Analisis Perbandingan Algoritma *Rabin-Karp* dan *Ratcliff/Obershelp* untuk Menghitung Kesamaan Teks dalam Bahasa Indonesia

Bustami Yusuf[*1], Sari Vivianie[2], Jiwa Malem Marsya[3], Zuhra Sofyan[4]

[1*]Program Studi Teknologi Informasi, Fakultas Sains dan Teknologi
[2,3,4]Program Studi Pendidikan Teknologi Informasi, Fakultas Tarbiyah dan Keguruan,
UIN Ar-Raniry Banda Aceh
E-mail: *[1]bustamiyusoef@ar-raniry.ac.id, [2]sarivivianie.sv@gmail.com, [3]jiwa96@gmail.com,
[4]zuhra.sofyan@ar-raniry.ac.id

Abstrak

Penelitian ini dilatarbelakangi oleh suatu kegiatan yang marak terjadi yaitu plagiat atau lebih dikenal dengan istilah copy-paste. Copy-paste merupakan suatu tindakan yang sudah sangat sering terjadi didunia pendidikan, melakukan tindakan copy-paste yang berlebihan dapat mengakibatkan kurangnya sikap menghargai hak cipta atau karya orang lain. Oleh karena itu, maka perlu pencegahan untuk menghindarinya, salah satunya adalah dengan menggunakan aplikasi antiplagiarisme. Pembuatan aplikasi tersebut tentu tidak terlepas dari algoritma similarity dibelakangnya. Diantara algoritma-algoritma tersebut adalah yaitu Rabin-Karp dan Ratcliff/Obershelp. Pada penelitian ini akan diuji perbandingan kedua algoritma tersebut dalam menghitung kesamaan dokumen teks. Pengujian penelitian ini dilakukan dengan menggunakan sebanyak 50 halaman berita dari web tempo.co dan Dice's Similarity Coefficient digunakan sebagai alat ujinya. Hasil pengujian didapatkan bahwa algoritma Ratcliff/Obershelp mendapatkan nilai simmilarity yang lebih bagus sebesar ± 3% dibandingkan dengan algoritma Rabin-Karp. Selain itu, algoritma Ratcliff/Obershelp juga lebih unggul dalam kecepatan dan konsistensinya dari algoritma Rabin-Karp. Hal ini dikarenakan algoritma Rabin-Karp cenderung membutuhkan waktu yang banyak dalam tahapan prosesnya.

Kata Kunci- *Similarity, Rabin-Karp, Ratcliff/Obershelp, perbandingan , berita online*

Abstract

This research is motivated by the activities that are rife in the academic environment namely plagiarism or better known as copy-paste. Excessive copy-paste actions can result in a lack of respect for the copyright or work of others. Therefore, it is necessary to avoid prevention, one of which is to use antiplagiarism applications. Creating the application of antiplagiarism is inseparable from the similarity algorithms behind it. Among these algorithms are Rabin-Karp and Ratcliff/Obershelp. In this study a comparison of the two algorithms will be tested in calculating the similarity of text documents. Testing process in this study was carried out using 50 news pages from the tempo.co website and Dice's Similarity Coefficient was used as a similarity measurement. The results show that the Ratcliff/Obershelp algorithm gets a better similarity value of ± 3% compared to the Rabin-Karp algorithm. In addition, the Ratcliff/Obershelp algorithm is also superior in speed and consistency. This is because the Rabin-Karp algorithm tends to require a lot of time during process stages

Keywords- Similarity, Rabin-Karp, Ratcliff/Obershelp, comparison, online newspaper

1. PENDAHULUAN

Plagiarisme atau plagiat merupakan suatu tindakan yang mengambil karya orang lain. Tindakan plagiat ini sering terjadi, termasuk salah satunya di dunia pendidikan. Berdasarkan pengamatan yang dilakukan pada beberapa mahasiswa Universitas Islam Negeri (UIN) Ar-raniry didapatkan hasil bahwa banyak mahasiswa yang sengaja atau tanpa sengaja sering melakukan tindakan plagiat yang lebih dikenal dengan *Copy-paste*. Berdasarkan hasil penelitian yang dilakukan oleh Rio Satria pada tahun 2016 di Prodi Pendidikan Fisika Unsyiah, dari 134 skripsi mahasiswa didapatkan 80.55% plagiat penuh, 15.8% plagiat tapi acak, 3.3% plagiat mengutip serta ditambah pendapat sendiri dan 0.35% mengutip menggunakan kalimat sendiri[1].

