Jurnal Mantik Penusa

Volume 22, No. 1, Agustus 2018

e-ISSN 2580-9741
p-ISSN 2088-3943

PENERAPAN ALGORITMA RABIN KARP UNTUK MENDETEKSI KEMIRIPAN JUDUL SKRIPSI

Sumarni Adi

Informatika
Universitas Amikom Yogyakarta, Jl. Ring Road Utara, Yogyakarta, 55281 Indonesia

sumarni.a@amikom.ac.id

Abstract

Every final year student who has completed his lecture must definitely make a thesis in order to get his degree. Before making a thesis, students must propose a thesis title to the study program admission section first. The title will be checked for feasibility by the study program. One of the requirements of the thesis title is said to be feasible, the title has never been used by other students. But every semester there are so many students who submit thesis titles, so the study program is overwhelmed if they have to match one by one the incoming thesis titles with the previous thesis title because it requires a lot of energy and time. Therefore, a system needs to be developed to detect the similarity of the thesis title so that the process of assessing the feasibility of the title can be done faster and easier. Rabin Karp is an algorithm that can detect similarities in documents, using the hash method in searching for a word. This theory is rarely used to find a single word, but it is quite important and very effective when used for multiple searches. For this reason, in this study Rabin Karp's ability to detect the similarity of the thesis title will be proven by giving the title similarity value. Thesis title is preprocessing, so that the data becomes "clean", so it is feasible to do the hashing process. After hashing, then apply the Rabin Karp algorithm to each word and measure its similarity using dice's Coefficient, resulting in a similarity value from the title of the thesis. The better the stemming process in the preprocessing process, the higher the similarity value.

Keywords*: Thesis Title, Similarity, Rabin Karp*

Abstrak

Setiap mahasiswa tingkat akhir pasti harus membuat skripsi agar mendapatkan gelar kesarjanaannya. Sebelum membuat skripsi, mahasiswa harus mengusulkan judul skripsi ke bagian admisi program studinya terlebih dahulu. Judul tersebut akan dicek kelayakannya oleh tim verifikasi judul skripsi yang ditunjuk oleh Ketua program studi. Salah satu syarat judul skripsi dikatakan layak adalah judul tersebut belum pernah digunakan oleh mahasiswa lainnya. Namun setiap semester begitu banyak mahasiswa yang mengajukan judul skripsi, sehingga pihak program studi kewalahan jika harus mencocokan satu persatu judul skripsi yang masuk dengan judul skripsi sebelumnya karena membutuhkan energi dan waktu yang cukup banyak. Oleh karena itu, perlu dikembangkan sistem untuk mendeteksi kemiripan judul skripsi agar proses penilaian kelayakan judul dapat dilakukan lebih cepat dan lebih mudah. Rabin Karp merupakan Algoritma yang dapat mendeteksi kemiripan di dalam dokumen, dengan menggunakan metode *hash* dalam mencari suatu kata. Teori ini jarangdigunakan untuk mencari kata tunggal, namun cukup penting dan sangat efektif biladigunakan untuk pencarian jamak. Untuk itu, dalam penelitian ini akan dibuktikan kemampuan Rabin Karp untuk mendeteksi kemiripan judul skripsi dengan memberikan nilai kemiripan (*similarity*)judul tersebut. Judul skripsi dilakukan preprocessing, agar data tersebut menjadi "bersih", sehingga layak untuk dilakukan proses *hashing*. Setelah dilakukan *hashing*, selanjutnya*stringmatching* menggunakan Rabin Karp pada setiap katadan ukur *similarity*-nya menggunakan *dice'sCoefficient*, sehingga menghasilkan nilai *similarity*dari judul skripsi tersebut.

