DECODE: JURNAL PENDIDIKAN TEKNOLOGI INFORMASI
P-ISSN: 2775-2984 | E-ISSN: 2775-1813
Vol. 4, No 3, November 2024, Hal: 844-853
Doi: http://dx.doi.org/10.51454/decode.v4i3.643
https://journal.umkendari.ac.id/index.php/decode

Implementasi Algoritma Ratcliff/Obershelp pada Pendeteksian Similaritas Dokumen

Maukar[1], Ety Sutanty[2*], Rini Arianty[2], Esti Setiyaningsih[3]

[1]Program Studi Manajemen Sistem Informasi, Universitas Gunadarma, Indonesia.
[2*]Program Studi Sistem Informasi, Universitas Gunadarma, Indonesia.
[3]Program Studi Informatika, Universitas Gunadarma, Indonesia.

Artikel Info

Kata Kunci:

Dokumen;
Plagiarsm
Ratcliff/Obershelp;
Similaritas.

Keywords:

Document;
Plagiarsm;
Ratcliff/Obershelp;
Similarity.

Riwayat Artikel:

Submitted: 25 Juni 2024
Accepted: 5 Oktober 2024
Published: 12 Oktober 2024

Abstrak: *Plagiarism* merupakan tindakan menjiplak karya orang lain dan mengakui sebagai hasil karya pribadinya. Pada penelitian ini melakukan pendeteksian similaritas dari dokumen dengan menghitung similaritas dokumen menggunakan algoritma Ratcliff/Obershelp. Tipe dokumen yang diuji adalah .pdf. Dokumen yang digunakan untuk perbandingan teks ini adalah dokumen yang berbahasa Indonesia. Tahapan *preprocessing* pada penelitian ini dilakukan dengan menghitung nilai similaritas yang terdiri dari *case folding*, tokenisasi, *filtering*, dan *stemming*. Setelah proses *preprocessing* maka tahap selanjutnya adalah dilakukan perhitungan menggunakan algoritma Ratcliff/Obershelp. Pada pengujian menggunakan 150 data dokumen yang akan dihitung nilai similaritasnya menggunakan algoritma Ratcliff/Obershelp menghasilkan nilai similaritas dokumen dengan tingkat kemiripan berdasarkan tiga kategori (tinggi, sedang dan rendah). Hasil deteksi similaritas pada penelitian ini diharapkan dapat membantu proses pengerjaan pendeteksian perbandingan dua buah dokumen dalam jumlah yang sangat banyak.

Abstract: *Plagiarism is the act of plagiarizing someone else's work and admitting it as their personal work. In this research, we detect the similarity of documents by calculating document similarity using the Ratcliff/Obershelp algorithm. The document type tested is .pdf. The documents used for this text comparison are documents in Indonesian. The preprocessing stages in this research were carried out by calculating similarity values consisting of case folding, tokenization, filtering, and stemming. After the preprocessing process, the next stage is to carry out calculations using the Ratcliff/Obershelp algorithm. In testing using 150 document data whose similarity values will be calculated using the Ratcliff/Obershelp algorithm, it produces document similarity values with a level of similarity based on three categories (high, medium and low). It is hoped that the results of similarity detection in this research can help the process of detecting comparisons of two documents in very large numbers.*

Corresponding Author:
Ety Sutanty
Email: ety_s@staff.gunadarma.ac.id

PENDAHULUAN

Pemanfaatan teknologi digital telah menjadi kebutuhan dalam era modern saat ini. Salah satu komponen yang ada dalam dunia yaitu dokumen teks. Dokumen dalam bentuk digital memudahkan dalam hal penyimpanan, efisien, mudah dicari, bahkan mudah dalam hal plagiarisme (Najm Mansoor & Al-Tamimi, 2022). Praktik plagiarisme sering terjadi dalam dunia akademik, baik tingkat sekolah maupun perguruan tinggi. Tindakan plagiarisme dilakukan dengan mengganti kata-kata yang mengandung sinonim, dengan maksud agar terlihat berbeda dari hasil pekerjaan teman(Fauzi et al., 2022). Plagiarisme berarti mencontoh atau meniru atau mencuri tulisan dan karya orang lain yang kemudian diakui sebagai karangannya sendiri dengan ataupun tanpa seizin penulisnya. Plagiarisme dokumen digital bukanlah hal yang sulit untuk dilakukan, cukup dengan menggunakan teknik *copy-paste-modify* pada sebagian isi dokumen dan bahkan keseluruhan isi dokumen sudah bisa dikatakan bahwa dokumen tersebut merupakan hasil duplikasi dari dokumen lain(Leonardo & Hansun, 2017).