Kegiatan plagiat yang terus terjadi di dunia pendidikan ini dapat mengakibatkan kurangnya sikap menghargai hak cipta atau karya orang lain. Bahkan perilaku plagiat dapat dikatakan sebagai perilaku yang harus dihindari dan dicegah serta diberikan sanksi secara objektif sesuai dengan tingkat kesalahan yang dilakukan. Dikarenakan hal tersebut, maka perlu dilakukan tahapan pecegahan terhadap tindakan plagiat. Pencegahan tersebut dapat dilakukan dengan menerapkan prinsip kejujuran pada diri sendiri, menumbuhkan sikap menjaga nama baik diri sendiri dan lembaga serta meningkatkan fungsi pengawasan, baik oleh sesama mahasiswa dan pembimbing maupun pimpinan perguruan tinggi (dosen dan peneliti)[2].

Pencegahan terhadap plagiat juga dapat dilakukan dengan mensosialisasikan penggunaan software antiplagiarisme. Software tersebut dapat digunakan untuk mendeteksi kesamaan suatu karya dengan karya yang lain. Salah satu software yang digunakan untuk mendeteksi plagiat tersebut adalah aplikasi Turnitin[3]. Selain aplikasi Turnitin, juga ada beberapa aplikasi lainnya yang dapat digunakan untuk mendeteksi plagiarisme, seperti viper anti-plagiarism scanner, plagtracker, writecheck dan aplikasi plagiarisme lainnya.

Pembuatan aplikasi tersebut tentu menggunakan algoritmanya masing-masing. Ada berbagai macam algoritma yang diterapkan dalam penelitian untuk membuat aplikasi pendeteksi plagiarisme. Diantaranya yaitu penelitian yang dilakukan oleh Yudhi Lady Joane,dkk dalam merancang bangun aplikasi deteksi kemiripan dokumen teks menggunakan algoritma *Ratclif/Obershelp*[4]. Dari penelitian tersebut didapatkan hasil bahwa dengan menggunakan algoritma *Ratcliff/Obershelp* sistem yang dibuat dapat melakukan proses pengidentifikasian kesamaan dokumen, banyaknya karakter didalam dokumen mempengaruhi waktu eksekusi dari proses pengidentifikasian kesamaan dokumen.

Penelitian lainnya juga dilakukan oleh Inta Widiastuti,dkk dalam membuat aplikasi pendeteksi kemiripan pada dokumen menggunakan algoritma *Rabin-Karp*[5]. Pada penelitian ini didapatkan hasil bahwa dengan menggunakan algoritma *Rabin-Karp* sistem pendeteksi kemiripan pada dokumen dapat berjalan dengan baik, semakin banyak kalimat yang berbeda pada suatu dokumen, hasil persentase yang dihasilkan akan semakin sedikit. Kedua algoritma yang dilakukan dalam penelitian diatas belum ditemukan penelitian yang membandingkan antara satu sama lainnya.

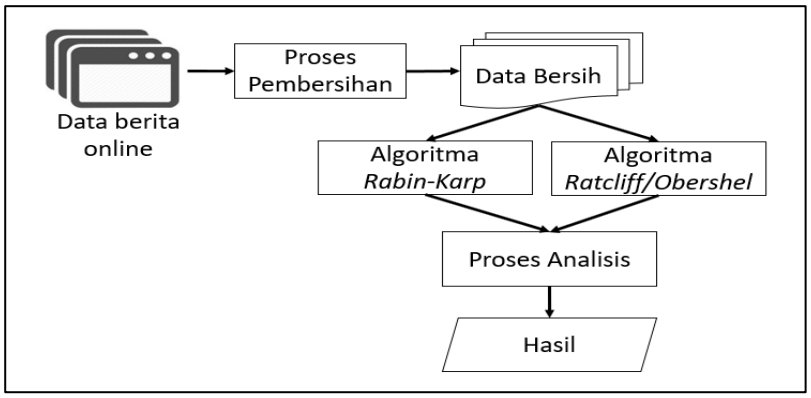
Oleh karena itu, pada penelitian ini akan dilakukan analisis perbandingan antara algoritma *Rabin-Karp* dan algoritma *Ratcliff/Obershelp* untuk menghitung persentase nilai *similarity* dokumen teks. Perbandingan yang ingin diteliti adalah kinerja dan hasil akhir dari kedua algoritma tersebut. Penelitian ini diharapkan dapat membantu peneliti untuk memilih algoritma yang lebih efesien untuk mendeteksi kemiripan kalimat pada suatu dokumen.

2. METODELOGI PENELITIAN

Prosedur yang dilakukan dalam penelitian ini adalah seperti yang terlihat dalam Gambar.1.

2.1 Dataset

Penelitian ini digunakan sebanyak 50 halaman berita dari portal berita online Tempo (https://www.tempo.co/). Halaman berita tersebut di-*crawling* secara acak tanpa dibatasi pada satu jenis berita.