Kata Kunci : Judul Skripsi, Kemiripan, *Rabin Karp*

1. PENDAHULUAN

Setiap mahasiswa tingkat akhir pasti harus membuat skripsi agar mendapatkan gelar kesarjanaannya. Sebelum membuat skripsi, mahasiswa harus mengusulkan judul skripsi ke bagian admisi program studinya terlebih dahulu. Judul tersebut akan dicek kelayakannya oleh tim verifikasi judul skripsi yang ditunjuk oleh Ketua program studi. Salah satu syarat judul skripsi dikatakan layak adalah judul tersebut belum pernah digunakan oleh mahasiswa lainnya. Namun setiap semester begitu banyak mahasiswa yang mengajukan judul skripsi, hal ini terjadi sepanjang tahun. Sehingga pihak program studi kewalahan jika harus mencocokan satu persatu judul skripsi yang masuk dengan judul skripsi sebelumnya, karena membutuhkan energi dan

waktu yang cukup banyak.

Perlu dilakukan terobosan agar mudah mendeteksi kemiripan judul skripsi. Karena perkembangan dunia pendidikan yang sangat pesat, mendorong terbentuknya suatu timbunan dokumen yang berukuran sangat besar di tingkat perguruan tinggi. Dokumen-dokumen tersebut pada umumnya berasal dari data entry, kemudian oleh komputer data tersebut disimpan ke dalam server. Informasi yang didapat dari dokumen tersebut sangat sedikit yang dapat dimanfaatkan oleh pihak manajemen perguruan tinggi dalam menganalisis kemiripan judul skripsi, oleh karena itu perlu adanya aktivitas penggalian (ekstraksi) dokumen yang masih tersembunyi untuk selanjutnya diolah menjadi pengetahuan yang bermanfaat dalam pengambilan keputusan. suatu proses menggali informasidimana seorang *user* berinteraksi dengan sekumpulan dokumen menggunakan *tools*analisis yang merupakan komponen-komponen dalam *data mining* dengan tujuanuntuk mendapatkan informasi yang berguna dari sekumpulan dokumendisebut dengan *textmining*[1].

Konsep kemiripan (*similarity)*sudah menjadi isu yang sangat penting dihampir setiap bidang ilmu pengetahuan[2], Pendeteksian menggunakan konsep *similarity* dokumen merupakan salah satu cara untuk mendeteksi kemiripan judul skripsi.*Dice's Coefficient* merupakan salah satu metode untuk mengukur kemiripan, dengan cara mengukur tingkat kesamaan dua buah objekdari segi jarak geometris dari variabel-variabel yang tercakup di dalam kedua objektersebut[2]. Agarmendapatkan kesamaan dari dua objek, maka diperlukan pencocokan string (*stringmatching*). *String matching* merupakan bahasan yang sangat penting dalam ilmu komputer, karena teks adalah bentuk utama dalam pertukaran informasi yang merupakan subjek penting terkait dengan *text-processing*. Ada banyak algoritma yang termasuk dalam *string matching*, diantaranya algoritma Naive, algoritma Rabin Karp, algoritmaFinite Automaton, dan algoritma Knuth Morris Pratt[3]. Namun dalam penelitian ini akan dikembangkan proses pencocokan string menggunakan algoritma Rabin Karp. Karena algoritma ini sangat efektif bila digunakan untuk pencarian kata/pola jamak [4]. Dengan *stringmatching* menggunakan algoritma rabin karp, diharapkan mampu mendeteksi kemiripan judul skripsi. sehingga dapat digunakan untuk menganalisis kelayakan judul skripsi mahasiswa.

