JATIKOM: Jurnal Teori dan Aplikasi Ilmu Komputer
Vol. 1, No.2; September 2018, hal. 91-95

Aplikasi Deteksi Plagiarisme Dokumen Skripsi dengan Algoritma Rabin-Karp

Teti Suryati[*], Yudi Wibisono[#], Yaya Wihardi[#]

Departemen Pendidikan Ilmu Komputer, Universitas Pendidikan Indonesia
Bandung, Indonesia

[1] teti.suryati@student.upi.edu

[2,3] {yudi, yaya.wihardi}@upi.edu

Abstrak — Saat ini plagiarisme menjadi masalah penting di lingkungan akademik. Kemajuan teknologi memudahkan akses dokumen skripsi dari *repository* milik universitas yang dapat memicu tindakan plagiarisme. Plagiarisme merupakan salah satu tindak kejahatan yang dapat ditindak pidana. Diperlukan deteksi plagiarisme terhadap dokumen skripsi yang disubmit mahasiswa. Penelitian ini menerapkan teknik *string searching* yaitu algoritma Rabin-Karp dalam mendeteksi plagiarisme dokumen skripsi. Prinsip dasar algoritma Rabin-Karp adalah fungsi *hashing*. Sebelum masuk ke fungsi *hashing*, dokumen yang berisi *string* akan di *parsing* sehingga membentuk kelompok kata, setelah itu kelompok kata tersebut akan di *generate* menjadi bentuk bilangan bulat (*hashing*). Setelah membentuk nilai *hash* maka nilai *hash* kedua dokumen akan dibandingkan sehingga mendapat hasil *similarity*. Untuk menguji sistem ini dilakukan eksperimen yang melibatkan 54 dokumen hasil plagiat dengan pengujian 1 hingga 5 gram. Hasil dari penelitian ini adalah nilai error yang dihasillkan dengan gram 1 hingga 5 secara berturut-turut yaitu 10.61%, 14.1%, 18.5%, 22.68%, 25.93%.

Kata Kunci – Deteksi Plagiarisme; Algoritma Rabin-Karp.

I. PENDAHULUAN

Kemajuan teknologi saat ini memudahkan tindak kecurangan. Salah satu tindak kecurangan yang dilakukan yaitu plagiarisme. Plagiarisme adalah tindakan penjiplakan atau pengambilan karangan, pendapat, dan sebagainya dari orang lain dan menjadikannya seolah karangan dan pendapat sendiri [1].

UU. No. 20 Tahun 2003 dan Peraturan Menteri Pendidikan Nasional mengatur sanksi yang ditetapkan oleh pemerintah bagi mahasi

Diperlukan upaya pencegahan agar mahasiswa tidak melakukan plagiarisme dalam pengerjaan skripsi. Salah satu upaya pencegahan adalah dengan deteksi kemiripan dokumen, untuk mengecek dokumen skripsi mahasiswa apakah memiliki kemiripan dengan skripsi atau penelitian yang telah ada sebelumnya.

II. PENELITIAN TERKAIT

Rabin-Karp telah digunakan di dalam sistem koreksi soal essay[2] yang bertujuan untuk memudahkan penilaian dengan mengecek jawaban essay. Pada sistem koreksi soal essay tersebut algoritma Rabin-Karp berhasil di impelementasikan dengan perbedaan rata-rata nilai sistem dengan guru yaitu hanya 0.01 - 0.07 Pada penelitian tersebut, peneliti menyampaikan saran agar data uji dan data latih yang digunakan lebih variatif [2].

Selain itu terdapat pula penelitian pendeteksi plagiarsime dengan menggunakan algoritma Jaro-Winkler, dengan hasil akurasi yang didapatkan pada penelitian ini rata-rata sebesar 30.58% – 30.68% [3].

