SISTEM DETEKSI KEMIRIPAN JUDUL SKRIPSI PRODI TEKNIK INFORMATIKA MENGGUNAKAN ALGORITMA RABIN-KARP

SKRIPSI

Diajukan Untuk Memenuhi Salah Satu Syarat Guna
Memperoleh Gelar Sarjana Komputer (S.Kom)
Pada Progam Studi Teknik Informatika UN PGRI Kediri

OLEH :

MOH. FUAD UDDIN
NIM : 11.1.03.02.0233

FAKULTAS TEKNIK INFORMATIKA
UNIVERSITAS NUSANTARA PERSATUAN GURU REPUBLIK INDONESIA
KEDIRI
2016

Skripsi oleh :

MOH. FUAD UDDIN

NIM : 11.1.03.02.0233

Judul :

SISTEM DETEKSI KEMIRIPAN JUDUL SKRIPSI PRODI TEKNIK INFORMATIKA MENGGUNAKAN ALGORITMA RABIN-KARP

Telah disetujui untuk diajukan Kepada
Panitia Ujian/Sidang Skripsi Progam Studi
Teknik Informatika UN PGRI Kediri

Tanggal : 12 Januari 2016

Pembimbing I	Pembimbing II
Dr. M. Muchson, S.E, M.M	**Ardi Sanjaya, M.Kom**
NIDN. 0018126701	**NIDN. 0706118101**

ii

Skripsi oleh :

MOH. FUAD UDDIN
NPM: 11.1.03.02.0233

Judul:

SISTEM DETEKSI KEMIRIPAN JUDUL SKRIPSI PRODI TEKNIK INFORMATIKA MENGGUNAKAN ALGORITMA RABIN-KARP

Telah dipertahankan di depan Panitia Ujian/Sidang Skripsi
Progam Studi Teknik Informatika UN PGRI Kediri
Pada tanggal : 12 Januari 2016

Dan Dinyatakan telah memenuhi persyaratan

Panitia Penguji:

1. Ketua : Dr. M. Muchson, S.E, M.M _____
2. Penguji I : Dr. Suryo Widodo,M.Pd _____
3. Penguji II : Ardi Sanjaya, M.Kom _____

Mengetahui,
Dekan Fak. Teknik
Universitas Nusantara PGRI Kediri

Dr. Suryo Widodo, M.Pd
NIP. 19640202 1991103 1 002

SISTEM DETEKSI KEMIRIPAN JUDUL SKRIPSI PRODI TEKNIK INFORMATIKA MENGGUNAKAN ALGORITMA RABIN-KARP

Moh. Fuad Uddin
11.1.03.02.0233
Fakultas Teknik - Prodi Teknik Informatika
san3fu@gmail.com
Dr. M. Muchson, SE, M.M dan Ardi Sanjaya, M.Kom
UNIVERSITAS NUSANTARA PGRI KEDIRI

ABSTRAK

Kemiripan judul skripsi tidak menutup kemungkinan isi dari skripsi tersebut sama kususnya pada prodi Teknik Informatika. Untuk mengantisipasi hal tersebut diperlukan suatu sistem deteksi kemiripan judul skripsi.

Untuk melakukan deteksi kemiripan judul skripsi (data teks) pada intinya adalah dengan melakukan pencocokan string / terms. Algoritma yang digunakan dalam skripsi ini adalah Rabin-Karp. Algoritma Rabin-Karp ini sangat cocok digunakan untuk pola pencarian jamak (multiple pattern search). Algoritma ini tidak melakukan pergesaran yang rumit dalam menyelesaikan masalah, karena algoritma ini dapat mempercepat pengecekan kata pada suatu teks dengan menggunakan fungsi hash.

Fungsi hash adalah fungsi yang mengkonveresikan suatu kata menjadi nilai yang disebut nilai hash (hashvalue). Ide dasarnya adalah menghitung posisi record yang dicari dalam larik, bukan membandingkan record dengan isi pada larik.