2. TINJAUAN PUSTAKA

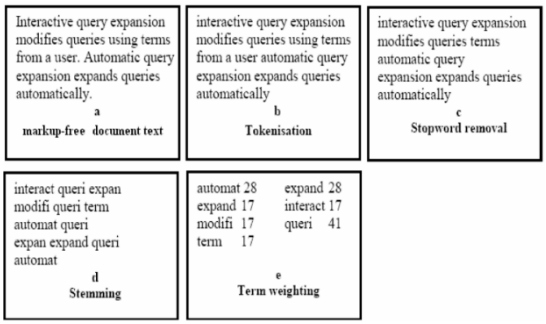
a. *TextMining*

*Textmining*adalah salah satu bidang khusus dari *data mining*. [1]mendefinisikan *text mining* sebagai suatu proses menggali informasidimana seorang *user* berinteraksi dengan sekumpulan dokumen menggunakan *tools*analisis yang merupakan komponen-komponen dalam *data mining*. Tujuan dari *textmining* adalah untuk mendapatkan informasi yang berguna dari sekumpulan dokumen.Dalam memberikan solusi, *text mining* mengadopsi dan mengembangkan banyakteknik dari bidang lain, seperti *Data mining, Information Retrieval* (IR), *Statistic and Mathematic, Machine Learning, Linguistic, Natural Languange Processing* (NLP), dan*Visualization*[5]. Kegiatan riset untuk *text mining* salah satunya adalah *preprocessing* akan konten teks, menurut [6] terdapat 5 langkah dan contohnya ditunjukkan pada Gambar 1, yaitu:

1) Penghapusan format dan *markup* dari dalam dokumen. Tahap ini menghapus semua *tag markup* dan format khusus dari dokumen, terutama pada dokumen yang mempunyai banyak *tag* dan format seperti dokumen (X)HTML. Jika isi dokumen telah berada di dalam database maka tahapan ini sering dilewatkan.
2) Pemisahan rangkaian kata (*tokenization*). *Tokenization* adalah tugas memisahkan deretan kata di dalam kalimat, paragraf atau halaman menjadi *token* atau potongan kata tunggal atau *termmed word*. Tahapan ini juga menghilangkan karakter-karakter tertentu seperti tanda baca dan mengubah semua *token* ke bentuk huruf kecil (*lower case*).
3) Penyaringan (*filtration*). Pada tahapan ini ditentukan *term* mana yang akan digunakan dengan Penghapusan *stop-word* dari dalam suatu koleksi dokumen .
4) Konversi *term* ke bentuk akar (*stemming*). *Stemming* adalah proses konversi term ke bentuk dasarnya.
5) Pemberian bobot terhadap term (*weighting*). Setiap term diberikan bobot sesuai dengan skema pembobotan yang dipilih, Banyak aplikasi menerapkan pembobotan kombinasi berupa perkalian bobot lokal *term frequency* dan global *inverse document frequency*, ditulis *tf .idf*. Adapun rumus untuk menghitung *tf.idf* dapat dilihat di Persamaan 1.

$$Tf.idf\ (d, w) = tf\ (d, w) \times \log N/dfw \quad (1)$$

Dimana: tf(d,w) adalah frekuensi kemunculan term w pada dokumen d, n adalah jumlah keseluruhan dokumen dan dfw adalah jumlah dokumen yang mengandung term w.



Gambar 1. Contoh lima tahap *preprocessing*secara urutmulai dari *markup removal* (a), *tokenization* (b), *stopwords filtration* (c), *stemming* (d)dan *weighting* (e)

Tujuan dari proses data *preprocessing* adalah untuk mengubah data input mentah menjadi format yang sesuai untuk analisis selanjutnya. Langkah-langkah yang dilakukan antara lain dengan memperbaiki data yang 'kotor', memilih fitur-fitur dari data yang relevan dengan proses pengolahan selanjutnya. Karena banyak cara dalam proses pengumpulandata dan penyimpanan data, maka proses pengolahan data mungkin akan memakan waktu yang lama dalam keseluruhan proses penemuan pengetahuan [7].

b. Stemming Indonesian Porter Stemmer

Stemming adalah proses konversi term ke bentuk dasarnya (*root word*), dengan menggunakan aturan-aturan tertentu. *Stemming* yang digunakan pada penelitian ini adalah *Indonesian Porter Stemmer.* Menurut [5], Algoritma *Porter* ditemukan oleh Martin Porter 1980. Algoritma inidigunakan untuk *stemming* bahasa Inggris, kemudian karena proses *stemming* BahasaInggris berbeda dengan bahasa Indonesia, maka dikembangkan algoritma *Porter* khususuntuk bahasa Indonesia (*Porter Stemmer for* Bahasa Indonesia) oleh W.B. Frakes padatahun 1992.Algoritma/langkah-langkah pada Porter Stemmer:

1) Menghapus partikel seperti: *-kah, -lah, -tah*
2) Mengapus kata ganti (Possesive Pronoun), seperti *–ku, -mu, -nya*
3) Mengapus awalan pertama.
   Jika tidak ditemukan, maka lanjut ke langkah 4a, dan jika ada maka lanjut ke langkah 4b.
4) a. Menghapus awalan kedua, dan dilanjutkan pada langkah ke 5a
   b. Menghapus akhiran, jika tidak ditemukan maka kata tersebut diasumsikan sebagai kata dasar (*root word*). Jika ditemukan maka lanjut ke langkah 5b.
5) a. Menghapus akhiran dan kata akhir diasumsikan sebagai kata dasar (*root word*).
   b. Menghapus awalan kedua dan kata akhir diasumsikan sebagai kata dasar (*root word*)

c. Rabin Karp

Algoritma Rabin Karp ditemukan oleh Michael O. Rabin dan Richard M. Karp.Algoritma ini menggunakan metode *hash* dalam mencari suatu kata. Teori ini jarangdigunakan untuk mencari kata tunggal, namun cukup penting dan sangat efektif biladigunakan untuk pencarian jamak [4].

Rabin Karp merepresentasikan setiap karakter ke dalam bentuk desimal digit(*digit radix-d*) $\sum = \{0, 1, 2, 3, \ldots, d\}$, dimana d = $|\sum|$. Sehingga didapat masukan *string k*berturut-turut sebagai perwakilan panjang *k* desimal. Karakter *string* 31415 sesuaidengan jumlah desimal 31,415. Kemudian pola *p* dihash menjadi nilai desimal dan *string*direpresentasikan dengan penjumlahan *digit-digit* angka menggunakan aturan Horner's,misal:

{ A, B, C, ..., Z } → { 0, 1, 2, ..., 26 }
• BAN → 1 + 0 + 13 = 14
• CARD → 2 + 0 + 17 + 3 = 22

Untuk pola yang panjang dan teks yang besar, algoritma ini menggunakanoperasi *mod*, setelah dikenai operasi *mod q*, nilainya akan menjadi lebih kecil dari *q*,misal:

• BAN = 1 + 0 + 13 = 14
= 14 *mod* 13 = 1
= BAN → 1
• CARD= 2 + 0 + 17 + 3 = 22
= 22 *mod* 13 = 9
= CARD → 9

Sedangkan *pseudocode* dan rumus matematis yang digunakan adalah sebagaiberikut[3] :

```
RABIN-KARP-MATCHER (T, P, d, q)
n  = T.length
m  = P.length
h  = d^(m-1) mod q
p  = 0
t0 = 0
for i = 1 to m                                  // preprocessing
    p = (dp + P[i]) mod q
    t0 = (dt0 + T[i]) mod q
for s = 0 to n – m
    if p == ts                                  // matching
            if P[1 .. m] == T[s + 1 .. s + m]
                    print "Pattern occurs with shift" s
    if s < n – m
            ts+1 = (d(ts - T[s + 1] h) + T[s + m + 1]) mod q
```

Adapun langkah-langkah algoritma rabin karp adalah sebagai berikut :

1) *Parsing*, yaitu term yang sudah melalui proses Preprocessing dipotong-potong per karakter huruf. Pemotongan per karakter menggunakan metode *k-Grams.* Cara kerj*a* metode *k-grams* dengan mengambil potongan-potongan karakter huruf sejumlah *k* dari sebuah kata yang secara kontinyu dibaca dari teks sumber hingga akhir dari dokumen. contoh *k-grams* dengan *k* = 4:

Teks : evenifnone
Hasil 4-*grams dari teks* :
even | veni | enif | nifn | ifno | fnon | none

2) *Hashing*, yaitu mengkonversi potongan huruf sejumlah *k* kedalam nilai *hash*. Biasanya dikonversi ke ASCII. Persamaan 1 adalah rumus dari hashing Rabin Karp.

$$Hash = T[i]b^{m-1} + T[i]b^{m-2} + \cdots + T[i+m-1] mod\ q \ldots (1)$$

Dimana,
T[i] = nilai ASCII huruf indeks ke-i
b = radix desimal (bilangan basis 10)
m = panjang teks
q = jumlah pola

3) Meningkatkan performansi rabin karp dengan Persamaan 2, menurut [8]memberikan solusi untuk tidak hanya membandingkan sisa hasil bagi, tetapi membandingkan hasil baginya juga.
REM (n1/q) = REM (n2/q)
and
QUOTIENT (n1/q) = QUOTIENT (n2/q) ...(2)

d. Pengukuran Nilai Similarity

Menurut [9], Karena parsing dilakukan dengan pendekatan *k-gram*maka untuk mengukur nilai similarity menggunakan *Dice'sSimilarityCoefficient*, rumus matematisnya ditunjukkan pada Persamaan 3.