Salah satu cara untuk mengatasi permasalahan terjadinya plagiarisme yaitu dengan mencegah dan mendeteksi. Mencegah berarti menjaga atau menghalangi agar tindakan plagiarisme tidak dilakukan. Usaha ini harus dilakukan sedini mungkin terutama pada sistem pendidikan dan moral masyarakat. Mendeteksi berarti melakukan usaha untuk menemukan tingkat plagiarisme yang telah dilakukan. Cara mendeteksi plagiarisme terbagi menjadi dua (Saeed & Taqa, 2022) yang pertama secara manual dan yang kedua secara otomatis. Secara manual dapat dilakukan dengan cara membandingkan antara kedua jenis dokumen tersebut namun hal tersebut tentulah kurang efektif. Pendeteksian secara manual sebenarnya mempunyai tingkat akurasi tinggi, dikarenakan kemampuan manusia dalam memahami makna dan gaya bahasa dari kata atau kalimat yang sangat terbatas. Keterbatasan pendeteksian secara manual membutuhkan waktu dan tenaga yang banyak jika mendeteksi dokumen yang sangat banyak sehingga menjadi tidak efektif di dalam proses pengerjaannya(Omar et al., 2024).

Beberapa teknik dalam mendeteksi kemiripan dua buah dokumen dapat dilakukan dengan mengimplementasikan penggunakaan metode metode *String Matching* atau *Pattern Matching*(Balani & Varol, 2021). Algoritma ini berasal dari sebuah *string* yang digunakan untuk menemukan satu atau semua keberadaan string dalam sekelompok besar. Konsep ini banyak dipakai di berbagai macam permasalahan dan digunakan dalam berbagai aplikasi seperti *text mining,* pengambilan data, maupun pencarian kata kunci dalam sebuah teks (Fadlil et al., 2022). Algoritma *string matching* memiliki dua teknik yaitu *Exact String Matching* (Prawira & Saputri, 2024), dimana pendeteksian terjadi apabila pola teks uji sepenuhnya cocok dengan pola teks penguji dam *Approximate matching*(Nuraminah & Ammar, 2023), dimana pendeteksian terjadi apabila ada bagian tertentu yang memenuhi dari pola.

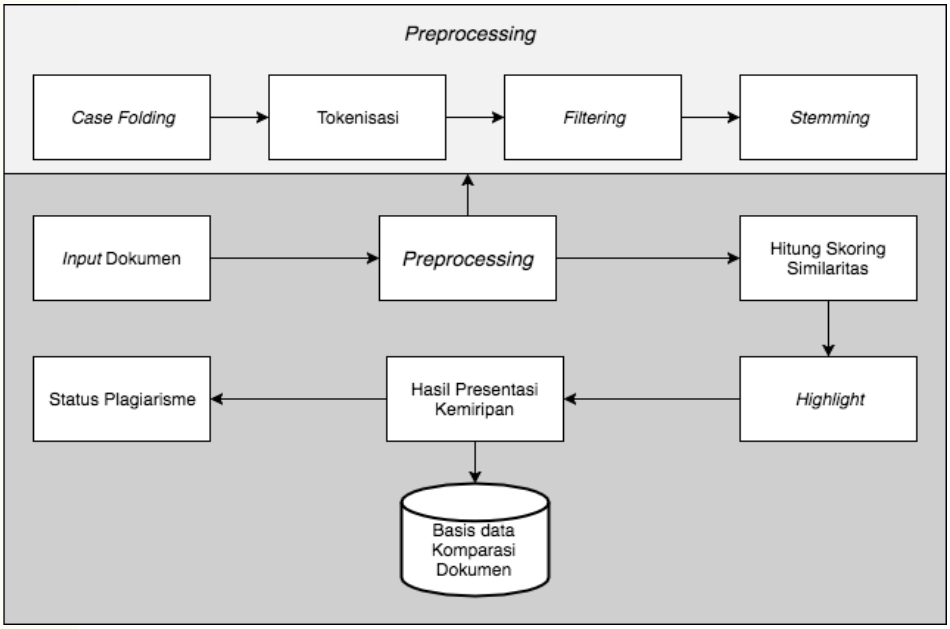
Penelitian terkait pendeteksian plagiarisme pada dokumen dilakukan peneliti terdahulu. Penelitian (Izzah et al., 2022) menghasilkan system untuk mendeteksi kemiripan teks pada teks berita berbahasa Indonesia dengan algoritma Ratcliff/Obershelp. Pendeteksian dilakukan dengan sebelumnya melalui tahapan stemming menggunakan algoritma Nazief dan Adriani . Algoritma Ratcliff/Obershelp yang digunakan pada penelitian ini dilakukan dengan menggabungkan string dari 2 buah teks untuk mendapatkan panjang karakter total (sequence (string) matching). Hasil penggabungan kemudian digunakan untuk mencari kata yang sama (subsequence) dengan menghitung panjang karakter. Hasil penelitian ini berhasil mengklasifikasikan jenis plagiarisme berupa 8 teks berita berbahasa indonesia terbagi atas 2 topik, menghasilkan persentase kesalahan sejumlah 0,26%. Penelitian (Puji Agung Kurniawan, 2023) mengimplementsikan penggunaan metode Ratcliff/Obershelp pada deteksi manajemen pendaftaran skripsi online. Metode Ratcliff/Obershelp yang digunakan pada penelitian ini menggunakan proses yang sama untuk menentukan seberapa mirip dua pola satu dimensi, karena string teks satu dimensi pada algoritma ini akan mengembalikan nilai yang dapat digunakan sebagai faktor kepercayaan atau persentase yang menunjukkan kesamaan dari dua string. Penggunaan konsep pencocokan dari algoritma pada penelitian ini dilakukan mencari substring terpanjang dengan string S1 dan S2. Penelitian dalam menguji tingkat similaritas dua dokumen menggunakan algoritma Rabin – karp dan Rolling Hash sebagai hashing dari K-gram yang dihasilkan dari kalimat pada dokumen juga dilakukan (Putera Utama Siahaan et al., 2017). Penelitian ini menganalisis dari sejumlah K-gram dari sebuah kalimat ayng mempengaruhi tingkat similaritas dari dokumen pengujian. Semakin besar