Gambar 1. Alur penelitian

2.2 Proses Pembersiahan data (Cleaning Process)

Cleaning process adalah langkah untuk membersihkan halaman web berita dari karekter-karakter yang tidak dibutuhkan, seperti *HTML Tags* dan CSS Tags. Selain itu simbol yang tidak bermakna, seperti * *, *&#5;C*, *&#2*, dll, juga dihapus dalam proses ini. Selanjutnya, tanda baca seperti koma (,), titik (.), tanda tanya (?), dll juga dihilangkan sekaligus seluruh teks berita dijadikan *lowercase*. Hal ini dilakukan untuk menghilangkan kerancuan pada proses cek *similarity*.

2.3 Data yang digunakan

Setelah melalui proses cleaning terhadap 50 halaman berita, maka diperoleh data bersih sebanyak 6.090 kata dengan 46.233 huruf. Data bersih inilah yang akan digunakan sebagai data masukan terhadap setiap algoritma similarity *Robin-Karp* dan *Ratcliff/Obershel*.

Tabel 1. Pembagian Data Uji (DU)

Data Uji (DU)	Total jumlah kata (%)
DU	100
DU1	90
DU2	80
DU3	70
DU4	60
DU5	50
DU6	40
DU7	30
DU8	20
DU9	10

Sebagai acuan dalam menguji hasil similarity dari kedua algoritma yang digunakan, pada tahapan ini *dataset* diurai menjadi beberapa bentuk *subset*. Proses penguraian tersebut dilakukan dengan menghilangkan sebesar 10% isi dokumen berita secara bertahap. Misalnya, jumlah kata yang terdapat dalam salah satu halaman berita (yang selanjutnya kami sebut sebagai *Data-Uji* (DU)) adalah 50 kata, maka untuk menghitung tingkat kesamaannya, maka DU akan diuji dengan *Data-Uji-1*(DU1), dimana DU1 adalah DU yang dikurangi sebanyak 10% jumlah

katanya dari total kata yang dimiliki. Begitu juga untuk selanjutnya, dimana dalam penelitian ini kami menguji DU sampai dengan hanya menyisakan 10% (DU9) kata dari total kata. Untuk lebih jelas dapat di lihat dalam tabel 1.

2.4 Algoritma Robin-Karp

Algoritma Rabin-Karp dikembangkan oleh dua peneliti yaitu, Michael O.Rabin dan Richard M.Karp tahun 1987[6]. Algoritma ini menggunakan metode hash untuk mencari suatu kata, dengan melakukan tahapan membandingkan nilai hash dari string masukan dan substring pada kalimat[5]. Pada metode hash digunakan fungsi hash sebagai pembanding antara substring pada kalimat (n) dengan string yang dicari (m). Jika nilai hash yang didapatkan tidak sama, maka substring akan berpindah ke kanan. Perpindahan itu dilakukan sebanyak (n-m) kali. Namun, jika nilai hash yang didapatkan sama, maka akan dilakukan perbandingan sekali lagi terhadap karakter-karakternya. Nilai hash yang dihitung secara efisien dapat mempengaruhi kinerja dari algoritma ini[7].

Hashing merupakan suatu proses untuk mengubah string menjadi suatu nilai yang unik dengan panjang tertentu (fixed length). Proses ini digunakan untuk penanda sebuah string. Fungsi untuk mendapatkan nilai yang unik disebut fungsi hash, sedangkan nilai yang didapatkan disebut nilai hash. Pada umumnya, nilai hash digambarkan sebagai suatu string terpendek yang terdiri atas angka dan huruf yang terlihat acak yaitu data biner dalam bentuk heksadesimal, gambaran tersebut dikenal dengan istilah fingerprint[8]. Adapun tahapan yang dilakukan dalam algoritma Rabin-Karp adalah sebagai berikut[9]:

1. *Parsing k-gram*, membagi teks ke dalam *gram-gram*, teks akan dibagikan ke dalam *gram* yang ditentukan nilai *k* nya, sehingga didapatkan *substring*nya.
2. *Hashing*, mencari nilai *hash* dengan mengubah *substring k* ke dalam nilai *hash*.
3. Setelah didapatkan nilai *hash*, maka akan dicari nilai *hash* yang sama antar 2 teks dengan melakukan pencocokan *string* hasil *hashing*.
4. Menghitung persamaan 2 buah teks tersebut menggunakan persamaan *Dice's Similarity Coefficient*, Nilai kemiripan tersebut dapat dihitung dengan menggunakan rumus:

$$S = \frac{2\ x\ C}{|A| + |B|}\ x\ 100 \qquad (1)$$

Keterangan :
C = Jumlah *k-gram* unik dan memiliki struktur yang sama dari masing- masing kalimat
|A|+|B| = Jumlah *k-gram* yang unik dari teks pertama dan teks kedua.

