Journal of Informatics Management and Information Technology
ISSN 2774-4744 (Media Online)
Vol 2, No 1, Januari 2022
Hal 12−17
https://hostjournals.com/jimat

Penerapan Algoritma Rabin-Karp Pada Pencarian Sinonim Kata

Mega Oktaviani Pohan, Mesran, Ronda Deli Sianturi

Program Studi Teknik Informatika, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Budi Darma, Medan
Jl. Sisingamangaraja No.338, Siti Rejo I, Kec. Medan Kota, Kota Medan, Sumatera Utara, Indonesia
Email: [1]vianimega01@gmail.com, [2]mesran.skom.mkom@gmail.com

Abstrak−Sinonim adalah suatu kata yang memiliki bentuk yang berbeda namun memiliki arti atau pengertian yang sama atau mirip. Sinonim biasa disebut juga dengan persamaan kata atau padanan kata. Sinonim digunakan untuk menyatakan sameness of meaning 'kesamaan arti'. Dalam kegiatan komunikasi sehari – hari, para pemakai bahasa masih sering melakukan kesalahan dalam menggunakan kata – kata yang bersinonim itu. Seringkali bentuk kebahasaan yang berbeda – beda begitu saja dianggap sinonim. Oleh karena itu, penguasaan sinonim secara benar harus dimiliki oleh para pemakai bahasa untuk kegiatan komunikasi sehari – hari (baik lisan maupun tulis). Permasalahan yang selama ini dihadapi adalah kurangnya sarana yang fleksibel untuk memudahkan di dalam pencarian sinonim kata tersebut. Untuk mengatasi masalah tersebut dibutuhkan sebuah aplikasi sederhana yang dioperasikan pada smartphone berbasis android yang berguna membantu kita mencari sinonim kata. Pengguna hanya tinggal memasukan kata yang ingin dicari kesinoniman katanya. Aplikasi ini nantinya akan menggunakan algoritma pencocokan string atau yang disebut juga dengan String matching atau adalah suatu metode yang digunakan untuk menemukan suatu keakuratan atau hasil dari satu atau beberapa pola teks yang diberikan. Hasil yang diharapkan pada aplikasi ini agar membantu mempercepat proses dalam mencari sinonim kata bahasa Indonesia, meningkatkan pengetahuan dan penguasaan kosakata para pemakai bahasa Indonesia, khususnya kata sinonim.

Kata Kunci: Sinonim Kata; String Matching; Rabin-Karp

Abstract−Synonym is a word that has a different form but has the same or similar meaning or understanding. Synonyms are also known as word equations or word equivalents. Synonyms are used to express the sameness of meaning. In daily communication activities, language users still often make mistakes in using synonymous words. Often different linguistic forms are simply considered synonyms. Therefore, the correct mastery of synonyms must be possessed by language users for daily communication activities (both oral and written). The problem that has been faced so far is the lack of flexible means to facilitate the search for synonyms for these words. To overcome this problem, we need a simple application that is operated on an Android-based smartphone that is useful for helping us find synonyms for words. Users only need to enter the word they want to find synonyms for. This application will use a string matching algorithm or also known as String matching or is a method used to find an accuracy or result from one or several text patterns given. The expected results in this application are to help speed up the process of finding synonyms for Indonesian words, increase knowledge and vocabulary mastery of Indonesian users, especially synonyms.

Keywords: Synonyms; String Matching; Rabin-Karp

1. PENDAHULUAN

Sinonim adalah suatu kata yang memiliki bentuk yang berbeda namun memiliki arti atau pengertian yang sama atau mirip. Sinomin bisa disebut juga dengan persamaan kata atau padanan kata. Sinonim digunakan untuk menyatakan *sameness of meaning* 'kesamaan arti'. Hal tersebut dilihat dari kenyataan bahwa para penyusun kamus menunjukkan sejumlah perangkat kata yang memiliki makna sama; semua bersifat sinonim, atau satu sama lain sama makna, atau hubungan di antara kata-kata yang mirip (dianggap sama) maknanya [1].