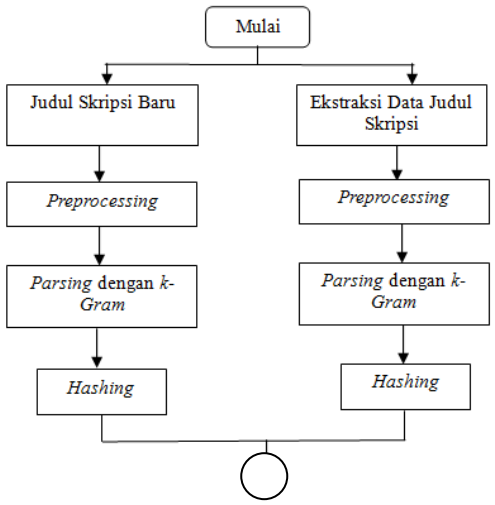
$$S = \frac{K \times C}{A+B} \text{ .....(3)}$$

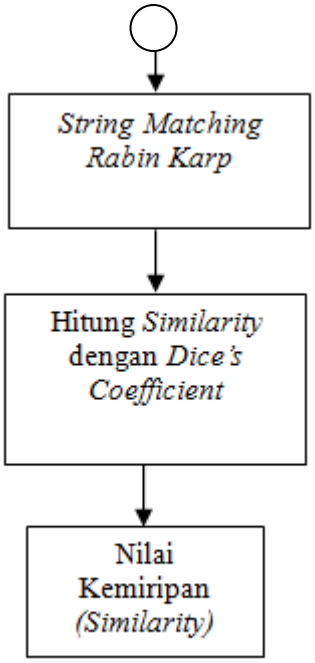
Dimana S adalah nilai *similarity*, A dan B adalah jumlah dari kumpulan *k-grams* dalam teks 1 dan teks 2. C adalah jumlah dari *k-grams* yang sama dari teks yang dibandingkan.

Penggunaan ukuran *similarity* yang tepat tidak hanya meningkatkan kualitaspilihan informasi tetapi juga membantu mengurangi waktu dan biaya proses [2].

3. METODE PENELITIAN

Metode penelitian yang dilakukan seperti ditunjukka pada Gambar 2.





Gambar 2. Alur Metodologi Penelitian

Dari Gambar 2 dapat dijelaskan sebagai berikut:

a. Usulan Judul skripsi diinputkan ke sistem dan data judul skripsi yang sudah ada diekstraksi
b. Kedua data tersebut mengalami *preprocessing*. Proses yang dialami pada saat *preprocessing* adalah :
   1) *Tokenization*, yaitu memisahkan deretan kata dalam kalimat menjadi potongan kata tunggal (*token*), menghilangkan tanda baca dan mengubah semua token ke bentuk huruf kecil.
   2) *Filtration*, yaitu penghapusan kata-kata yang sering ditampilkan dalam dokumen seperti: kata hubung, kata depan dan sebagainya.
   3) *Stemming*, yaitu mengubah *term*menjadi bentuk dasarnya. Proses *stemming*menggunakan algoritma *Indonesian Porter Stemmer*
c. Setelah mengalami *preprocessing*, selanjutnya adalah *parsing*. *Parsing*, yaitu *term* yang sudah melalui proses *Preprocessing* dipotong-potong per karakter huruf. Pemotongan per karakter menggunakan metode k-Gram.
d. Hasil dari pemotongan menggunakan k-gram, dilanjutkan proses hashing, yaitu mengubah huruf-huruf tersebut ke ASCII
e. Setelah di*hashing*, selanjutnya lakukan string matching menggunakan algoritma Rabin Karp pada Persamaan 1 dan juga menggunakan Persamaan 2 untuk meningkatkan performansi rabin karp. Sehingga pencocokannya tidak dari

hasil modulo saja, tapi dari hasil pembagiannya juga.

f. Setelah nilai *string matching* sudah didapatkan, maka dilanjutkan dengan menghitung *similarity* judul skripsi menggunakan *Dice'sCoefficient.*
g. Informasi *similarity* akan ditampilkan dengan diberikan presentase nilai kemiripannya.