Kata Kunci

Algoritma Rabin-Karp, *hashing, string-matching, patterm, K-gram.*

I. LATAR BELAKANG

Fenomena plagiarisme yang lebih spesifik sering terjadi adalah pada dunia akademis. Hal ini dikarenakan kegiatan tulis menulis sering dilakukan oleh mahasiswa untuk menyelesaikan tugas kuliah, terlebih lagi pada judul skripsi yang akan diajukan. Kebanyakan mahasiswa masih kebingungan dalam menentukan judul yang hendak ingin diajukan dan tidak menutup kemungkinan akan mencari reverensi judul-judul terdahulu yang sudah pernah diajukan di perpustakaan dan terlebih lagi di internet.

Dengan adanya judul yang sama, tidak menutup kemungkinan isi dari judul skripsi tersebut juga sama, sehingga tindak plagiarisme tersebut akan terwujud dalam hal ini, meski tidak semuanya.

Pada sistem ini membandingkan antara kemiripan judul yang benar-benar dibuat oleh mahasiswa dengan judul yang sudah pernah diajukan mahasiswa terdahulu. Dengan mengetahui persentase kemiripan kedua judul tersebut dapat dijadikan bahan pertimbangan apakah judul yang diajukan oleh mahasiswa tersebut menjiplak judul seseorang atau tidak.

Algoritma yang digunakan adalah Rabin-Karp Algorithm. Algoritma ini digunakan karena sangat efektif untuk pencarian lebih dari satu kata (multi pattern) (Karp-Rabin,1987).

Di dalam algoritma Rabin-Karp menggunakan fungsi hashing. Fungsi hashing menyediakan metode sederhana untuk menghindari perbandingan jumlah karakter yang quadratic di dalam banyak kasus atau situasi.

II. METODE

Metode Algoritma Rabin-Karp

Algoritma Karp-Rabin diciptakan oleh Michael O.Rabin dan Richard M. Karp pada tahun 1978 dengan menggunakan fungsi hashig untuk menemukan pattern di dalam string teks (Fernando, 2009).

Karakteristik Algoritma Karp-Rabin :

1) Menggunakan sebuah fungsi hashing
2) Fase preprocessing menggunakan kompleksitas waktu O(m)
3) Untuk fase pencarian kompleksitasnya : O(mn)
4) Waktu yang diperlukan O(n+m)

Fungsi hashing menyediakan metode sederhana untuk menghindari perbandingan jumlah karekter yang quadratic di dalam banyak kasus dan situasi. Daripada melakukan pemeriksaan terhadap setiap posisi dari teks ketika untuk melakukan pemeriksaan hanya jika teks yang sedang kita proses memilik kemiripan seperti pada pattern. Untuk melakukan pengecekan kemiripan anatar dua kata ini digunakan fungsi hash. (Fernando, 2009).

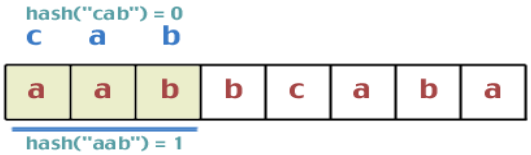
Algoritma Rabin-Karp ini paling banyak digunakan dalam pendeteksian pencontekan atau kecurangan. Contohnya pada makalah, paper dan lain sebagainya.

Bila diberikan source matrial atau dokumennya, algoritma ini dapat dengan cepat mencari seluruh paper dari setiap kalimat, mengabaikan lowercase atau uppercase, tanda titik, tanda seru, tanda Tanya serta tanda baca lainnya.

Konsep Algoritma Rabin-Karp

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 (Firdaus, 2008).

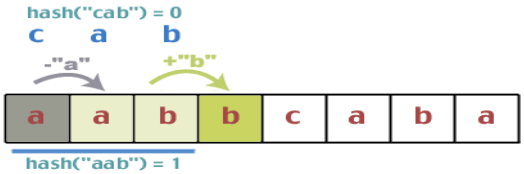
Berikut ini adalah ilustrasi dari konsep algoritma Rabin-Karp. Diberikan masukan "cab" dan teks "aabbcaba". Fungsi *hash* yang dipakai misalnya akan menambahkan nilai keterurutan setiap huruf dalam alphabet (a = 1, b = 2, dst) dan melakukan modulo dengan 3. Didapatkan nilai *hash* dari "cab" adalah 0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