Dalam kegiatan komunikasi sehari-hari, para pemakai bahasa masih sering melakukan kesalahan dalam menggunakan kata-kata yang bersinonim itu. Semua itu dikarenakan kekurangtahuan mereka terhadap nilai makna suatu kata maupun kelompok kata. Seringkali bentuk kebahasaan yang berbeda-beda begitu saja dianggap sinonim, misalnya antara bentuk "kembali ke pangkuan ilahi" dengan "meninggalkan dunia kehidupan", antara "merencanakan" dengan "menginginkan", serta antara "gambaran" dengan "bayangan". Selain itu, suatu kata yang semula memiliki kolokasi sangat ketat, misalnya antara kata "kopi" dengan "minuman", "kuncup" dengan "kembang", maupun "pohon" dengan "batang" seringkali dipakai secara tumpang tindih karena masing-masing kata tersebut dianggap memiliki kesinoniman.

Pemakaian yang tumpang tindih tersebut dapat mengakibatkan adanya salah pengertian. Kesalahan lain yang biasa dilakukan oleh para pemakai bahasa adalah penggunaan pasangan kata ilmu dan pengetahuan. Pasangan-pasangan kata tersebut dianggap bersinonim, padahal berdiri sendiri-sendiri karena memiliki karakteristik sendiri. Adanya kesalahan dalam penentuan fitur semantis kata yang satu dengan kata lainnya tersebut dapat menimbulkan kejanggalan dan kesalahan penerimaan informasi.

Oleh karena itu, penguasaan sinonim secara benar harus dimiliki oleh para pemakai bahasa untuk kegiatan komunikasi sehari-hari (baik lisan maupun tulis), terutama yang berkaitan dengan diksi. Permasalahan yang selama ini dihadapi adalah kurangnya sarana yang fleksibel untuk memudahkan didalam pencarian sinonim kata tersebut.

Untuk mengatasi masalah tersebut dibutuhkan sebuah aplikasi sederhana yang dioperasikan pada *smartphone* berbasis *android* yang berguna membantu kita mencari sinonim kata. Pengguna hanya tinggal memasukan kata yang ingin di cari kesinoniman katanya. Aplikasi ini nantinya akan menggunakan algoritma pencocokan *String Maching*.

Algoritma pencocokan *string* atau yang disebut juga dengan *String matching* atau adalah suatu metode yang digunakan untuk menemukan suatu keakuratan atau hasil dari satu atau beberapa pola teks yang diberikan. Pencocokan *string* merupakan pokok bahasan yang penting dalam ilmu komputer karena teks merupakan adalah bentuk utama dari pertukaran informasi antar manusia, misalnya pada literatur, karya ilmiah, halaman web dan sebagainya [2].

ISSN 2774-4744 (Media Online)
Vol 2, No 1, Januari 2022
Hal 12−17
https://hostjournals.com/jimat

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 [3].

Pada penelitian yang dilakukan oleh Abu Salam dengan judul "Sistem Rekomendasi Penentuan Dosen Pembimbing Tugas Akhir Dengan Menggunakan Algoritma Rabin-Karp" menghasilkan bahwa algoritma *Rabin-Karp* berhasil diterapkan pada sistem rekomendasi penentuan dosen pembimbing tugas akhir sehingga memudahkan mahasiswa untuk memilih dosen pembimbing [4].

2. METODOLOGI PENELITIAN

2.1 String Matching

String Matching adalah proses pencarian semua kemunculan query yang selanjutnya disebut *pattern* ke dalam *string* yang lebih panjang (teks). *Pattern* di lambangkan dengan x=x[0..m-1] dan panjangnya adalah m. Teks dilambangkan dengan y=y[0..n-1] dan panjangnya adalah n. Kedua *string* terdiri dari sekumpulan karakter yang disebut alfabet yang dilambangkan dengan ∑ dan mempunyai ukuran σ. *String matching* dibagi menjadi dua, yakni *exact matching* dan *heuristic* atau *statistical matching*[5]. Algoritma *string matching* adalah suatu metode yang digunakan untuk menemukan suatu keakuratan atau hasil dari satu atau beberapa pola teks yang diberikan. *String matching* merupakan pokok bahasan yang penting dalam ilmu komputer karena teks merupakan bentuk utama dari pertukaran informasi antar manusia, misalnya pada literatur, karya ilmiah, halaman web, dan sebagainya.Pencocokan *string* juga dapat digunakan untuk mencari pola bit dalamsejumlah besar file *binary*. Dalam algoritma *string matching*, teks diasumsikan berada di dalam *memory*, sehingga bila kita mencari *string* di dalam sebuah teks, maka semua isi teks perlu dibaca terlebih dahulu kemudian disimpan didalam memory. *String matching* fokus pada pencarian satu, atau lebih umum, semua kehadiran sebuah kata *(pattern*) dalam sebuah teks. Semua algoritma yang akan dibahas mengeluarkan semua kehadiran pola dalam teks[6].