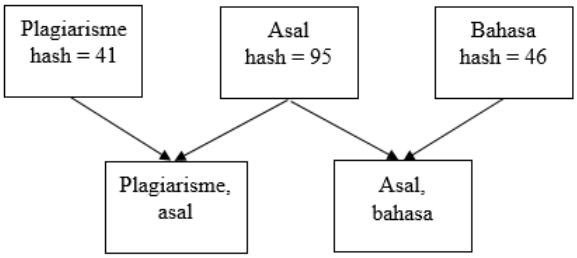
III. ALGORITMA RABIN-KARP

Algoritma Rabin-Karp merupakan algoritma pencarian string yang ditemukan oleh Michael Rabin dan Richard Karp. Algoritma Rabin-Karp merupakan algoritma pencarian string yang menggunakan fungsi hashing untuk membandingkan string yang dicari (m) dengan string yang dibandingkan (n) [4].

Fungsi hash adalah fungsi matematis yang digunakan untuk mengubah data menjadi bilangan bulat yang relatif kecil yang dapat berfungsi sebagai indeks pada array[5].

Fungsi hashing merupakan proses inti pada perhitungan algoritma Rabin-Karp. Hashing merupakan representasi ASCII yang menggantikan atau mentransformasikan karakter atau tanda baca menjadi sebuah nilai atau angka. Algoritma Rabin-Karp menggunakan fungsi rollinghash, yaitu perhitungan hash dengan menghitung *window* pertama, dan perhitungan *window* selanjutnya dengan mengurangi nilai hash kata yang dihilangkan dan menambah nilai hash kata yang ditambahkan. Misalnya terdapat tiga potong string dengan masing-masing hash seperti pada Gambar 1.

Dari tiga potong string pada gambar 1, akan dibentuk dua potongan *window* dengan 2-gram. Pada perhitungan rollinghash, *window* pertama akan dihitung terlebih dahulu dengan menggunakan persamaan (1).

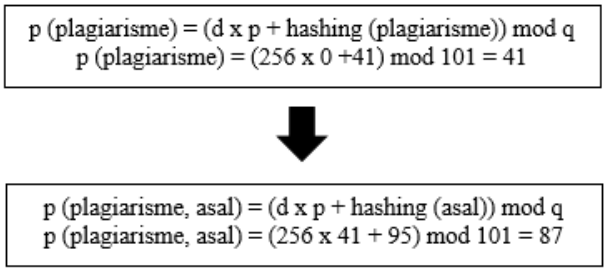


Gambar 1. Contoh window.

Nilai basis (d) yang digunakan pada penelitian ini adalah 256, karena merepresentasikan 256 karakter yang terdapat pada tabel ASCII. Nilai hash awal yang digunakan pada perhitungan adalah 0, nilai ASCII karakter (c) didapatkan dari tabel ASCII yang akan dilampirkan, serta nilai operand modulus (q) yang dipilih yaitu 101.

$$h = (d \times h + c)\%q \qquad (1)$$

Contoh perhitungan hash untuk window pertama ditunjukkan pada Gambar 2.



Gambar 2. Hashing window pertama.

Dari perhitungan *window* pertama, didapatkan nilai hash window pertama tersebut adalah 87. Selanjutnya akan dilakukan perhitungan *window* selanjutnya dengan fungsi rollinghash. Konsep perhitungan rollinghash yaitu dengan mengurangi hash window pertama dengan nilai hash dari kata yang dihilangkan, dan menambahkan nilai hash kata yang ditambahkan, seperti pada persamaan (2).

$$h = (d \times (h - T[i] \times m) + T[i + n])\%q \qquad (2)$$

h: nilai hash

d: basis

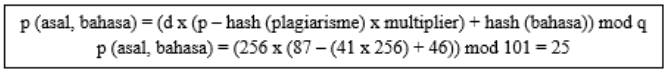
T [i]: nilai ASCII kata yang dihapus

T[i+n]: nilai ASCII kata yang ditambahkan

n: nilai ngram

q:operand modulus (bilangan prima yang cukup besar)

