Jurnal Teknik Informatika dan Teknologi Informasi
Volume. 5 Nomor. 1 April 2025
E-ISSN: 2827-9387, P- ISSN: 2827-9379, Hal 132-145
DOI: https://doi.org/10.55606/jutiti.v5i1.5154
Available Online at : https://journalshub.org/index.php/jutiti

Penerapan Algoritma *Knuth Morris Pratt* untuk Mendeteksi Tingkat Plagiarisme pada Tugas Mahasiswa

Erisya Herlina[1*], Elvi Rahmi[2]
[1,2] Jurusan Teknik Informatika, Program Studi Rekayasa Perangkat Lunak,
Politeknik Negeri Bengkalis, Indonesia
Email: erisyaherlina@gmail.com[1], elvirahmi@polbeng.ac.id [2]

Alamat: Jl. Bathin Alam, Sungai Alam, Bengkalis - Riau 28712
Korespondensi penulis: erisyaherlina@gmail.com[*]

Abstract*. The rapid development of information technology has made it easier to access various sources of information, but it has also increased the risk of plagiarism, especially among university students. To maintain academic integrity and the quality of education, plagiarism detection is essential. This study aims to develop a plagiarism detection system using the Knuth-Morris-Pratt (KMP) algorithm combined with the Jaccard Similarity method to measure the similarity between documents. The system is designed to analyze Indonesian-language essay documents in Word (.docx) and PDF (.pdf) formats that do not contain images or tables. The detection process begins with data preprocessing steps, including case folding, tokenizing, stopword removal, and stemming, to enhance efficiency and accuracy. The system is developed using the Python programming language and the Flask framework. Test results show that the system can detect plagiarism with an average accuracy of 98.66%. The level of similarity between documents is then categorized into three levels: Minor, Moderate, and Severe Plagiarism. This system is expected to serve as an effective tool in minimizing plagiarism practices in higher education and supporting the creation of an honest and responsible academic culture.*

Keywords*: Jaccard Similarity, Knuth Morris Pratt, Plagiarism Detection, Preprocessing, Student Assignments.*

Abstrak. Perkembangan teknologi informasi yang pesat telah memudahkan akses terhadap berbagai sumber informasi, namun juga meningkatkan risiko terjadinya plagiarisme, khususnya di kalangan mahasiswa. Untuk menjaga integritas akademik dan kualitas pendidikan, deteksi plagiarisme menjadi sangat penting. Penelitian ini bertujuan membangun sistem deteksi plagiarisme menggunakan algoritma Knuth Morris Pratt (KMP) yang dikombinasikan dengan metode Jaccard Similarity untuk mengukur tingkat kemiripan antar dokumen. Sistem ini dirancang untuk menganalisis dokumen esai berbahasa Indonesia dalam format Word (.docx) dan PDF (.pdf) yang tidak mengandung gambar atau tabel. Proses deteksi diawali dengan tahapan preprocessing data yang meliputi case folding, tokenizing, stopword removal, dan stemming guna meningkatkan efisiensi dan akurasi. Pengembangan sistem dilakukan menggunakan bahasa pemrograman Python dengan kerangka kerja Flask. Hasil pengujian menunjukkan bahwa sistem mampu mendeteksi tingkat plagiarisme dengan akurasi rata-rata mencapai 98,66%. Tingkat kemiripan antar dokumen kemudian dikelompokkan ke dalam tiga kategori: Plagiarisme Ringan, Sedang, dan Berat. Sistem ini diharapkan dapat menjadi alat bantu efektif dalam meminimalisir praktik plagiarisme di lingkungan pendidikan tinggi serta mendukung terciptanya budaya akademik yang jujur dan bertanggung jawab.

Kata Kunci: Deteksi Plagiarisme, *Preprocessing*, *Knuth Morris Pratt*, *Jaccard Similarity*, Tugas Mahasiswa.