2.2 Algoritma*Rabin-Karp*

Algoritma *Rabin-Karp* diciptakan oleh Michael O. Rabin dan Richard M. Karp pada tahun 1987 yang menggunakan fungsi *hashing* untuk menemukan *pattern* di dalam *string* teks[9]. Charas and Lecroq, menyatakan bahwa fungsi hashing menyediakan metode sederhana untuk menghindari perbandingan jumlah karakter yang kuadratik di dalam banyak kasus atau situasi. Dari pada melakukan pemeriksaan terhadap setiap posisi dari teks ketika terjadi pencocokan pola, akan lebih baik dan efisien melakukan pemeriksaan hanya jika teks yang sedang proses memiliki kemiripan seperti pada *pattern*. Untuk melakukan pengecekan kemiripan antaradua kata ini digunakan fungsi hash. Algoritma ini merupakan menggunakan metode hash dalam mencari suatu kata. Teori ini jarang digunakan untuk mencari kata tunggal, namun cukup penting dan sangat efektif untukpencarian *pattern*.

Rabin-Karp mempresentasikan setiap karakter ke dalam bentuk desimal digit (digit radix-d)Σ = {0, 1, 2, 3, ..., d}, dimana d= |Σ|. Sehingga didapat masukkan *string k* berturut-turut sebagai perwakilan panjang *k* desimal. Karakter *string* 31415 sesuai dengan jumlah desimal 31,415. Kemudian pola *p* dihash menjadi nilai desimal dan *string* di presentasikan dengan penjumlahan angka dengan 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 menggunakan operasi mod, setelah dikenai operasi mod q, nilainya akan menjadi lebih kecil dari q. Tetapi tidak semua nilai hash yang cocok berarti polanya cocok. Hal ini sering terjadi pada beberapa kasus, ini disebut spurious hits. Kemungkinan terjadinya diantaranya karena:

Operasi*mod*terinterfensiolehkeunikannilai*hash*(nilai *mod q* biasanya dipilih bilangan prima sehingga 10*q* hanya cocok dengan 1 katakomputer)

14 mod 13 =1
27 mod 13 =1

Informasi hilang setelahpenjumlahan

BAN → 1 + 0 + 13 =14
CAM → 2 + 0 + 12 =14

Sedangkan *pseudocode* Dan rumus matematis yang digunakan adalah sebagai berikut:

```
RABIN -KARP -MATCHER (T, P, d, q)
n = T.length
m = P.length
h = d^(m-1) mod q
p=0
t= 0
for i = 1 to m
// preprocessing
```

```
p = (dp + P [i]) mod q
t = (dt + T [i]) mod q
for s = 0 to n-m
// matching
        if p == t
                if P [1... m] == T [s + 1...s + m]
                        print “Pattern occurs with shift” s
        if s < n-m
                t  = (d(t-T[s + 1]h) + T [s + m + 1]) mod q
```

Rumus matematis:

H = c1 * bk – 1 + c2 * bk – 2 +c3 * bk – 3 + …+ ck * b0 --- (1)
Dimana :
c : nilai ASCII karakter
b : basis (bilangan prima)
k : banyak karakter
H : Hashing

2.3 Tahap Pencarian *Pattern*

Adapun tahap pencarian *pattern* menggunakan 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-gram.* Cara kerj*a* metode *k-gram* dengan mengambil potongan-potongan karakter huruf sejumlah *k* dari sebuah kata yang secara kontinyu dibaca dari teks sumber hingga akhir dari dokumen. contoh *k-gram* dengan *k* = 4:
   Teks : evenifnone Hasil 4-*gram* 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 *hashingRabin-Karp.*
   $Hash=Tibm-1+Tibm-2+\cdots+Ti+m-1\ mo$ ..(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, 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)

Adapun langkah-langkah algoritma *Rabin-Karp* yang dilakukan dalamperancangan sistem ini adalah sebagai berikut :

1. Menghilangkan tanda baca.
2. Membagi teks kedalam bentuk *k-gram*, dimana nilai *k* merupakan nilai parameteryang dipilih oleh pengguna.
3. Menghitung nilai *hash* dari setiap *k-gram*.
4. Memilih nilai *hash* yang sama.
5. Melakukan pengecekan *similitary*