Berdasarkan hasil penelitian Salmuasih[9], algoritma *Rabin-Karp* memiliki beberapa kelebihan yaitu, memiliki proses perhitungan yang relatif mudah dan dapat digunakan dalam kasus pencarian string dengan pola yang panjang. Sedangkan kekurangan Algortima *Rabin-Karp* adalah memiliki tahapan processing yang panjang sehingga membutuhkan waktu yang agak lama dan keakuratan pendeteksian algoritma ini sangat tergantung dari posisi kalimat.

2.5 Algoritma Ratcliff/Obershel

Algoritma *Ratcliff/Obershelp (RO)* merupakan algoritma yang menggunakan proses yang sama untuk memutuskan seberapa mirip dua pola satu dimensi, karena *String* teks merupakan satu dimensi, algoritma ini mengembalikan nilai yang dapat di gunakan sebagai faktor kepercayaan atau persentase, menunjukkan kesamaan dua *string*[10]. Adapun tahapan yang dilakukan dalam algoritma *Ratcliff/Obershelp* adalah sebagai berikut:

1. *Sequence(String) Matching*
   Pada tahapan ini dilakukan kegiatan mencocokan dua buah *string* untuk mendapatkan *subsequence(substring)* yang dimiliki oleh *String1* dan *String*2 dengan menghitung banyak karakter.
2. Pencarian *subsequence(substring)*
   Menghitung pencarian *subsequence* (*subtring* terpanjang dinamakan *anchor*) dan mencari *substring* lainnya dari kedua *string.*
3. Menghitung Kesamaan

Setelah semua tahapan dilakukan, dan *substring* sudah ditemukan, maka tahap selanjutnya adalah penilaian algoritma *Ratcliff/Obershelp* dilakukan dengan rumus perhitungannya:

$$S = \frac{2\ x\ Km}{|s1|+|s2|} x\ 100\% \qquad (2)$$

Keterangan:
Km = Jumlah Karakter yang sama
|S1| = Panjang dari String 1
|S2| = Panjang dari String 2

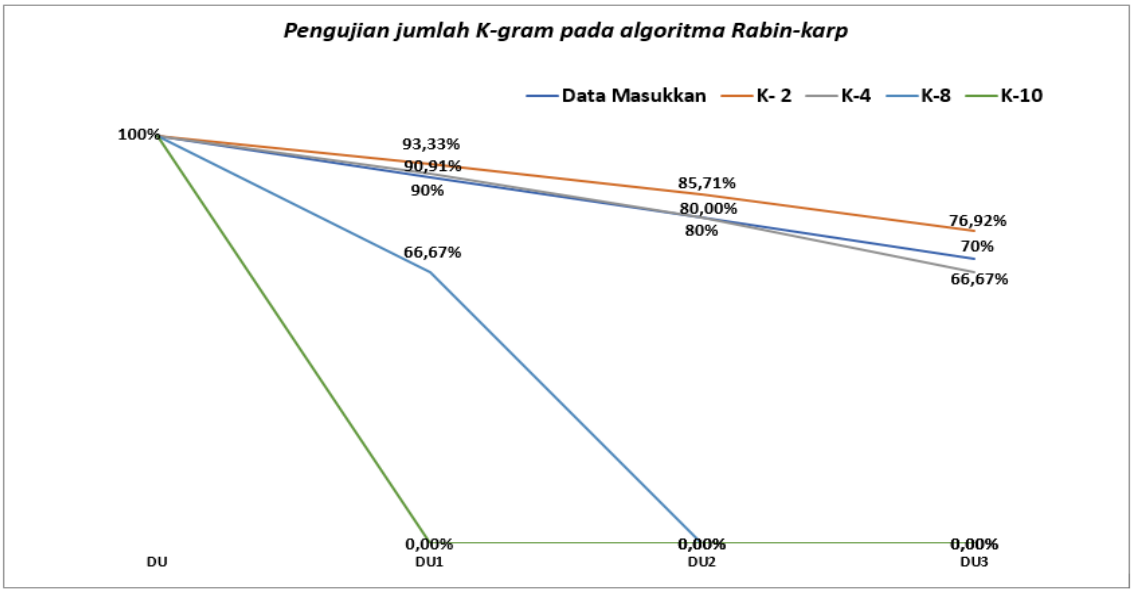
Algoritma *Ratcliff/Obershelp* memiliki kelebihan dan kekurangan yang hampir sama dengan dari algoritma *Rabin-Karp* seperti yang dikutip dari penelitian Joane dkk[4], yaitu; proses perhitungan yang lebih mudah, dapat digunakan dalam kasus pencarian string dengan pola yang lebih panjang dan memiliki tahapan *processing* yang lebih singkat. Sedangkan kekuranganya adalah Tidak bisa menentukan persamaan makna sinonim kata dan keakuratan deteksinya sangat dipengaruhi oleh posisi kalimat.