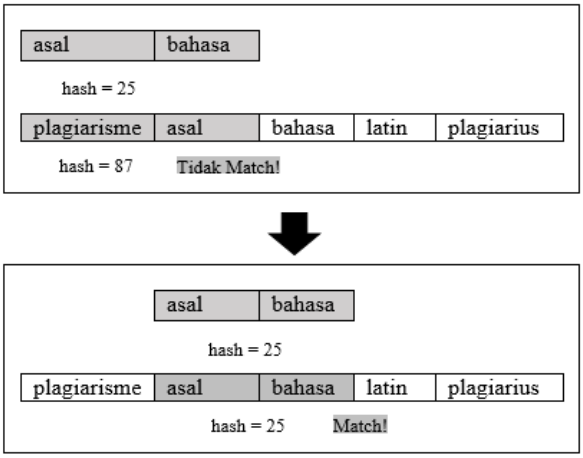
Berikut implementasi perhitungan hash window selanjutnya dengan menggunakan rollinghash.



Gambar 3. Perhitungan hash selanjutnya.

Dari perhitungan *window* selanjutnya, didapatkan nilai hashing dari kata "asal bahasa" adalah 25. Nilai hash tersebut akan digunakan untuk perbandingan kesamaan antara string satu dengan string lain. Perhitungan nilai hash akan terus berlanjut hingga window habis.

Setelah menghitung nilai hash, maka hash dari satu dokumen dengan dokumen lainnya akan dibandingkan. Jika nilai hash terdeteksi sama, maka akan dicek kembali pola string dari windownya, apakah memiliki kesamaan susunan string atau tidak, jika tidak sama maka dinamakan *spurious hit*, sedangkan jika memiliki susunan string yang sama maka dianggap mirip.



Gambar 4. Pencarian hash yang sama.

Pencarian akan berhenti setelah kata yang sama ditemukan. Persentase nilai kemiripan dihitung dengan membagi jumlah yang similar dengan jumlah *window* dari string pembanding seperti pada persamaan (3).

$$\frac{Jml.window\ yang\ similar}{Jml.window} \qquad (3)$$

Berikut contoh variabel yang digunakan untuk menentukan nilai persentase similarity yang dipaparkan pada Gambar 5.

String1 = asal bahasa
String2 = plagiarisme asal bahasa latin plagiarius
N-gram = 2
Jml. Window yang similar = 1 (asal bahasa)
Jml window = 4 ([pagiarisme asal], [asal bahasa], [bahasa latin], [latin plagiarius]

Gambar 5. Contoh variabel penentu persentase

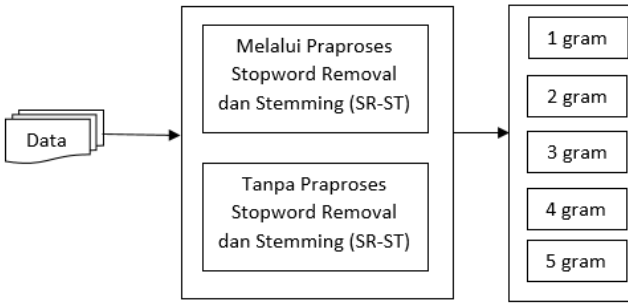
Sehingga dilakukan perhitungan persentase similarity seperti pada persamaan (4).

$$Similarity = \frac{1}{4} \times 100\% = 25\% \quad (4)$$

Maka didapat dari hasil perhitungan pada Gambar 5 bahwa string2 memiliki kesamaan sebesar 25% jika dibandingkan dengan string1.

IV. SKENARIO EKSPERIMEN

Eksperimen dilakukan dengan melakukan pendeteksian plagiarisme dalam beberapa skenario. Skenario eksperimen ditentukan dengan mengatur praproses yang digunakan dan besar gram yang digunakan.



Gambar 6. Skenario eksperimen.

Pada gambar, terdapat dua parameter dalam skenario yang digunakan yaitu praproses dan ngram. Berikut adalah penjelasan masing-masing parameter yang digunakan pada eksperimen.