3. HASIL DAN PEMBAHASAN

Aplikasi pencarian sinonim kata dirancang menggunakan algoritma pencarian yaitu algoritma *Rabin-Karp* sebagai solusi dalam melakukan pencarian terhadap *pattern* yang dijadikan sebagai acuan untuk melalukan pencarian karakter yang sesuai dengan *pattern* tersebut. Pada tahap ini dilakukan analisa terhadap algoritma *Rabin-Karp*. Denganmenentukan nilai *k-gram* dan basis bilangan primanya.Hasil dari hashing asli danhashing uji kemudian dibandingkan untuk dicari *hashing* yang sama. Jika hashingyang sama ditemukan, maka dari hasil hashing yang sama tersebut dihitung tingkatpersentase kesamaannya (*similarity*).

Pada aplikasi pencarian sinonim kata berbasis *android* dengan menerapkan algoritma *Rabin-Karp* yang akan dirancang oleh penulis akan menginputkan kata yang akan dicari kesinoniman katanya dan akan menghasilkan *Output* berupa list sinonim katanya dengan inputan yang di cari pada aplikasi pencarian sinonim kata berikut tabel 1, list sinonim kata:

Tabel 1. Kode – Kode ASCII

No	Kata	Sinonim Kata
1	Hiruk-Pikuk	Membatu runtuh, berisik, bising, gadung, gegap, hingar, kacau, mintuna, ramai, riuh

Untuk melakukan proses pada algoritma *Rabin-Karp* dengan mencocokan nilai *hash*, oleh sebab itu harus mengetahui terlebih dahulu nilai ASCII yang menjadi teks dan juga *pattern* pada pencocokan *string* yang akan dicari. Adapun kode teksyang akan digunakan adalah “HIRUK-PIKUK”, sehingga dapat dilihat kode ASCII pada tabel dibawah ini.

Journal of Informatics Management and Information Technology
ISSN 2774-4744 (Media Online)
Vol 2, No 1, Januari 2022
Hal 12−17
https://hostjournals.com/jimat

Tabel 2. Kode – Kode ASCII

No	Char	Kode ASCII
1	H	072
2	I	073
3	K	075
4	P	080
5	R	082
7	U	085

3.1 Penerapan Algoritma Rabin-Karp

Pada contoh kasus ini penulis akan melakukan pencarian sinonim kata, misalnya pencarian dengan teksHIRUK-PIKUK yang menggunakan kata "HIRUK-PIKUK" sebagai *pattern*-nya maka proses pencarian adalah sebagai berikut.
Contoh:
Text :**HIRUKPIKUK**
Pattern : **HIRUKPIKUK**

1. Langkah Pertama :
Teks yang digunakan pada contoh kasus ini adalah "HIRUK-PIKUK", pada langkah pertama menghapuskan spasi atau tanda baca.
Text : **HIRUKPIKUK**

2. Langkah Kedua:
Langkah selanjutnya adalah parsing *k-gram*, dimana pada proses ini katadipecah menjadi potongan-potongan dimana setiap potongan mengandung karaktersebanyak *k*. Berikut ini adalah contoh proses parsing *k-gramk*=7. Untuk pertama, membuat *k-gramk* =7 untuk teks seperti dibawah ini:

H	I	R	U	K	P	I			
	I	R	U	K	P	I	K		
		R	U	K	P	I	K	U	
			U	K	P	I	K	U	K

Setelah membuat *k-gram* untuk *pattern*, selanjutnya membuat *k-gramk* =7 untuk *pattern* seperti dibawah ini.

H	I	R	U	K	P	I			
	I	R	U	K	P	I	K		
		R	U	K	P	I	K	U	
			U	K	P	I	K	U	K
			U	K	P	I	K	U	K

3. Langkah ketiga :
Langkah selanjutnya menghitung nilai *hash* dari *k-gram* yang sudah dibagi dengan menggunakan rumus :

H = c1 * bk – 1 + c2 * bk – 2 +c3 * bk – 3 + …+ ck * b0

Dimana :
c : nilai ASCII karakter
b : basis (bilangan prima)
k : banyak karakter
H : Hashing

Untuk nilai b (basis bilangan prima) digunakan nilai basis bilangan prima adalah 3, sehingga dapat dihitunga untuk nilai *hash* setiap *k-gram* dan juga *pattern*.