k-gram maka akan semakin kecil tingkat similaritasnya, tetapi jika k-gram yang digunakan kecil maka akan semakin tinggi tingkat similaritasnya namun mengurangi tingkat akurasi dari similaritas kedua dokumen yang dibandingkan.

Pada penelitian ini diimplementasikan penggunaan algoritma Ratcliff/Obershelp dengan menghitung presentasi nilai similaritas antara kedua dokumen berbentuk .pdf. Text Preprocessing dilakukan ditahap awal yaitu : *case folding*, tokenisasi, *filtering* dan *stemming*. Hasil tahapan preprocessing digunakan untuk menghitung nilai similaritas dengan menghitung nilai panjang *string* dari S_1 dan S_2, kemudian mencari karakter yang sama dari *string* dari S_1 dan S_2. Penelitian ini juga menghitung panjang karakter yang sama dari *string* dari S_1 dan S_2 dan memasukkan hasil perhitungan tersebut ke dalam variabel K_m. Pada karakter yang sama dari *string* dari S_1 dan S_2 maka langkah dalam melakukan mencari karakter yang sama sampai memasukkan panjang karakter yang sama ke dalam variabel K_m. Nilai similaritas penelitian ini dihiutng menggunakan rumus algoritma Ratcliff/Obershelp. Hasil penelitian bertujuan menghasilkan sistem pendeteksian untuk menentukan kesamaan (similaritas) dari dua buah dokumen secara cepat.

METODE

Proses pendeteksian similaritas dokumen berbentuk .pdf terdiri dari beberapa tahapan antara lain: *input* dokumen berupa *file* dokumen pdf, *preprocessing* yang terdiri atas *case folding*, tokenisasi, *filtering*, dan *stemming*. Setelah didapatkan hasil *stemming* selanjutnya adalah proses perhitungan persetanse similaritas, menampilkan *highlight*, dan hasil similaritas dokumen dimasukkan ke dalam basis data komparasi dokumen untuk ditampilkan hasil status plagiarisme. Bagan tahapan proses penelitian ini dapat dilihat pada Gambar 1.



Gambar 1. Tahapan Penelitian Pendeteksian Similaritas Dokumen

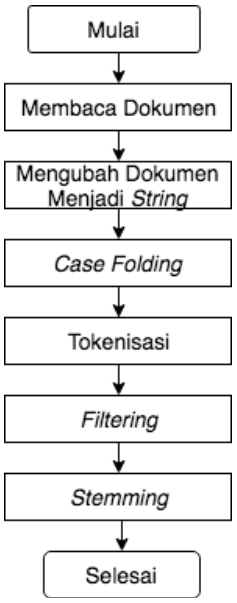
Seperti dapat dilihat pada Gambar 1, dokumen input disini adalah dokumen yang akan dilakukan analisa kemiripan dokumen. Segala konten yang di inputkan ke dalam sistem akan di analisa perkalimat dari konten tersebut. Kalimat per kalimat akan dipisahkan dari dokumen tersebut pada langkah selanjutnya. Setelah itu, per kalimat tersebut akan dilakukan tahapan selanjutnya yaitu querying google search. Dokumen kemudian masuk kedalam tahap *preprocessing*, untuk dihitung skor similaritasnya menggunakan algoritma Ratcliff/Obershelp. Hasil prosentase kemiripan menghasilkan status plagiarisme (*Low*, *Medium* dan *High Plagiarsm*).

***Input* Dokumen**

Pada tahap ini dokumen yang dimasukkan akan dibaca karakter perkarakter. Dokumen yang diizinkan hanya dokumen berekstensi pdf. Dokumen akan dibaca secara *binary*, kemudian diubah menjadi *string* yang disimpan pada sebuah variabel untuk satu dokumen.

Preprocessing

Proses *preprocessing* dilakukan sebelum menghitung nilai similaritas dokumen yang terdiri dari proses *case folding*, tokenisasi, *filtering*, dan *stemming*. Fungsi *preprocessing* pada program ini adalah untuk mendapatkan *keyword* yang akan digunakan sebagai pencocokan *string* atau perbandingan dokumen. Proses tersebut dapat dilihat pada Gambar 2.