1. LATAR BELAKANG

Era digital telah membawa dampak signifikan dalam berbagai aspek kehidupan, termasuk pendidikan. Internet memudahkan individu untuk berbagi karya tulis dan mencari informasi dengan cepat. Situs pencari seperti Google dan platform seperti youtube menyediakan beragam informasi, mulai dari artikel, gambar, video, hingga berita. Kemajuan teknologi ini membantu siswa dalam mengakses referensi untuk tugas-tugas, namun juga membuka peluang untuk plagiarisme. Perkembangan pada dunia teknologi informasi mengakibatkan perguruan tinggi mengurangi penggunaan kertas sehingga

Received: April 23, 2025; Revised: Mei 08, 2025; Accepted: Mei 22, 2025; Published: Mei 24, 2025

banyak tugas mahasiswa yang dikumpulkan dalam bentuk digital. Penggunaan digital menyebabkan semakin mudahnya mahasiswa untuk melakukan plagiarisme. Sehingga diperlukan penerapan algoritma yang efektif untuk mendeteksi tingkat plagiarisme antar dokumen. Salah satu algoritma yang dapat digunakan adalah Algoritma *Knuth Morris Pratt,* yang mampu melakukan pencocokan string secara efisien. Dengan algoritma yang tepat, pendeteksian tingkat plagiarisme pada dokumen tugas mahasiswa dapat dilakukan dengan akurasi dan kecepatan yang tinggi.

Plagiarisme adalah sebuah tindakan mengambil atau menjiplak karya, gagasan atau ide-ide orang lain baik sengaja maupun tidak sengaja dan mengeklaim sebagai karya sendiri tanpa menyebutkan sumber atau penulisnya. Seringkali mahasiswa melakukan plagiat pada tugas yang diberikan oleh dosen sehingga terkadang mahasiswa hanya menyalin tugas orang lain untuk menyelesaikan tugas tersebut. Sehingga hal ini bisa menyebabkan adanya ketergantungan kepada orang lain dalam mengerjakan tugas dan tidak dapat secara mandiri mengerjakan tugas yang diberikan oleh dosen (Ramli et al., 2021).

Penelitian ini bertujuan untuk menerapkan algoritma Knuth Morris Pratt dalam sistem deteksi tingkat plagiarisme pada tugas mahasiswa serta untuk mengevaluasi efektivitas algoritma tersebut dalam pencocokan dan pendeteksian teks secara digital. Algoritma yang digunakan untuk mendeteksi tingkat plagiarisme adalah algoritma Knuth Morris Pratt, dan sistem yang dibangun hanya difokuskan untuk mendeteksi tingkat plagiarisme pada tugas mahasiswa yang berbentuk esai dalam bahasa Indonesia berupa dokumen Word (.doc/.docx) dan PDF yang tidak mengandung gambar maupun tabel.

Algoritma KMP adalah salah satu algoritma pencarian string, dikembangkan secara terpisah oleh Donald E. Knuth pada tahun 1967 dan James H. Morris bersama Vaughan R. Pratt pada tahun 1966, namun keduanya mempublikasikannya secara bersamaan pada tahun 1977 (Astuti, 2017). Jika kita melihat algoritma brute force lebih mendalam, kita mengetahui bahwa dengan mengingat beberapa perbandingan yang dilakukan sebelumnya kita dapat meningkatkan besar pergeseran yang dilakukan. (M. Ilham., 2020).

2. KAJIAN TEORITIS

Text Mining adalah proses menemukan hal baru, yang sebelumnya tidak diketahui, mengenai informasi yang berpotensi untuk diambil manfaatnya dari sumber data yang tidak terstruktur, mencakup dokumen bisnis, komentar pelanggan, halaman web, dan file XML. *Text mining* hampir sama dengan *data mining* dalam hal tujuan dan proses, tetapi

pada text mining inputnya adalah file data yang tidak terstruktur seperti dokumen dalam bentuk Word, PDF, XML, dan sebagainya. Tujuan dari *text mining* adalah untuk mendapatkan informasi yang berguna dari sekumpulan dokumen. Jadi, sumber data yang digunakan pada text mining adalah kumpulan teks yang memiliki format yang tidak terstruktur atau minimal semi terstruktur. Adapun tugas khusus dari *text mining* antara lain pengkategorisasian teks (*text categorization*) dan pengelompokan teks (*text clustering*(Kambey et al., n.d., 2020).

Jaccard Algorithm atau dikenal dengan *Jaccard Coefficient* dan atau *Jaccard Similarity* adalah salah satu metode yang dipakai untuk menghitung similarity antara dua objek (items). Algoritma *Jaccard Similarity* digunakan untuk menghitung nilai similaritas dari dokumen uji yang melakukan pembandingan dengan dokumen pembanding, Algoritma *Jaccard Similarity* menerima nilai dari dokumen uji dan nilai dari dokumen pembanding, setelah mendapatkan nilai kemudian melakukan proses perhitungan, dari proses perhitungan nilai dari dokumen uji dan dokumen pembanding tersebut kemudian didapat nilai *similarity* dari dokumen uji(Utomo et al., n.d., 2022).