3. HASIL DAN PEMBAHASAN

3.1 Pengujian tingkat similarity

3.1.1 Penentuan k-gram pada algoritma Rabin-Karp.

Seperti yang sudah disebutkan pada *subbab* 2.4 bahwa langkah pertama pada algoritma *RK* adalah melakukan *parsing* terhadap *string* masukan kedalam bentuk *k-gram substring*. Penentuan jumlah *k* pada proses parsing ini sangat menentukan hasil uji similarity algoritma *Rabin-Karp*. Pada tahapan ini diuji dengan 5 jenis nilai *k* yaitu, 2, 4, 6, 8 dan 10. Hasil pengujian tersebut dapat dilihat pada gambar 2 berikut;

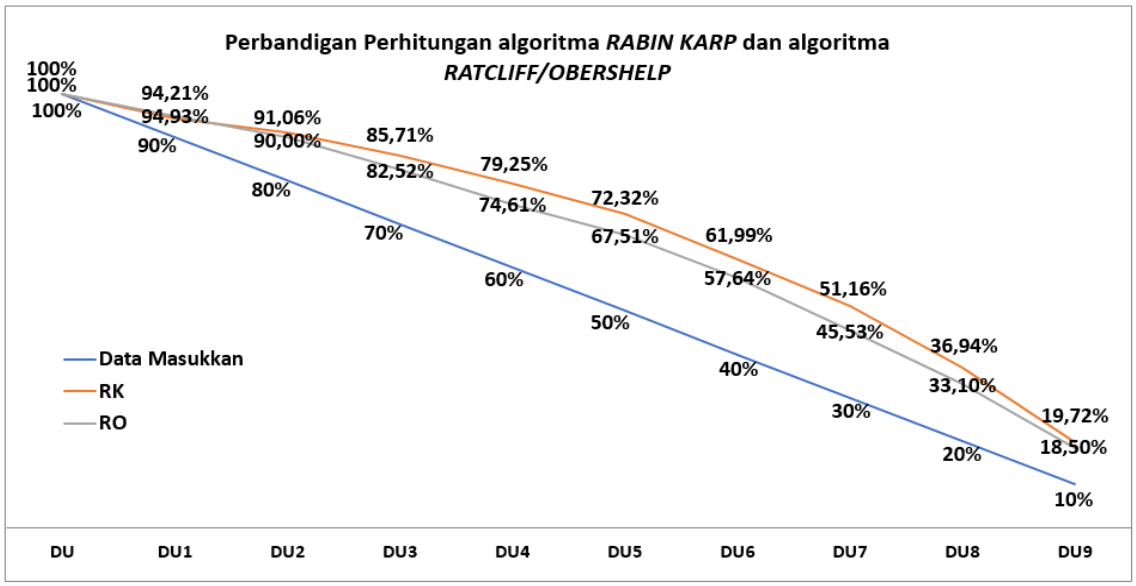


Gambar 2 Pengujian nilai *k* untuk algoritma *Rabin-Karp*

Berdasarkan gambar 2 diatas, terlihat bahwa hasil pengujian dengan menggunakan nilai *k*=4 menghasilkan nilai *similarity* yang sangat dekat pada setiap *Data-Uji* dibandingkan dengan nilai *k=2, k=6, k=8* atau *k*=10. Ketika nilai *k*=2 jelas terlihat nilai similarity-nya berada diatas *Data-Uji*, sedangkan ketika *k*=6, *k=8 dan k=10* nilai similarity jauh berada di bawah *Data-Uji*. Sehingga, dalam pada pengujian algoritma *RK* selanjutnya hanya menggunakan *k=4*.