4. HASIL DAN PEMBAHASAN

a. Hasil Preprocessing

Gambar 3 merupakan contoh data judul skripsi hasil ekstraksi. Judul skripsi yang digunakan untuk sampel ada 117 judul.

idJudulSkripsi	judulSkripsi
121	Analisa Dan Perancangan Sistem Informasi Penggajian Karyawan PT. Indonesia Raya Audivisi
122	Analisa Dan Perancangan Sistem Pengisian Formulir Rencana Studi Secara Online Di Stimik Perbanas
123	Analisa Struktur Kalimat Bahasa Indonesia dengan Menggunakan Pengurai Kalimat Berbasis Linguistic
124	Analisa Traffic Internet pada Jaringan Local Area Network SMU Negeri 13
129	Aplikasi Text To Speech (TTS) Berbahasa Indonesia Sebagai Pembaca SMS
130	Detektor Pengaman Rumah
133	IEEE 802.11b Sebagai Standar Teknologi Jaringan Komputer Nirkabel
140	Monitoring Ruangan Menggunakan Kamera IP Dengan Bahasa Personal Home Page PHP

Gambar 3. Judul Skripsi Hasil Ekstraksi

Judul skripsi hasil ekstraksi dan judul skripsi baru yang dinputkan, akan mengalami *preprocessing*. Berikut pada Gambar 4 merupakan hasil *preprocessing* judul skripsi.

idJudulSkripsi	Bersih_judulskripsi
121	analisarancangsisteminformasigajikaryawanptindonesiarayaaudivisi
122	analisarancangsistemisiformulirrencanastudionlinestimikperbanas
123	analisisstrukturkalimatbahasaindonesiagunauraikalimatbasislinguistic
124	analisatrafficinternetjaringanlocalareanetworksmunegeri
129	aplikasitexttospeechttsbahasaindonesiabacasms
130	detektoramanrumah
133	ieeebstandarteknologijaringankomputernirkabel
140	monitoringruanggunakameraipbahasapersonalhomepagephp

Gambar 4. Judul Skripsi Hasil *Preprocessing*

b. Proses Rabin Karp

Contoh perhitungan manual untuk pengecekan dua buah file yang sudah mengalami proses *preprocessing*.

File 1 : detektoramanrumah
File 2 : aplikasiwebtraffic

File 1 dan *file* 2 dilakukan *parsing*, dan dilanjutkan proses *hashing* menggunakan modulo (sisa bagi) dengan 101 dan *hashing* menggunakan hasil pembagian dengan 101, serta nilai *match*-nya bernilai "ya" jika cocok dan bernilai "tidak" jika tidak cocok.

Pada Gambar 5 dapat dilihat bahwa*Parsing* menggunakan *k-gram* dengan *k=4*, pada setiap teks pada *File* 1 dan *File* 2.Dan dilanjutkan proses *hashing* dengan mengkonversi string menjadi nilai ASCII. a-z = 97-122), dan dilanjutkan dengan proses *rabin karp* menggunakan Persamaan 1dan peningkatan performa *rabinkarp* dengan Persamaan 2.Penghitungan nilai *hash* dengan modulo 101 dan nilai *hash* dibagi dengan 10, sebagai berikut:

$$\text{dete} = (100*10^3) + (101*10^2) + (116*10^1) + (101*10^0)$$
$$= 100000 + 10100 + 1160 + 101$$
$$= 111361$$

$$\text{Modulo} = 111361 \bmod 101$$
$$= 59$$
$$\text{Div} = 111361 / 101$$
$$= 1102{,}58416$$