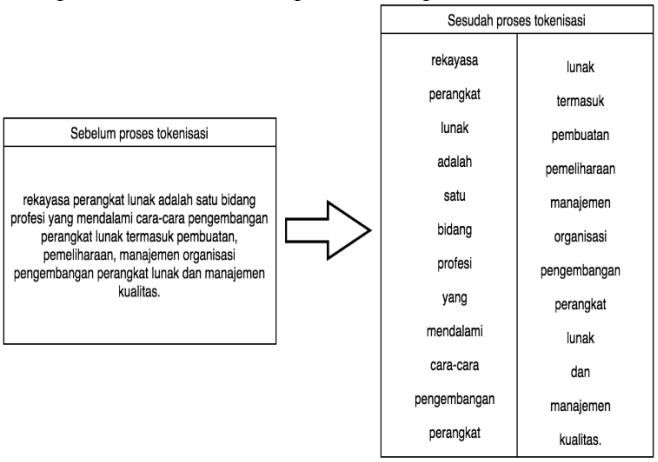
Gambar 2. Tahapan *Preprocessing* Pendeteksian Similaritas Dokumen

Tahapan yang dilakukan dalam *preprocessing* (Thombare et al., 2024) untuk mendeteksi similaritas dokumen pada penelitian ini antara lain :

1. ***Case Folding.*** Proses *case folding* adalah tahap mengubah semua huruf dalam dokumen menjadi huruf kecil (*lower case*) hanya huruf a sampai z yang diterima. Fungsi dari proses *case folding* ini adalah untuk menyamakan nilai similaritas dari kata yang sama tetapi hanya dibedakan dari penggunaan huruf besar dan kecil. Misalnya untuk penggunaan kata "Saya" dan "saya" akan memiliki nilai similaritas yang berbeda karena kesamaan karakter dalam kata tersebut adalah "aya" karena huruf depan kata "Saya" menggunakan huruf besar tidak sama dengan kata "saya" yang menggunakan huruf kecil. Contoh proses *case folding* dapat dilihat pada Gambar 3.

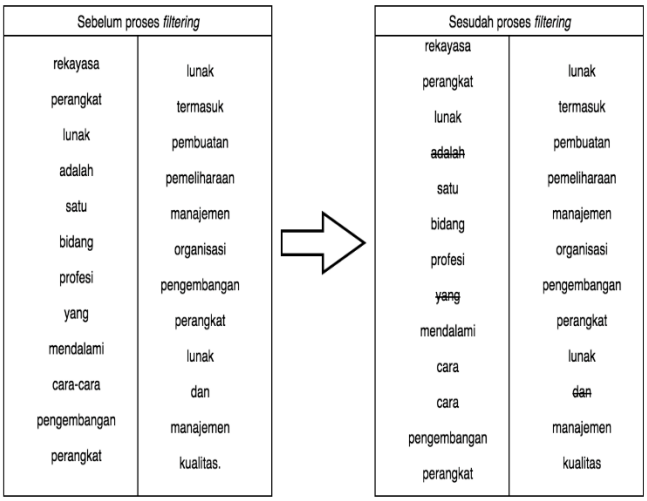
Gambar 3. Contoh Proses *Case Folding*

2. **Tokenisasi.** Pada dasarnya proses tokenisasi yaitu proses memisahkan setiap kata yang menyusun suatu dokumen. Pengertian tokenisasi itu sendiri adalah proses pemotongan *input* berdasarkan tiap kata yang menyusunnya. Umumnya setiap kata teridentifikasi atau terpisahkan dengan kata lain oleh karakter spasi, sehingga proses tokenisasi mengandalkan karakter spasi pada dokumen untuk melakukan pemisahan kata. Karakter spasi dan titik digunakan sebagai *delimiter* yang berfungsi untuk memisahkan kata perkata dan kalimat perkalimat. Contoh proses tokenisasi dapat dilihat pada Gambar 4.