3. METODE PENELITIAN

Pengumpulan Data

Pada tahap ini, peneliti menggunakan data tugas mahasiswa jurusan Teknik Informatika, prodi Rekayasa Perangkat Lunak, pada mata kuliah Artificial Intelligence. Data ini digunakan sebagai data latih untuk sistem pendeteksi tingkat plagiarisme, dengan melakukan pengumpulan data tugas mahasiswa pada mata kuliah Artificial Intelligence, sebanyak 31 data tugas mahasiswa.

Preprocessing Data

Pada tahapan ini peneliti melakukan preprocessing pada data latih sebagai berikut:

a. *Question Separator* , adalah sebuah proses atau fungsi yang digunakan untuk memisahkan teks soal dari teks lainnya, seperti jawaban atau bagian dokumen lain yang tidak termasuk soal.
b. *Case Folding*, adalah proses untuk memanipulasi text, semua text akan diubah menjadi huruf kecil.
c. *Punctuation Removal*, adalah proses menghapus karakter-karakter unik seperti karakter tanda seru, tanda tanya, tanda koma, dan sebagainya.
d. *Stopword Removal*, adalah proses penghapusan kata yang tidak relevan atau kata kata

yang dianggap tidak memiliki informasi penting dalam analisis teks. Contoh

e. stopword adalah “yang”, “dan”, “di”, “dari” dan sebagainya.
f. *Tokenizing*, adalah proses pemisahan kata berdasarkan susunan kata. Hasil dari pemisahan kata disebut token.
g. Stemming, adalah proses untuk merubah kata menjadi kata dasar.

Penerapan Algoritma *Knuth Morris Pratt*

Algoritma Knuth Morris Pratt pada saat pencocokan string adalah sebagai berikut:

a) Algoritma *Knuth Morris Pratt* mulai mencocokan pattern pada awal teks.
b) Dari kini ke kanan, Algoritma ini akan mencocokan karakter per karakter *pattern*, dengan karakter di teks yang bersesuaian sampai salah satu kondisi berikut terpenuhi.
   - Karakter di *pattern* dan diteks yang dibandingkan tidak cocok (*mismatch*)
   - Semua Karakter di *pattern* cocok. Kemudian Algoritma akan memberitahu penemuan posisi ini.
c) Algoritma kemudian menggeser pattern berdasarkan table next, lalu menghitung langkah 2 sampai pattern berada di ujung teks.

Jaccard Similarity

Jaccard Algorithm atau dikenal dengan *Jaccard Coefficient* dan atau *Jaccard Similarity* adalah salah satu metode yang dipakai untuk menghitung *similarity* antara dua objek (items) (Utomo et al., n.d., 2022).. Persamaan dapat dilihat pada (1)

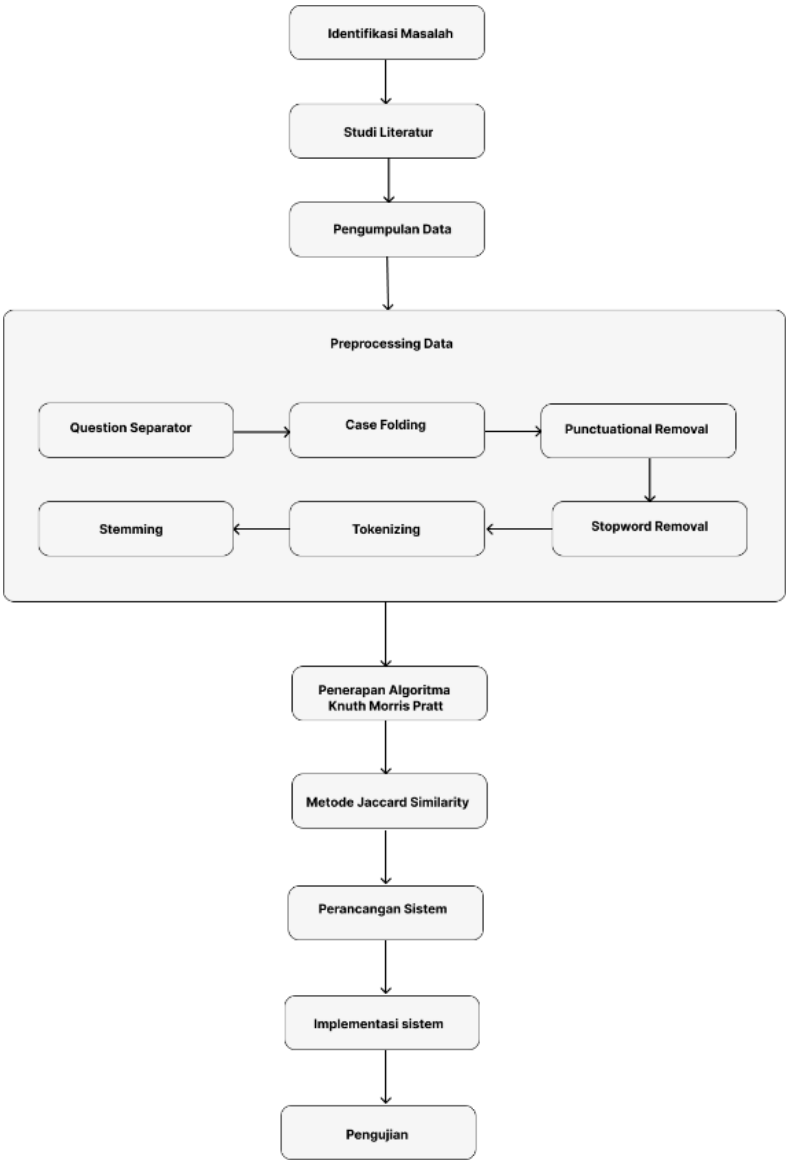
$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Sumber: Sapto Utomo (2022)