(Firdaus, 2008)

Gambar 2.1 *Fingerprint* awal

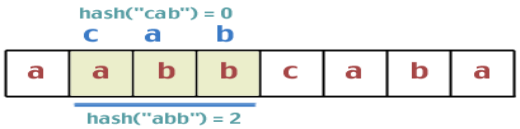
Hasil perbandingan ternyata tidak sama, maka substring pada teks akan bergeser satu karakter ke kanan. Algoritma tidak menghitung kembali nilai *hash* substring. Disinilah dilakukan apa yang disebut *rooling hash* yaitu mengurangi nilai karakter yang keluar dan menambahkan nilai karakter yang masuk sehingga didapatkan kompleksitas waktu yang relative konstan pada setiap kali pergeseran.



(Firdaus, 2008)

Gambar 2.2 Menggeser *fingerprint*

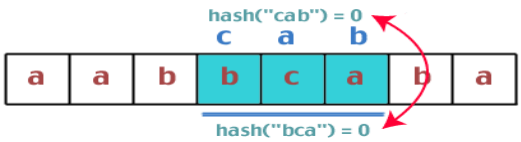
Setelah pergeseran, didapatkan nilai *hash* dari *fingerprint* "abb" (abb = aab – a + b) menjadi dua (2 = 1 – 1 + 2).



(Firdaus, 2008)

Gambar 2.3 Pembandingan kedua

Hasil perbandingan juga tidak sama, maka dilakukan pergeseran. Begitu pula dengan perbandingan ketiga. Pada perbandingan keempat, didapatkan nilai *hash* yang sama.

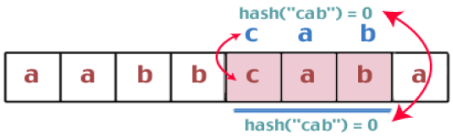


(Firdaus, 2008)

Gambar 2.4 Perbandingan keempat (nilai *hash* sama)

Karena nilai *hash* sama, maka dilakukan perbandingan string karakter per karakter

antara "bca" dan "cab". Didapatkan hasil bahwa kedua string tidak sama. Kembali substring bergeser ke kanan.



(Firdaus, 2008)

Gambar 2.5 Perbandingan kelima (srting ditemukan)

Pada perbandingan yang kelima, kedua nilai *hash* dan karakter pembentuk string sesuai, sehingga solusi ditemukan. Dari hasil perhitungan, kompleksitas waktu yang dibutuhkan adalah O(m+n) dengan m adalah paling panjang string masukan dan n adalah jumlah *looping* yang dilakukan untuk menemukan solusi. Hasil ini jauh lebih mangkus daripada kompleksitas waktu yang didapat menggunakan algoritma *brute-force* yaitu O(mn) (Firdaus, 2008).

Pencocokan String/Kata

String Matching atau pencocokan *string* adalah suatu metode yang digunakan untuk menemukan suatu keakuratan/hasil dari satu atau beberapa pola teks yang diberikan. *String Matching* marupakan pokok bahasan yang penting dalam ilmu computer karena teks merupakan bentuk utama dari pertukaran informasi antar manusia, misalkan pada literature, karya ilmiah, halaman web dsb (Eko Nugroho, 2011).

Srting Matching focus pada pencarian satu, atau lebih umum, semua kehadiran sebuah kata (lebih umum disebut *pattern*) dalam sebuah teks. Semua algoritma yang akan dibahas mengeluarkan semua kehadiran pola teks. Pola dinotasikan sebagai x = x[0….m-1]; m adalah panjangnnya.

Teks dinotasikan sebagai y = y[0….n-1]; n adalah panjannya. Kedua string dibentuk dari set karakter yang disebut alphabet dinotasikan ∑ dengan ukuran σ (Atmopawiro,2006).

K-*gram*

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

1) *Text*

   A do run run run, a do run run

2) Kemudian dilakukan penghilangan spasi.