A. Nilai *HashK-gram* Teks
Untuk nilai *hashk-gram* teks dapat dilihat pada dibawah ini :

1. K-gram 1 [HIRUKPI]

$$H = 72 * 3^{7-1}+73 * 3^{7-2}+82 * 3^{7-3}+85 * 3^{7-4}+75 * 3^{7-5}+80 * 3^{7-6}+73 * 3^{7-7}$$
$$=72 * 3^{6}+73 * 3^{5}+82 * 3^{4}+85 * 3^{3}+75 * 3^{2}+80 * 3^{1}+73 * 3^{0}$$
$$= 72 * 729+73 * 243+82* 81+85 * 27+75 * 9+80 * 3+73 * 1$$
$$= 52488+17739+6642+2295+675+240+73$$
$$= 80152$$

B. Nilai *Hash K-gramPattern*
Untuk nilai *hashk-gram pattern* dapat dilihat pada dibawah ini :

1. K-gram 1 [HIRUKPI]

$$H = 72 * 3^{7-1}+73 * 3^{7-2}+82 * 3^{7-3}+85 * 3^{7-4}+75 * 3^{7-5}+80 * 3^{7-6}+73 * 3^{7-7}$$
$$=72 * 3^{6}+73 * 3^{5}+82 * 3^{4}+85 * 3^{3}+75 * 3^{2}+80 * 3^{1}+73 * 3^{0}$$
$$= 72 * 729+73 * 243+82* 81+85 * 27+75 * 9+80 * 3+73 * 1$$
$$= 52488+17739+6642+2295+675+240+73$$
$$= 80152$$

4. Langkah keempat

Langkah selanjutnya adalah mencocokan nilai hash*k-grampattern* dengan nilai hash*k-gram* teks. Nilai hash*k-grampattern* memiliki kesamaan dengan nilai hash*k-gram*teks 1 sampai dengan nilai hash*k-gram*4[80152, 83067, 89635, 89646]. dengan [80152, 83067, 89635, 89646]. Kemudian mencocokan karakter *string* antara *pattern* dengan teks [HIRUK PIKUK] dengan [HIRUK PIKUK].

5. Langkah Kelima

Langkah terakhir adalah melakukan pengecekan *similitary* nilai hash*pattern* dengan nilai hash pada teks.

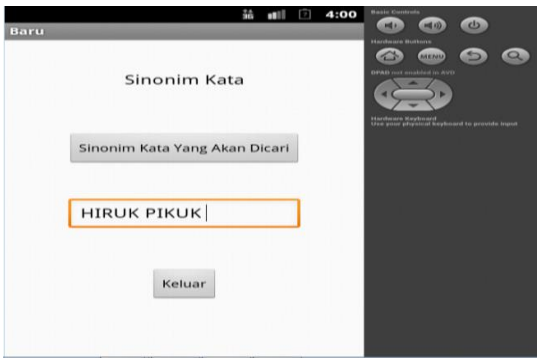
$$\text{Similitary } (Text,Pattern) = \frac{\sum Htext \cap \sum Hpattern}{\sum Htext \cup \sum Hpattern} \text{ x } 100\% \quad (1)$$

$$= \frac{\sum Htext \cap \sum Hpattern}{\sum Htext + \sum Hpattern - \sum Htext \cap \sum Hpattern} \text{ x } 100\%$$
$$= \frac{4}{4+4-4} \text{ x } 100\%$$
$$= \frac{4}{8-4} \text{ x } 100\%$$
$$= \frac{4}{4} \text{ x } 100\%$$
$$= 100\%$$

3.2 Implementasi Program

a. Tampilan Menu Pencarian

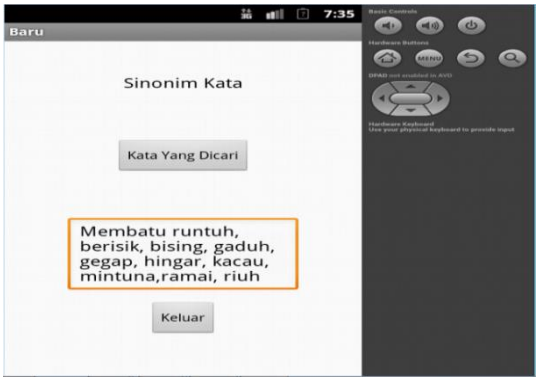
Menu pencarian merupakan halaman untuk user melakukan pencarian sinonim kata yang diinginkan dengan cara menginputkan kata yang ingin di cari kesinoniman katanya pada *Edittext* yang telah disediakan pada aplikasi. Terdapat dua objek pada halaman ini yaitu *Edittext* dan *Listview*, *Edittext* berfungsi sebagai tempat user menginputkan kata yang dicari sedangkan *Listview* berfungsi menampilkan isi yang ada di database. Screenshot halaman menu pencarian dapat dilihat pada gambar 2 dibawah.