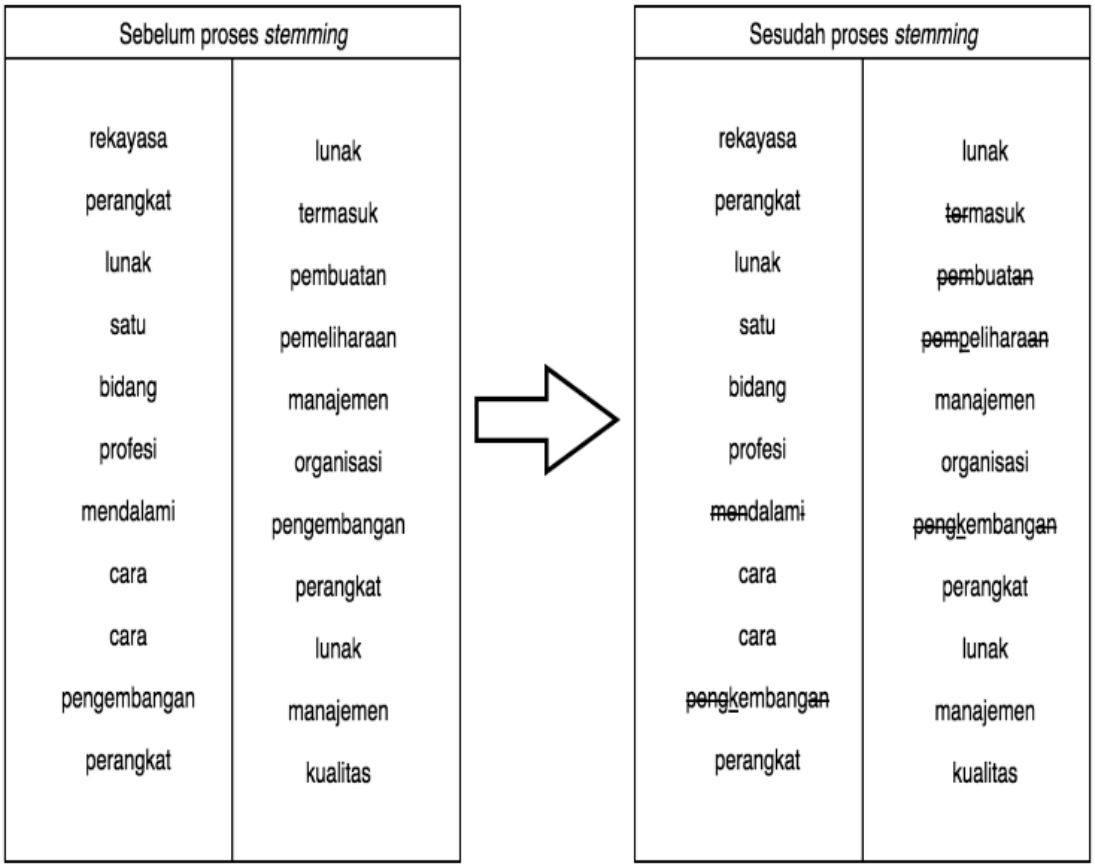
Gambar 4. Contoh Proses Tokenisasi

3. **Tahap *filtering*** merupakan tahap pengambilan kata yang penting dan membuang kata yang tidak penting dari hasil *tokenizing*. Pada tahap ini dapat menggunakan algoritma *stoplist* atau *wordlist*. *Stoplist* yaitu menyaring terhadap kata-kata yang tidak layak untuk dijadikan pembeda dari kata kunci sehingga kata-kata tersebut dapat dihilangkan. Kata-kata hasil tokenisasi akan dicocokkan dengan tabel dari *stoplist* yang telah dibuat. Apabila kata tersebut sama dengan kata yang ada dalam tabel *stoplist* maka akan dihilangkan. Pada proses *filtering* juga akan menghilangkan simbol-simbol seperti titik, koma, dan sebagainya sehingga untuk penutup kalimat seperti "terima kasih." dan "terima kasih" memiliki nilai similaritas yang berbeda karena pada kata "kasih." terdapat titik. Contoh proses *filtering* dapat dilihat pada Gambar 5.



Gambar 5. Contoh Proses *Filtering*

4. ***Stemming*** (El-Rashidy et al., 2024) merupakan proses mengubah kata menjadi kata dasarnya dengan menghilangkan imbuhan pada kata dalam dokumen. Dalam tahap *stemming* diperlukan suatu algoritma yaitu Nazief dan Andriani. Algoritma ini mengacu pada aturan morfologi bahasa Indonesia yang mengelompokkan imbuhan, yaitu imbuhan yang diperbolehkan atau imbuhan yang tidak diperbolehkan (Enni Lindrawati et al., 2023). Pengelompokan ini termasuk imbuhan di depan (awalan), imbuhan di belakang (akhiran) atau imbuhan di tengah kata (sisipan) dan kombinasi imbuhan pada awal dan akhir kata (konflik). Algoritma ini menggunakan kamus kata keterangan yang digunakan untuk mengetahui bahwa proses *stemming* telah mendapatkan kata dasar. Contoh proses *stemming* dapat dilihat pada Gambar 6. Pada penelitian ini menggunakan dokumen berbahasa indonesia.



Gambar 6. Contoh Proses *Stemming*

Perhitungan Nilai Similaritas Dokumen Dengan Algoritma Ratcliff/Obershelp

Secara umum, proses penentuan nilai similaritas dokumen menggunakan algoritma Ratcliff/Obershelp dapat dilihat pada Gambar 7. Pada algoritma Ratcliff/Obershelp (Hiebel et al., 2022)dalam menghitung nilai similaritas langkah pertama adalah menghitung nilai panjang *string* dari S_1 dan S_2, kemudian mencari karakter yang sama dari *string* dari S_1 dan S_2 lalu hitung panjang karakter yang sama dari *string* dari S_1 dan S_2 dan masukkan ke dalam variabel K_m (Özçevik et al., 2022). Jika ada karakter yang sama lagi dari *string* dari S_1 dan S_2 maka ulangi langkah (Apridiansyah et al., 2022)mencari karakter yang sama sampai memasukkan panjang karakter yang sama ke dalam variabel K_m.