   Adorunrunrunadorunrun

3) Sehingga dihasilkan rangkaian 5-*grams* yang diturunkan dari *text* :

   Adoru dorun orunr runru unrun nrunr runru unrun nruna runad unado nador adoru dorun orunr runru unrun (Eko Nugroho, 2011).

Hashing

Hashing adalah transformasi aritmatika sebuah string dari karakter menjadi nilai yang merepresentasikan string aslinya. Menurut bahasanya, *hash* berarti memenggal dan kemudian menggabungkan. *Hashing* digunakan sebagai metode untuk menyimpan data dalam sebuah larik (*array*) agar penyimpanan data, pencarian data, penambahan data, dan penghapusan data dapat dilakukan dengan cepat. Ide dasarnya adalah menghitung posisi *record* yang dicari dalam larik. Fungsi yang mengembalikan nilai atau kunci disebut fungsi *hash* dan larik yang digunakan disebut tabel *hash.* Secara teori, kompleksitas waktu (T(n)) dari fungsi *hash* yang ideal adalah O(1). Untuk mencapai itu setiap *record* membutuhkan suatu kunci yang unik (Rizal Isnanto dkk, 2009).

Fungsi *Hashing*

Fungsi *Hash* (dilambangkan dengan *h(k)*) bertugas untuk mengubah k (*key*) menjadi suatu nilai dalam interval [0…..X], dimana "X" adalah jumlah maksimum dari *record-record* yang dapat ditampung dalam table. Jumlah maksimum ini bergantung pada ruang memori yang tersedia. Fungsi *hash* yang ideal adalah mudah dihitung dan bersifat acak, agar dapat menyebarkan semua *key*. Dengan *key* yang tersebar, berarti data dapat terdistribusi secara seragam tabrakan dapat dicegah. Sehingga kompleksitas waktu model *hash* dapat mencapai O(1), di mana komplesitas tersebut tidak ditemukan pada struktur model lain. (Rizal Isnanto dkk, 2009)

Contoh sederhana *hashing* adalah :

Firdaus, Hari	7864 : Firdaus, Hari
Rabin, Michael	1990:Rabin, Michael
Munir, Rinaldi	9802: Munir, Rinaldi
Karp, Richard	8822 : Karp, Richard

Contoh diatas adalah penggunaan *hashing* dalam pencarian pada *database.* Apabila tidak di-*hash*, pencarian akan dilakukan karakter per karakter pada nama-nama yang panjangnnya bervariasi dan ada 26 kemungkinan pada setiap karakter. Namun pencarian akan menjadi lebih mangkus setelah di-*hash* karena hanya akan membandingkan empat digit angka dengan Cuma 10 kemungkinan setiap angka (Firdaus, 2008).

Pengukuran Nilai *Similarity*

Inti dari pendekatan K-*grams* dibagi menjadi dua tahap. Pada tahap pertama, membagi kata menjadi K-*grams.* Sedangkan pada tahap 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 (Eko Nugroho, 2011).

Untuk menghitung nilai similaritas, digunakan hitungan sebagai berikut :

$$S = \frac{2C}{A+B}$$

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 kedua teks yang dibandingkan.

Contoh perhitungan nilai *similarity* 3 kata dengan K = 2 (bi-*grams*)

Tabel Perhitungan Nilai *Similarity* 3 kata dengan K = 2 (Eko Nugroho, 2011).

Kata yang dibandingakan (*)	*K-grams yang sama*	*Similarity*
Photography (9) dan Photographic (10)	*Ph ho ot to gr ra ap ph = 8*	*2*8/(9+10)=0.84*
Photography (9) dan Phonetic (7)	*Ph ho = 2*	*2*2/(9+7) = 0.25*
Photographic (10) dan Phonetic (7)	*Ph ho ic = 3*	*2*3/(10+7)=0.35*

Persentase Nilai *Similarity*

Untuk menentukan jenis plagiarism antara dokumen yang diuji ada 5 jenis penilaian persentase *similirarity* (A. Benny dan Sinta, 2008).

a) 0% : Hasil uji 0% berarti kedua dokumen tersebut benar-benar berbeda baik dari segi isi dan kalimat secara keseluruhan.
b) < 15% : Hasil uji 15% berarti kedua dokumen tersebut hanya mempunyai sedikit kesamaan.
c) 15-50% : Hasil uji 15-50% berarti menandakan dokumen tersebut termasuk plagiat tingkat sedang.
d) > 50% : Hasil uji lebih besar 50% berarti dapat dikatakan bahwa dokumen tersebut mendekati plagiarisme.
e) 100% : Hasil uji 100% menandakan bahwa dokumen tersebut adalah plagiat karena dari awal sampai akhir mempunyai isi yang sama persis.