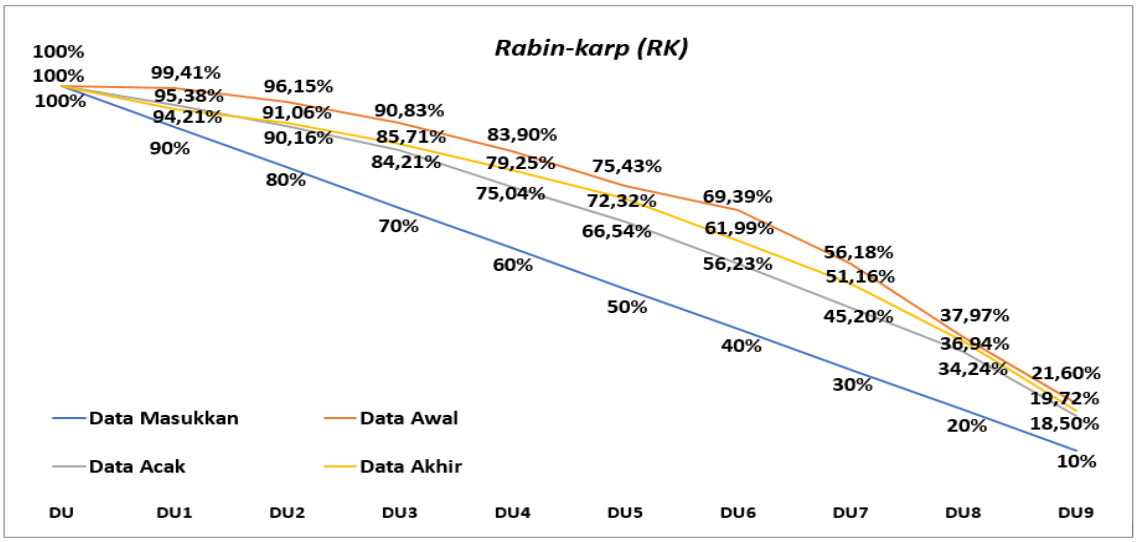
3.1.2 Perbandingan uji similarity algoritma Rabin-Karp (RK) dan Ratcliff/Obershelp (RO)

Gambar 3 menujukkan bahwa nilai *similarity* algoritma *RO* memiliki persentase yang lebih bagus dibandingkan dengan hasil dari algoritma *RK* pada hampir di seluruh *subset Data-Uji*. Dimana, nilai *similarity* algoritma *RO* lebih mendekati garis persentase *Data-Uji* dibandingkan dengan hasil pengujian dengan algoritma *RK*. Selisih nilai *similarity* rata-rata yang didapatkan dari dua algoritma tersebut adalah ± 3%.



Gambar 3 Grafik Perbandingan *RK* dan *RO*

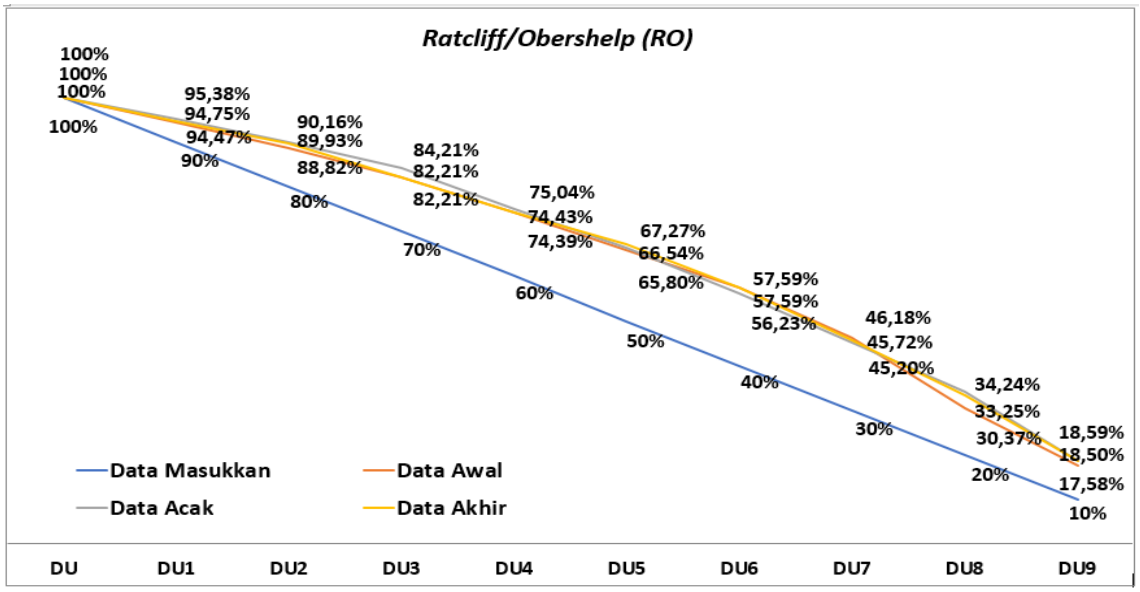
3.2 Pengujian tingkat perbedaan pemotongan dokumen teks



Gambar 4 Hasil perbedaan posisi penghapusan *Data-Uji* dengan *RK*

Seperti yang sudah dijelaskan dalam subbab 2.3, *Data-Uji* yang digunakan dalam penelitian ini diurai menjadi 10 Data-Uji yang berbeda dengan menghapus sebesar 10% isi dokumen secara berkala. Proses penghapusan tersebut menjadi topik yang menarik bagi peneliti untuk melihat adanya pengaruh tata letak kata terhadap nilai *similarity* dari kedua algoritma yang di uji. Pada pengujian kali ini, penghapusan isi dokumen dilakukan dengan tiga cara yang berbeda, yaitu pengahapusan di awal, akhir, dan secara acak. Penghapusan di awal adalah

penghapusan *Data-Uji* secara berkala yang dilakukan dari awal isi dokumen dan penghapusan di akhir adalah penghapusan *Data-Uji* secara berkala yang dilakukan dari bagian akhir isi dokumen. Sedangkan Penghapusan acak adalah penghapusan *Data-Uji* secara berkala yang dilakukan dari posisi acak isi dokumen. Adapun hasil yang didapat dari pengujian tersebut tersaji dalam Gambar 4 dan Gambar 5.