Gambar 7. Tahap Menghitung Nilai Similaritas

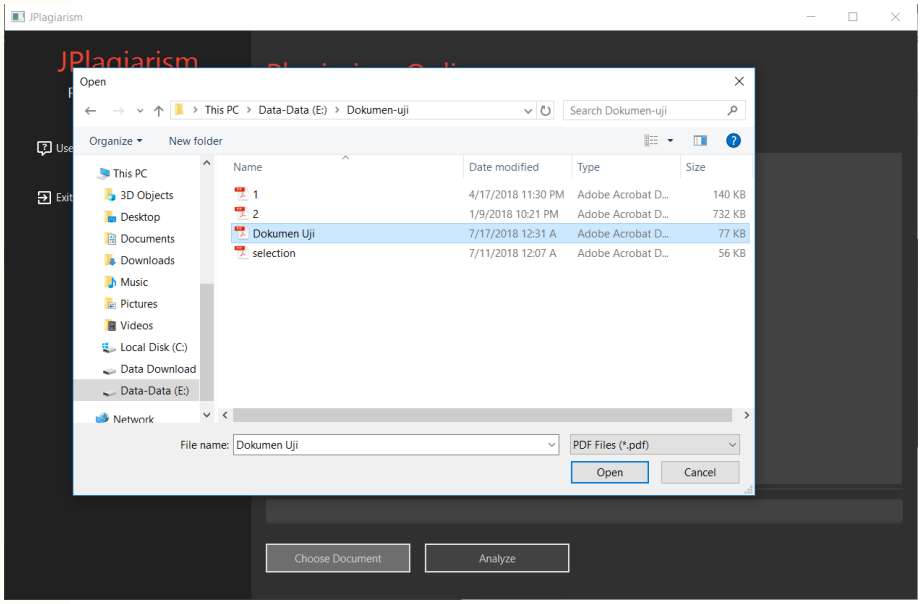
Perhitungan similaritas dilakukan jika sudah tidak ada lagi karakter yang sama. Nilai similaritas dapat dihitung dengan menggunakan rumus algoritma Ratcliff/Obershelp pada persamaan :

$$Dro = \frac{2*Km}{|S1|+|S2|} \quad (1)$$

Pada persamaan (1), variabel Dro merupakan nilai similaritas. Nilai ini dihitung menggunakan variabel Km yang merupakan jumlah panjang karakter yang sama, $|S1|$ yang merupakan panjang *string* S_1 dan $|S2|$ yang merupakajn panjang *string* S_2(Rahmatulloh et al., 2019).

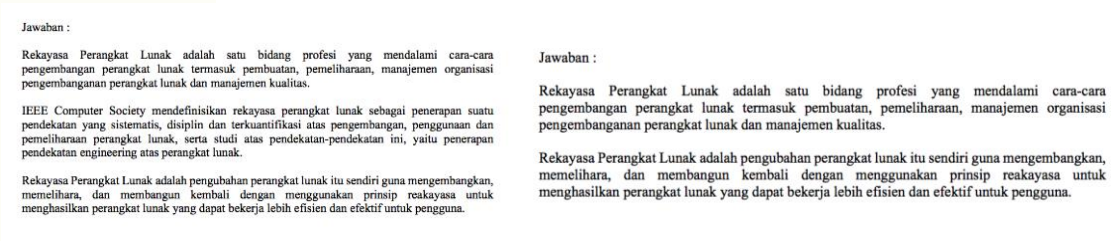
HASIL DAN PEMBAHASAN

Ujicoba dilakukan dengan menganalisa dokumen dengan memasukkan dokumen input berupa file pdf. Pada kasus ini dokumen input bernama "Dokumen Uji.pdf" akan dilakukan pengujian plagiarisme. Pilih dokumen lalu tekan open untuk membuka file tersebut seperti dapat dilihat pada Gambar 8.



Gambar 8. Pemilihan File Uji

Contoh dokumen uji yang digunakan dapat dilihat pada Gambar 9 merupakan file berbentuk .pdf.



Jawaban :

Rekayasa Perangkat Lunak adalah satu bidang profesi yang mendalami cara-cara pengembangan perangkat lunak termasuk pembuatan, pemeliharaan, manajemen organisasi pengembanganan perangkat lunak dan manajemen kualitas.

IEEE Computer Society mendefinisikan rekayasa perangkat lunak sebagai penerapan suatu pendekatan yang sistematis, disiplin dan terkuantifikasi atas pengembangan, penggunaan dan pemeliharaan perangkat lunak, serta studi atas pendekatan-pendekatan ini, yaitu penerapan pendekatan engineering atas perangkat lunak.

Rekayasa Perangkat Lunak adalah pengubahan perangkat lunak itu sendiri guna mengembangkan, memelihara, dan membangun kembali dengan menggunakan prinsip reakayasa untuk menghasilkan perangkat lunak yang dapat bekerja lebih efisien dan efektif untuk pengguna.

Jawaban :

Rekayasa Perangkat Lunak adalah satu bidang profesi yang mendalami cara-cara pengembangan perangkat lunak termasuk pembuatan, pemeliharaan, manajemen organisasi pengembanganan perangkat lunak dan manajemen kualitas.

Rekayasa Perangkat Lunak adalah pengubahan perangkat lunak itu sendiri guna mengembangkan, memelihara, dan membangun kembali dengan menggunakan prinsip reakayasa untuk menghasilkan perangkat lunak yang dapat bekerja lebih efisien dan efektif untuk pengguna.