Gambar 1. Tampilan Menu Pencarian

b. Tampilan Menu Hasil Pencarian

Menu hasil pencarian merupakan halaman hasil dari pencarian yang dilakukan, pada halaman ini akan memuat informasi yang dicari berupa kata yang di cari sinonim kata. Screenshot halaman menu hasil pencarian dapat dilihat pada gambar 2 dibawah.



Gambar 2. Tampilan Menu Hasil Pencarian

4. KESIMPULAN

Berdasarkan hasil penelitian dapat diambil kesimpulan aplikasi pencarian sinonim kata berbasis *android*hanya dapat menampilkan hasil pencarian yaitu sinonim kata berdasarkan kata yang tersimpan di dalam *database*. Proses pencarian yang dilakukan aplikasipencarian sinonim kata sangat membantu karena dapat melakukan pencarian string dengan cepat dan memberikan hasil yang tepat. Penggunaan algoritma *Rabin-Karp* dapat mempercepat proses pencarian sinonim katapada aplikasi sinonim kata berbasis *android* yang dibangun pada pada penelitian ini. Aplikasi pencarian sinonim kata ini dirancang berbasis *android* yang mana perancangannya dibangun menggunakan *Eclipse juno* dan *Sqlite*, tujuan dalam pembuatan aplikasi pencarian sinonim kata ini agar pemakai bahasa menguasai penggunaan sinonim kata yang benar dalam komunikasi sehari-hari.

REFERENCES

[1] Djajasudarma and T. Fatimah, Semantik 1 :Pengantar ke Arah Ilmu Makna, Bandung: Eresco.
[2] A. Ervana and A. Pertiwi, "Implementasi Algoritma Pencocokan String pada Aplikasi Pengarsipan Berbasis Web, " *Journal Informatika,* vol. 3, no. 2, pp. 1-14, 2012.
[3] D. A. Putra, H. Sujaini and H. S. Pretiwi, " Implementasi Algoritma Rabin-Karp untuk Membantu Pendeteksian Plagiat pada Karya Ilmiah," *Jurnal Sistem dan Teknologi Informasi (JUSTIN),* vol. 1, no. 1, pp. 1-9, 2015.
[4] A. Salam V.P. Wicaksana and K. Hastuti, "Sistem Rekomendasi Penentuan Dosen Pembimbing Tugas Akhir Dengan Menggunakan Algoritma Rabin- Karp, " *Techno COM,* Vol. 14, no. 3 pp 225-223, 2015.
[5] R. Sarno, Y. Anistyasari and R. Fitri, Semantic Search, Yogyakarta: Andi, 2012.
[6] Suyadi, "Pengaruh Program Sapta Pesona terhadap Peningkatan Pengunjung Obyek Wisata Guci Tegal,"*Jurnal Utilitas,* vol.1, no. 2, pp. 158-197, 2015.
[7] N. A. H. I. S. S. Firman Matondang, "PERANCANGAN APLIKASI TEXT EDITOR DENGAN MENERAPKAN ALGORITMA KNUTH-MORRIS-PRATT, "*Jurnal Riset Komputer" (JURIKOM)* vol. 3 no. 4 p. 16, 2016.
[8] A. Juansyah, "Pembangunan Aplikasi Ghild Trakerberbasis Asisted – Global Positioning System (A-GPS) dengan Platform Android, "*Jurnal Ilmiah Komputer dan Informatika (KOMPUTA),* vol. 1, no 1, pp. 1-8, 2015.
[9] S. S. M. & P. B. S. Hamza, "Sistem Koreksi Soal Essay Otomatis Dengan Menggunakan Metode Rabin Karp, "*Jurnal EECCIS,* vol. 7, no. 2, pp. 153-158, 2013.