1. Praproses

Praproses dilakukan pada empat tahap, yaitu *case folding*, *stopword removal* dan *stemming*. Untuk proses deteksi plagiarisme sebenarnya cukup dengan praproses *case folding*, sehingga praproses *stopword removal* dan *stemming* hanya menjadi praproses tambahan agar mencapai hasil yang optimal. Pada eksperimen ini akan dibandingkan skenario eksperimen dengan *praproses stopword removal* dan *stemming* (SR-ST) dan tanpa praproses tersebut.

2. Jumlah gram

Pada algoritma Rabin-Karp, gram adalah banyak kata yang diparsing dan akan membentuk kumpulan *window* yang akan dibandingkan similaritynya. Semakin besar gram maka semakin banyak kata yang akan dikelompokkan menjadi sebuah window. Semakin besar gram maka semakin banyak kata yang tercampur menjadi satu window, sehingga semakin sulit untuk mencari window yang similar jika plagiator banyak melakukan modifikasi pada teks yang ditirunya. Maka dari itu perlu ditentukan besar ngram yang digunakan untuk mendapatkan hasil yang optimal.

Berdasarkan parameter tersebut dilakukan beberapa skenario pada eksperimen ini agar dapat mengetahui pengaruh dari kombinasi parameter yang digunakan. Pada eksperimen ini besar ngram yang digunakan yaitu 1, 2, 3, 4 dan 5 ngram.

V. HASIL

Eksperimen dilakukan dengan menggunakan 54 buah dokumen hasil plagiat dari sembilan buah dokumen sumber. Sembilan dokumen tersebut di plagiat dengan lima teknik plagiat, yaitu teknik hapus kata, tambah kata, modifikasi kata, hapus modifikasi dan tambah modifikasi.

Teknik hapus kata yaitu teknik plagiat dengan menghilangkan sebagian kata dari dokumen aslinya. Teknik tambah kata yaitu teknik penambahan sebagian kata yang tidak terdapat pada dokumen aslinya, dan digabungkan dengan teks pada dokumen aslinya. Teknik modifikasi kata yaitu teknik plagiat dengan mengganti kata dengan kata sinonimnya dan atau mengganti lokasi kalimat yang terdapat pada dokumen aslinya. Sedangkan teknik hapus modifikasi yaitu penggabungan dari teknik hapus dan modifikasi kata. Begiu pula teknik tambah modifikasi yaitu teknik penggabungan teknik tambah kata dan modifikasi kata.

Sebelum melakukan proses pendeteksian kemiripan, data dibersihkan pada tahap praproses. Contoh hasil praproses dapat dilihat pada Tabel 1.

TABEL 1. CONTOH HASIL PRAPROSES

Teks awal	**Pelaku plagiat dapat dinamakan plagiator.**
Teks setelah praproses tanpa stopword-removal dan stemming (SR-ST)	pelaku, plagiat, dapat, dinamakan, plagiator
Teks setelah praproses dengan stopword-removal dan stemming (SR-ST)	pelaku, plagiat, nama, plagiator

dengan istilah plagiator.	dengan plagiator.

Setelah tahap praproses, dokumen akan diolah dengan algoritma Rabin-Karp. Tabel 1 menunjukkan hasil similarity yang dihasilkan dari eksperimen.

Tabel 2 menunjukkan bahwa untuk jumlah gram, nilai error terbesar terjadi pada 5-gram, dan nilai error terkecil pada saat 1-gram. Dapat disimpulkan bahwa dari hasil skenario ini, semakin besar ngram maka nilai error semakin tinggi.

Pada Tabel III ditunjukkan bahwa teknik plagiat dengan nilai error yang tinggi dihasilkan oleh teknik plagiat modifikasi dan kedua gabungan teknik modifikasi kata.