Gambar 9. Dokumen Uji

Setelah file terpilih maka akan tampil teks hasil *parsing* file pdf tersebut menjadi teks dengan library iText, maka teks akan terisi dan pencarian similaritas akan dapat dilakukan. Pada tahap pencarian similaritas ini waktu yang dibutuhkan dalam pencarian bergantung pada koneksi internet dan banyaknya kalimat pada masukan dokumen. Hasil uji kedua dokumen dapat dilihat pada Gambar 10.

Jawaban :
Rekayasa Perangkat Lunak (RPL, atau dalam bahasa Inggris: Software Engineering atau SE) adalah satu bidang profesi yang mendalami cara-cara pengembangan perangkat lunak termasuk pembuatan, pemeliharaan, manajemen organisasi pengembangan perangkat lunak dan manajemen kualitas.
Kriteria yang dapat digunakan sebagai acuan dalam merekayasa perangkat lunak, yaitu dapat terus dirawat dan dipelihara (maintainability), dapat mengikuti perkembangan teknologi (dependability), dapat mengikuti keinginan pengguna (robust), efektif dan efisien dalam menggunakan energi dan penggunaannya, dan dapat memenuhi kebutuhan yang diinginkan (usability).

Jawaban :
Rekayasa Perangkat Lunak (RPL, atau dalam bahasa Inggris: Software Engineering atau SE) adalah satu bidang profesi yang mendalami cara-cara pengembangan perangkat lunak termasuk pembuatan, pemeliharaan, manajemen organisasi pengembangan perangkat lunak dan manajemen kualitas.
Tujuan Rekayasa Perangkat Lunak secara lebih khusus kita dapat menyatakan tujuan dan Rekayasa Perangkat Lunak ini adalah memperoleh biaya produksi perangkat lunak yang rendah, menghasilkan perangkat lunak yang kinerjanya tinggi, andal dan tepat waktu, menghasilkan perangkat lunak yang dapat bekerja pada berbagai jenis platform, dan menghasilkan perangkat lunak yang biaya perawatannya rendah.

Tingkat Kesamaan : 53 %
Keterangan : High Plagiarism

Highlight Dokumen 1

jawab rekayasa perangkat lunak rpl dalam bahasa inggris software engineering se adalah satu bidang profesi dalam cara cara kembang perangkat lunak masuk buat pelihara manajemen organisasi kembang perangkat lunak manajemen kualitas

Highlight Dokumen 2

jawab rekayasa perangkat lunak rpl dalam bahasa inggris software engineering se adalah satu bidang profesi dalam cara cara kembang perangkat lunak masuk buat pelihara manajemen organisasi kembang perangkat lunak manajemen kualitas

Gambar 10. Hasil Uji Deteksi Simimlaritas

Seperti dapat dilihat pada Gambar 10, hasil uji menghasilkan prosentase berupa nilai indeks simlatitas. Indeks similaritas dihitung berdasarkan persentase hasil algoritma Ratcliff/Obershelp seperti dapat dilihat pada Tabel 1. Dua buah dokumen yang telah dihitung nilai similaritasnya dapat di kategori *low plagiarism* (plagiarisme ringan) jika nilai indeks similaritasnya kurang dari 15%, jika dua buah dokumen yang telah dihitung nilai similaritasnya memiliki nilai indeks similaritas antara 15% – 40% maka di kategorikan *middle plagiarism* (plagiarisme sedang) dan jika dua buah dokumen yang telah dihitung nilai similaritasnya memiliki nilai indeks similaritas lebih dari 40% maka di kategorikan (plagiarisme berat) *high plagiarism*.

Tabel 1. Tabel Indeks Similaritas (Turnitin, 2021)

Presentase Similaritas	Keterangan
< 15 %	*Low Plagiarism*
15 – 40 %	*Middle Plagiarism*
> 40 %	*High Plagiarism*

KESIMPULAN

Berdasarkan hasil uji coba deteksi kesamaan dokumen, pada proses proses seperti case folding berhasil melakukan normalisasi terhadap sejumlah dokumen. Proses tokenisasi juga berhasil melakukan pemisahan kata pada sejumlah dokumen dan proses filtering berhasil menghilangkan kata-kata yang tidak mempunyai arti seperti kata sambung pada sebuah dokumen. Penggunaan algoritma Nazief-Adriani berhasil menghilangkan imbuhan pada kata untuk dijadikan kata dasar yang terdapat pada dokumen. Similaritas dokumen juga berhasil dihitung menggunakan algoritma Ratcliff/Obershelp, sehingga persentase indeks similaritas dokumen berhasil ditentukan dengan mengacu pada tabel indeks similaritas Turnitin. Pengembangan lebih lanjut dapat dilakukan untuk menyempurnakan proses perhitungan nilai similaritas dengan aplikasi dapat menerima informasi dari gambar dan aplikasi dapat mengenali dokumen menggunakan bahasa Inggris.

DAFTAR PUSTAKA

Apridiansyah, Y., Wijaya, A., & Purjiawan, A. (2022). Penerapan Fungsi Metode Rolling Hash Pada Algoritma Winnowing Untuk Mendeteksi Kemiripan Teks Abstrak Berbasis Web. *Jurnal Media Infotama*, *18*(1), 128–133.

Balani, Z., & Varol, C. (2021). Combining Approximate String Matching Algorithms and Term Frequency In The Detection of Plagiarism. *International Journal of Computer Science and Security (IJCSS)*, *15*(4), 97–105.