III. HASIL DAN KESIMPULAN

HASIL

Tampilan Halaman Utama

Index halaman pada browser akan menampilkan halaman utama atau disebut juga dengan halaman beranda.

Tampilan Halaman Uji Judul

Setelah masuk pada halaman utama, mahasiswa langsung dapat menguji judul yang hendak akan diajukan, yaitu dengan masuk pada menu "Uji Judul" dan mengetikkan judul skripsi pada kotak yang telah disediakan, kemudia meng-klik "Generate"

Tampilan Halaman Judul

Pada menu "Judul" Admin dapat mengelola data judul yang sudah disimpan dalamam data base, baik itu menambah judul skripsi dalam data base dengan klik "Tambah Judul", atau "edit" dan "hapus" judul yang sudah ada dalam data base.

Hasil Perjobaan Sistem

Hasil uji coba penerapan Algoritma Rabin-Karp dalam menghitung persentase kesamaan dua kata yakni "Rabin Karp Algoritma" dengan "Algoritma Rabin Karp" dengan tingkat k-*gram* yang berbeda dapat dilihat pada table berikut ini :

No	Kata	*kGram*	*Similarity*
1	Rabin Karp Algoritma terhadap Algoritma Rabin Karp	1	100,00%
2		2	89,47%
3		3	77,78%
4		4	64,71%
5		5	50,00%

KESIMPULAN

Setelah melakukan analisis, perancangan, implementasi dan pengujian maka dapat diperoleh kesimpulan bahwa aplikasi berbasis web ini dapat diterapkan dalam memperoleh hasil prosentase kemiripan judul skripsi yang diajukan oleh mahasiswa dengan data judul skripsi yang sudah ada dalam data base dan sudah layak untuk digunakan, yang dibuktikan dengan hasil uji coba luas dengan rata-rata prosentase 88%.

IV. DAFTAR PUSTAKA

[1] Atmopawiro, Alsasian. 2006. *Pengkajian Dan Analisis Tiga Algoritma Efisien Rabin-Karp, Knut-Morris-Pratt, Dan Boyer-Moore Dalam Pencarian Pola Dalam Suatu Teks*. Progam Studi Teknik Informatika, Institut Teknologi Bandung.

[2] Fernando, Hary. 2009. *Perbandingan dan Pengujian Beberapa Algoritma Rabin-Karp*. Progam Studi Teknik

Informatika, Institut Teknologi Bandung (ITB). Bandung.

[3] Firdaus, Hari Bagus. 2008. *Deteksi Dokumen Menggunakan Algoritma Rabin-Karp.* Progam Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung (ITB). Bandung.

[4] Isnanto, R., Rizal, Fatchurrohim, Adian, Gunawan, Nardho. 2009. *Implementasi Metode Hash (Hashing) Dalam Pencarian Data Pada Kamus Kebidanan.* Universitas Diponegoro, Semarang, Indonesia.

[5] Karp, Richart M., Rabin, Michael O. 1987. *Efficient Randomzed Pattern-Matching Algoritms.*

[6] Nugroho, Eko. 2011. *Perancangan Sistem Deteksi Plagirisme Dokumen Teks Dengan Menggunakan Algoritma Rabin-Karp.* Fakultas Matematika Dan Ilmu Pengetahuan Alam. Universitas Brawijaya. Malang, Indonesia.

[7] Stein, B., Meyer, S. zu Eissen. 2006. *Near Similarity Search and Plagiarism Analysis*, 29th Annual Conference of the German Classification Society (GfKI), Magdeburg, ISDN 1431-8814,pp. 430 – 437.