VI. PEMBAHASAN HASIL

Dilihat dari hasil eksperimen dari setiap skenario, dapat disimpulkan bahwa di setiap skenario semakin besar nilai gram, maka semakin tinggi nilai errornya, karena semakin besar kombinasi kata yang dibandingkan maka semakin sulit ditemukan kombinasi kata yang similar, meskipun kata yang dibandingkan memiliki kemiripan yang tinggi. Dapat dilihat kedua teks yang terdapat pada tabel 4.

TABEL 2. TABEL HASIL SKENARIO 1

Ngram	Tingkat Error Skenario tanpa SR-ST	Tingkat Error Skenario dengan SR-ST
1 gram	10.61%	10.79%
2 gram	14.10%	10.92%
3 gram	18.50%	17.39%
4 gram	22.68%	22.91%
5 gram	25.93%	27.85%

TABEL I. HASIL SKENARIO 2

Ngram	Tingkat Error HK	Tingkat Error TK	Tingkat Error MK	Tingkat Error HM	Tingkat Error TM
1 gram	11.84%	4.63%	9.19%	14.61%	14.68%
2 gram	6.53%	8.09%	18.38%	20.63%	17.95%
3 gram	8.75%	10.49%	24.92%	27.06%	22.68%
4 gram	10.92%	12.57%	31.01%	33.15%	27.37%
5 gram	12.81%	14.29%	35.67%	37.79%	31.06%

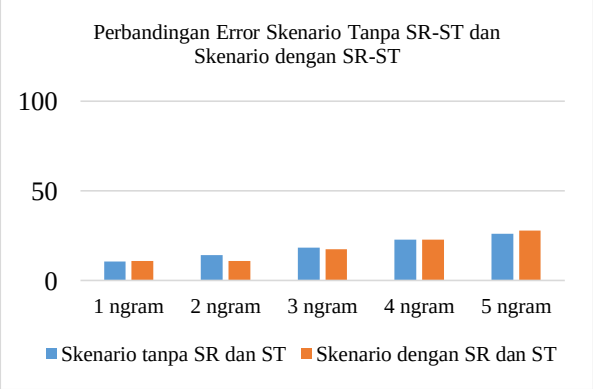
TABEL 4. CONTOH HASIL PARSING

Teks 1 (setelah di parsing 5-gram)			Teks 2 (setelah di parsing 5-gram)		
Pelaku	plagiat	dikenal	Pelaku	plagiat	dikenal

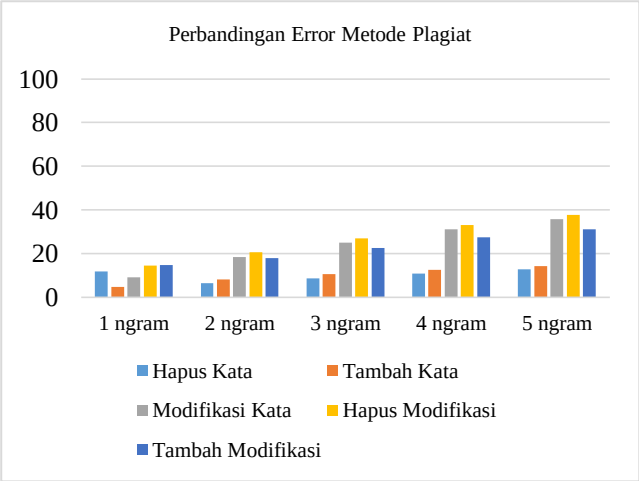
Misal teks 1 dan teks 2 dibandingkan dengan 5 ngram, maka didapatkan hasil bahwa kedua teks tersebut tidak mirip. Berbeda jika ngram yang digunakan 2 atau 3 ngram, maka teks tersebut dapat dianggap memiliki kesamaan atau similarity.

Selanjutnya untuk perbandingan hasil eksperimen dengan praproses *stopword removal* dan *stemming* dengan tanpa praproses tersebut ternyata tidak terlalu signifikan perbedaannya (Gambar 7).