El-Rashidy, M. A., Mohamed, R. G., El-Fishawy, N. A., & Shouman, M. A. (2024). An effective text plagiarism detection system based on feature selection and SVM techniques. In *Multimedia Tools and Applications* (Vol. 83, Issue 1). Springer US. https://doi.org/10.1007/s11042-023-15703-4

Enni Lindrawati, Ema Utami, & Yaqin, A. (2023). ANoM STEMMER: Nazief & Andriani Modification for Madurese Stemming. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *7*(6), 1341–1347. https://doi.org/10.29207/resti.v7i6.5086

Fadlil, A., Sunardi, S., & Ramdhani, R. (2022). Similarity Identification Based on Word Trigrams Using Exact String Matching Algorithms. *INTENSIF: Jurnal Ilmiah Penelitian Dan Penerapan Teknologi Sistem Informasi*, *6*(2), 253–270. https://doi.org/10.29407/intensif.v6i2.18141

Fauzi, R., Iqbal, M., & Haryanti, T. (2022). Design and Implementation of a Final Project Plagiarism Detection System Using Cosine Similarity Method. *IJAIT (International Journal of Applied Information Technology)*, *05*(02), 1. https://doi.org/10.25124/ijait.v5i02.4146

Hiebel, N., Ferret, O., Fort, K., & Névéol, A. (2022). CLISTER : A Corpus for Semantic Textual Similarity in French Clinical Narratives. *Traitement Automatique Des Langues Naturelles, TALN 2022 - Actes*

de La 29e Conference Sur Le Traitement Automatique Des Langues Naturelles: Conference Principale, *1*(June), 287–296.

Izzah, N., Yusliani, N., & Roodiah, D. (2022). Sistem Deteksi Kemiripan Teks Pada Berita Berbahasa Indonesia Menggunakan algoritma Ratcliff/Obershelp. *Jurnal Linguistik Komputasional (JLK)*, *5*(1), 1. https://doi.org/10.26418/jlk.v5i1.65

Leonardo, B., & Hansun, S. (2017). Text documents plagiarism detection using Rabin-Karp and Jaro-Winkler distance algorithms. *Indonesian Journal of Electrical Engineering and Computer Science*, *5*(2), 462–471. https://doi.org/10.11591/ijeecs.v5.i2.pp462-471

Najm Mansoor, M., & Al-Tamimi, M. S. H. (2022). Computer-based plagiarism detection techniques: A comparative study. *Int. J. Nonlinear Anal. Appl*, *13*(February), 2008–6822. https://doi.org/10.22075/ijnaa.2022.6140

Nuraminah, A., & Ammar, A. (2023). Damerau-Levenshtein Distance Algorithm Based on Abstract Syntax Tree to Detect Code Plagiarism. *Scientific Journal of Informatics*, *11*(1), 11–20. https://doi.org/10.15294/sji.v11i1.48064

Omar, K., Esmaeel, N., & Ebrahim, Z. (2024). *Intelligent Systems And Applications In Engineering Review on Plagiarism Detection Systems , Algorithms , Weakness Points*.

Özçevik, Y., Yücalar, F., & Demircioğlu, M. (2022). Determining a Proper Text Similarity Approach for Resume Parsing Process in a Digitized HR Software. *Celal Bayar University Journal of Science*, *18*(4), 371–378. https://doi.org/10.18466/cbayarfbe.1049845

Prawira, J., & Saputri, T. R. D. (2024). Lost item identification model development using similarity prediction method with CNN ResNet algorithm. *Journal of Autonomous Intelligence*, *7*(2), 1–14. https://doi.org/10.32629/jai.v7i2.1381

Puji Agung Kurniawan, M. A. E. N. (2023). Jurnal ITCC (Information Technology and Cyber Crime). *Jurnal ITCC*, *2*(1), 2964–755.

Putera Utama Siahaan, A., Rahim, R., & Siregar, D. (2017). K-Gram As A Determinant Of Plagiarism Level In Rabin-Karp Algorithm. *International Journal of Scientific & Technology Research*, *6*(07), 7.

Rahmatulloh, A., Kurniati, N. I., Darmawan, I., Asyikin, A. Z., & Witarsyah, J. D. (2019). Comparison between the stemmer porter effect and nazief-adriani on the performance of winnowing algorithms for measuring plagiarism. *International Journal on Advanced Science, Engineering and Information Technology*, *9*(4), 1124–1128. https://doi.org/10.18517/ijaseit.9.4.8844

Saeed, A. A. M., & Taqa, A. Y. (2022). A proposed approach for plagiarism detection in Article documents. *SinkrOn*, *7*(2), 568–578. https://doi.org/10.33395/sinkron.v7i2.11381

Thombare, V., Joshi, S., & Deshpande, L. (2024). *AI ML Techniques To Find Nearest Matching Records*. *11*(5), 450–453.

Turnitin. (2021). *Understanding the Turnitin Similarity Report A student guide*. 1–2. https://help.turnitin.com/Resources/PDF/understanding_the_turnitin_similarity_report-a_student_guide.pdf