Gambar 5 Hasil perbedaan posisi penghapusan *Data-Uji* dengan *RO*

Dari Gambar 4, jelas terlihat bahwa hasil pengujian *similarity* dengan algoritma *RK* sangat dipengaruhi oleh posisi penghapusan data. Penghapusan data secara acak memiliki nilai *similarity* yang lebih tinggi dibandingkan dengan penghapusan pada posisi awal dan akhir dokumen. Sedangkan hasil pengujian dengan algoritma *RO* pada Gambar 5 sama sekali tidak dipengaruhi oleh posisi penghapusan berkala *Data-Uji*. Hal ini menunjukkan algoritma *RO* lebih stabil dalam menghitung nilai *similarity* suatu dokumen tanpa dipengaruhi oleh tata letak kesamaan kalimat atau kata dalam dokumen tersebut dibandingkan dengan algoritma *RK*.

3.3 Pengujian Konsistensi Algoritma

Tabel 4 Hasil perbandingan konsistensi algoritma *RK* dan *RO*

	RK	***RO***
Data-Uji	Nilai *Similarity* (%)	Nilai *Similarity* (%)
DU1	93; 94; 95; 96; 97	94; 95
DU2	87; 89; 90; 91; 92; 93	87; 88; 89
DU3	82; 84; 85; 86; 87; 88	80; 81; 82; 83; 84
DU4	76; 79; 80; 81; 82; 83	74; 75; 76; 77; 78
DU5	69; 70; 71; 72; 73; 75; 76	64; 65; 66; 67; 70
DU6	61; 62; 64; 65; 66; 69	55; 56; 57; 58; 60
DU7	50; 51; 52; 54; 55; 56; 57	44; 45; 46; 47
DU8	36; 37; 38; 40; 42; 43; 44; 45; 47	31; 32; 33; 34; 35
DU9	19; 20; 22; 24; 28; 32; 33	15; 17; 18; 20; 22

Pengujian untuk melihat konsistensi perbandingan antara algoritma *RK* dan *RO* dilakukan dengan menggunakan 10 data yang terdiri dari 50-200 kata dengan jumlah huruf yang berbeda.

Berdasarkan hasil pada tabel 4 didapatkan kesimpulan bahwa algoritma *RO* memiliki

hasil nilai *similarity* yang lebih konsisten dibandingkan dengan algoritma *Rabin-Karp*, hal ini dikarenakan:

a) Dari 10 data yang diuji sesuai dengan persentase data masukan, pada algoritma *RO* tidak memiliki banyak perbedaan pada setiap hasil yang didapatkan, sedangkan pada algoritma *RK*, memiliki banyak perbedaan atau lebih beranekaragam nilai yang didapatkan.
b) Pada hasil nilai *similarity* yang didapatkan, algoritma *RK*memiliki beberapa nilai *similarity* yang sama pada *Data-Uji* yang berbeda. Sedangkan pada algoritma *RO* setiap hasil yang didapatkan akan memiliki nilai yang berbeda.

3.4Pengujian Kecepatan Algoritma

Berdasarkan hasil pengujian waktu dalam Tabel 5 didapatkan bahwa waktu eksekusi dari algoritma *RK* cenderung lebih lama dari waktu eksekusi algoritma *RO*, hal ini dikarenakan algoritma *RK* memiliki tahapan proses yang lebih panjang.

Tabel 5 Perbandingan waktu eksekusi

Data Uji (DU)	*Waktu Eksekusi RK(millisecond)*	*Waktu Eksekusi RO(millisecond)*
DU4	2 ms	0.1 ms
DU3	3 ms	0.1 ms
DU2	4 ms	1 ms
DU1	5 ms	1 ms
DU	7 ms	2 ms

4. KESIMPULAN

Berdasarkan hasil pengujian yang telah dilakukan pada masing-masing algoritma yaitu algoritma *Rabin-Karp* dengan algoritma *Ratcliff/Obershelp* dalam menghitung kesamaan kalimat, maka didapatkan kesimpulan sebagai berikut:

1. Algoritma *Rabin-Karp* dan *Ratcliff/Obershelp* dapat digunakan untuk mendeteksi atau menghitung tingkat kemiripan teks/kalimat dalam Bahasa Indonesia.
2. Perhitungan menggunakan algoritma *Ratcliff/Obershelp* akan menghasilkan persentase yang lebih mendekati persentase data masukan dibandingkan algoritma *Rabin-Karp.* Selisih nilai similarity rata-rata yang didapatkan dari dua algoritma tersebut adalah ± 3%.
3. Tata letak kalimat sangat mempengaruhi nilai *similarity* algoritma *Rabin-Karp,* dimana algoritma ini menghasilkan nilai *similarity* yang berbeda satu sama lainnya ketika diuji dengan *Data-Uji* yang dihapus/dipotong pada tempat yang berbeda (di awal, akhir dan acak). Sedangkan algoritma *Ratcliff/Obershelp* sama sekali tidak dipengaruhi oleh hal yang demikian.
4. Algoritma *Rabin-Karp* juga memiliki tingkat konsistensi yang rendah dibandingkan dengan algoritma *Ratcliff/Obershelp.* Algoritma *Ratcliff/Obershelp* hanya memiliki 2 nilai kemiripan (94 dan 95) untuk pengujian dengan menggunakan 10 data yang berbeda pada *Data-Uji* 90% (DU1) sedangkan Algoritma *Rabin-Karp* menghasilkan 5 nilai kemiripan yang berbeda (93, 94, 95, 96 dan 97) pada pengujian yang sama.
5. Banyak jumlah karakter atau kalimat dalam data akan mempengaruhi waktu yang dibutuhkan dalam mendeteksi tingkat kemiripan. Dalam hal ini, algoritma *Rabin-Karp* memiliki waktu eksekusi yang lebih lama dibandingkan algoritma *Ratcliff/Obershelp*. Algoritma *Rabin-Karp* membutuhkan waktu 2 sampai dengan 7 *ms* untuk pengujian dengan *Data-Uji DU4, DU3, DU2, DU1,* dan *DU* sedangkan algoritma *Ratcliff/Obershelp* hanya memerlukan waktu 0,1 sampai dengan 2 *ms* untuk pengujian data yang sama.

DAFTAR PUSTAKA

[1] R. Satria, Tarmizi, and Melviana, "Identifikasi bentuk tindak plagiat pada penulisan skripsi mahasiswa program studi pendidikan fisika unsyiah," *J. Ilm. Mhs. Pendidik. Fis.*, vol. 2, no. 2, pp. 231–237, 2017.

[2] B. Nurgiyantoro, W. Purbani, and Sutiyono, *Panduan Antiplagiarisme*. Yogyakarta, 2015.

[3] Afdhal; Taufan Chalis; Tauiq A. Gani, "Analisa Perbandingan Aplikasi Pendeteksi Plagiat Terhadap Karya Ilmiah," *Semin. Nas. dan Expo Tek. Elektro*, no. May, pp. 193–199, 2014.

[4] Y. L. Joane, A. Sinsuw, and A. Jacobus, "Rancang Bangun Aplikasi Deteksi Kemiripan Dokumen Teks Menggunakan Algoritma Ratcliff/Obershelp," *J. Tek. Inform.*, vol. 11, no. 1, 2017.

[5] I. Widiastuti, C. Rahmad, and Y. Ariyanto, "Aplikasi pendeteksi kemiripan pada dokumen menggunakan algoritma Rabin Karp," *J. Inform. Polinema*, no. 2, pp. 13–18, 2015.

[6] K. T. Tung, N. D. Hung, L. Thi, and M. Hanh, "A Comparison of Algorithms used to measure the Similarity between two documents," *Int. J. Adv. Res. Comput. Eng. Technol.* , no. April, 2015.

[7] N. ALAMSYAH, "Perbandingan Algoritma Winnowing Dengan Algoritma Rabin Karp Untuk Mendeteksi Plagiarisme Pada Kemiripan Teks Judul Skripsi," *Technol. J. Ilm.*, vol. 8, no. 3, p. 124, 2017.

[8] A. H. Purba and Z. Situmorang, "Analisis Perbandingan Algoritma Rabin-Karp Dan Levenshtein Distance Dalam Menghitung Kemiripan Teks," *J. Tek. Inform. Unika St. Thomas*, vol. 02, pp. 24–32, 2017.

[9] Salmuasih and A. Sunyoto, "Implementasi Algoritma Rabin Karp untuk Pedeteksi Plagiat Dokumen Teks Menggunakan Konsep Similarity," *Semin. Nas. Apl. Teknol. Inf.*, pp. F23–F28, 2013.

[10] I. Ilyankou, "Comparison of Jaro-Winkler and Ratcliff / Obershelp algorithms in spell check," UWC Adriatic, 2014.