Untuk 1-gram nilai error lebih tinggi dihasilkan oleh skenario dengan *stopword removal* (SR) dan *stemming* (ST) dengan error sebesar 10.79%. Pada 2-gram nilai error lebih tinggi dihasilkan oleh skenario tanpa SR-ST dengan error sebesar 14.1%. Untuk 3-gram nilai error lebih tinggi dihasilkan oleh skenario tanpa SR-ST dengan error sebesar 18.5%. Sedangkan untuk 4-gram nilai error lebih tinggi dihasilkan oleh skenario dengan SR-ST dengan error sebesar 22.91%. Serta untuk 5-gram nilai error lebih tinggi dihasilkan oleh skenario dengan SR-ST dengan error sebesar 27.85%. Selanjutnya pada gambar 8 digambarkan perbandingan nilai error antar kelima metode plagiat yang digunakan pada data eksperimen.



Gambar 7. Perbandingan nilai error skenario praproses.



Gambar 8. Perbandingan nilai error skenario teknik plagiat.

Dapat dilihat dari Gambar 8 hampir pada semua percobaan dengan variasi jumlah gram dihasilkan nilai error tertinggi dari hasil eksperimen dengan metode plagiat hapus modifikasi, sedangkan yang kedua adalah hasil metode modifikasi kata. Selanjutnya tertinggi ketiga yaitu diperoleh dari eksperimen dengan metode tambah modifikasi. Hal ini menunjukkan bahwa algoritma yang digunakan tidak dapat mendeteksi metode plagiat modifikasi kata, karena pada metode modifikasi ini dilakukan penggantian kata dengan sinonim atau kata yang berbeda namun sama makna, sehingga jika kata tersebut berbeda maka nilai hash pun berbeda sehingga tidak akan terdeteksi oleh algoritma. Selain itu, semakin besar jumlah gram maka kombinasi kata akan semakin sulit dideteksi kemiripannya, karena kombinasi kata tercampur dengan kata sinonim yang telah dimodifikasi, sehingga menghasilkan nilai error yang tinggi.

VII. KESIMPULAN

Algoritma Rabin-Karp dapat digunakan untuk menghitung kemiripan antara dokumen input dengan setiap dokumen pada *database*. Dengan melihat perbandingan hasil evaluasi dari berbagai skenario eksperimen yang telah dilakukan, dapat disimpulkan bahwa semakin besar gram yang digunakan maka semakin tinggi nilai error persentase kemiripannya. Dari hasil evaluasi juga dapat disimpulkan bahwa algoritma Rabin-Karp tidak dapat mengatasi plagiat dengan teknik modifikasi kata dengan kata sinonimnya, karena pada algoritma Rabin-Karp nilai hash akan berbeda jika kata tersebut dimodifikasi.

Perlu penelitian lanjutan untuk mengatasi plagiarisme yang menggunakan modifikasi kata dan sinonim dan dengan jumlah data yang lebih banyak.

REFERENSI

[1] S. Hamza, M. Sarosa and P. B. Santoso, "Sistem Koreksi Soal Essay Otomatis Dengan Menggunakan Metode Rabin-Karp," *Jurnal EECCIS Vol. 7, No.2*, p. 153, 2013.

[2] V. Stepchyshyn and R. S. Nelson, "Library Plagiarism Policies," *Association of College and Research Libraries*, p. 65, 2007.

[3] N. Singla and D. Garg, "String Matching Algorithms and their Applicability in Various Applications," *International Journal of Soft Computing and Engineering (IJSCE), Volume 1*, p. 218, 2012.

[4] D. Lemire and O. Kaser, "Recursive N-Gram Hashing is Pairwest Independent," *Computer Speech and Language*, pp. 698-710, 2010.

[5] A. F. Y. Kornain, "Penerapan Algoritma Jaro-Winkler Distance untuk Sistem Pendeteksi Plagiarisme pada Dokumen Teks Berbahasa Indonesia," 2014.

[6] T. Hoad and J. Zobel, "Methods for Identifying Versioned and Plagiarised Documents," *Journal of The American Society for Information Science and Technology*, pp. 203-215, 2003.