teknik, Universitas Bengkulu, Bengkulu, Vol.10, No.1, pp. 97-110.

[4] Firdaus,H.B., 2008, Deteksi Plagiat Dokumen Menggunakan Algoritma Rabin-Karp. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika , Institut Teknologi Bandung, Bandung, Vol.3, No.2.

[5] Kurniawan, Erick. 2011. Cepat Mahir Visual Basic 2010. Edisi satu. C.V ANDI OFFSET. Yogyakarta.

[6] Kurniawati,A., Wicaksana,I.W.S., 2008, Perbandingan Pendekatan Deteksi Plagiarisme Dokumen Dalam Bahasa Inggris, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Gunadarma, Depok, pp. 284-291.

[7] Noprisson,H., Susilo,B., and Ernawati, 2014, *Implementasi Algoritma Rabin-Karp Untuk Menentukan Keterkaitan Antarpublikasi Penelitian Dosen Tahun 2013*, Program Studi Teknik Informatika, Fakultas Teknik, Universitas Bengkulu, Bengkulu, Vol.10, No.1, pp. 110-127.

[8] Pratama,B.P., Pamungkas,S.A., 2016, Analisis Kinerja Algoritma Levenshtein Distance Dalam Mendeteksi Kemiripan Dokumen Teks. Program Studi Matematika, Fakultas Sains dan Teknologi, UIN Syarif Hidayatullah, Jakarta, Jilid 6, No.2, hal. 131-143.

[9] Putra,D.A., Sijaini,H., dan Pratiwi,H.S., 2015, Implementasi Algoritma Rabin-Karp untuk Membantu Pendeteksian Plagiat pada Karya Ilmiah, Program Studi Teknik Informatika, Fakultas Teknik, Universitas Tanjungpura, Vol.1, No.1.

[10]Putri,D.R., Mardiyono dan Handoko,S., 2013, Portal Open Source Pendeteksi Plagiarisme di Kalangan Negeri Semarang, Jurusan Teknik Elektro, Politeknik Negeri Semarang, Semarang, Vol.2, No.2.

[11]Rinusantoro,S. (2014). *Aplikasi Deteksi Kemiripan Dokumen Teks*. Tesis. Yogyakarta: Universitas Gadjah Mada.

[12]Salmuasih, Sunyoto,A., 2013, Implementasi Algoritma Rabin Karp untuk Pendeteksian Plagiat Dokumen Teks Menggunakan Konsep Similarity, Program Studi teknik Informatika, STMIK AMIKOM Yogyakarta, Yogyakarta, pp. 23-28.