No	File 1			File 2			Match
	Parsing	Hashing Modulo	Hashing Div	Parsing	Hashing Modulo	Hashing Div	
1	dete	59	1102,58416	apli	2	1083,0198	Tidak
2	etek	92	1125,91089	plik	30	1227,29703	Tidak
3	tekt	26	1260,25743	lika	83	1184,82178	Tidak
4	ekto	53	1118,52475	ikas	29	1156,28713	Tidak
5	ktor	38	1186,37624	kasi	88	1167,87129	Tidak
6	tora	67	1270,66337	asiw	84	1085,83168	Tidak
7	oram	57	1222,56436	siwe	36	1255,35644	Tidak
8	rama	51	1236,50495	iweb	40	1168,39604	Tidak
9	aman	1	1079,0099	webt	7	1289,06931	Tidak
10	manr	27	1187,26733	ebtr	65	1109,64356	Tidak
11	anru	76	1081,75248	btra	40	1097,39604	Tidak
12	nrum	65	1214,64356	traf	0	1272	Tidak
13	ruma	31	1256,30693	raff	87	1235,86139	Tidak
14	umah	98	1276,9703	affi	53	1072,52475	Tidak
15				ffic	27	1122,26733	Tidak

Gambar 5. Hasil Perhitungan Proses Rabin Karp

c. Hasil Kemiripan (*Similarity*)

Dari Tabel 5 dapat dilihat bahwa baris satu tidak cocok karena nilai hash dan reminder antara *file*1 dan *file*2 tidak sama. Jadi teks dokumen cocok ketika nilai hashing modulo dan hashing div sama.

Untuk menghitung *similarity* (kecocokan) menggunakan Persamaan 3. Jadi similarity teks *file* 1 dan *file* 2adalah

Similarity= ((2*0) / (14+15)) * 100%
= (0/29) * 100%
= 0* 100%
= 0 %

Dari hasil tersebut, teks file 1 dan file 2 tidak mirip karena hasil kemiripannya adalah 0%.

5. KESIMPULAN

Berdasarkan percobaan-percobaan yang dilakukan dapat disimpulkan sebagi berikut:

1. Sistem dalam membandingkan *file* memberikan hasil berupa prosentase *similarity*
2. *Preprocessing* sangat membantu sekali dalam proses rabin karp. Karena dengan melakukan preprocessing maka dapat mengurangi volume dataset tanpa mengurang esensi nilai dataset tersebut, sehingga proses parsing jumlah k-nya menjadi lebih sedikit
3. Jika file tidak mengalami proses *preprocessing*, maka waktu yang diperlukan semakin kecil namun nilai *similaritas*-nya berkurang

Penelitian ini masih sangat jauh dari

6. REFERENSI

[1] F. Ronen and J. Sanger, *The Text Mining Handbook*. New York: Cambridge University Press, 2007.
[2] B. Zaka, "Theory and Applications of Similarity Detection Technique," Graz University of Technology Austria, 2009.
[3] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, Third Edit., vol. 25, no. 4. Cambridge, Massachusetts London, England: The MIT Press, 2009.
[4] A. Atmopawiro, "Pengkajian dan Analisis Tiga Algoritma Efisien Rabin-Karp, Knuth-Morris-Pratt, dan Boyer-Moore dalam Pencarian Pola dalam Suatu Teks," *Makalah tidak Terpublikasi*, 2006.
[5] Salmuasih, "Perancangan sistem deteksi plagiat pada dokumen teks dengan konsep similarity menggunakan algoritma rabin karp naskah publikasi," *Makalah tidak Terpublikasi*, pp. 1–20, 2013.
[6] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval Introduction*. England: Cambridge University Press, 2009.
[7] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, First edit. England: Pearson Education Limited, 2014.
[8] R. S. Chillar and B. Kochar, "RB-Matcher: String Matching Technique," *World Academic Science EngineeringTechnology. 42 2008*, vol:2 No:6, pp. 132–135, 2008.
[9] S. Kosinov, "Evaluation of N-Grams Conflation Approach In Text-Based Information Retrieval," *Proceedings. Eighth Symposium on String Processing and Information. Retrieval*, 2001.