Keterangan :

J(A,B)	: Merupakan Jaccard Similarity, nilai yang menunjukkan tingkat kesamaan anatara dua himpunan A dan B.
|A|	: Banyaknya elemen (kata unik) dalam himpunan A
|B|	: Banyaknya elemen (kata unik) dalam himpunan B
|A∩B|	: Banyaknya elemen yang sama atau terjadi overlap antara himpunan A dan B,ini disebut sebagai himpunan irisan A dan B.
|A∪B|	: Banyaknya elemen gabungan antara dan B, yaitu semua elemen unik yang terdapat dalam kedua himpunan tanpa duplikasi.

Pada penelitian ini, prosedur yang akan dilakukan berdasarkan alur penelitian pada gambar 1.



Gambar 1. Alur Penelitian

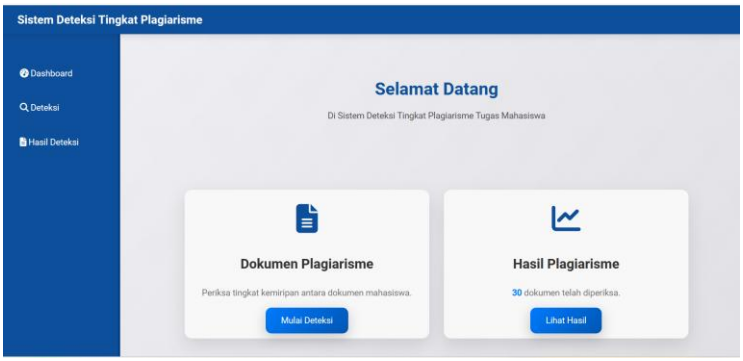
4. HASIL DAN PEMBAHASAN

Hasil

Hasil Penelitian ini merupakan Source Coude dan tampilan sistem yang telah dibangun.

a. Halaman Dashboard

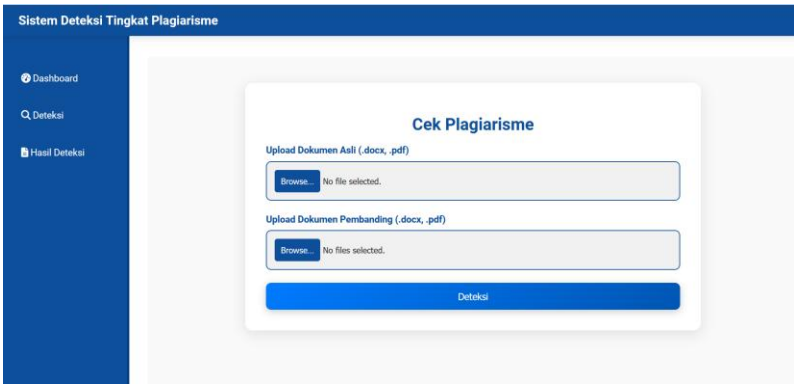
Halaman Dashboard merupakan halaman petama yang dapat dilihat pengguna pada saat mereka mengakses website. Tampilan halaman dashboard dapat dilihat pada gambar 2.



Gambar 2. Halaman Dashboard

b. Halaman Deteksi

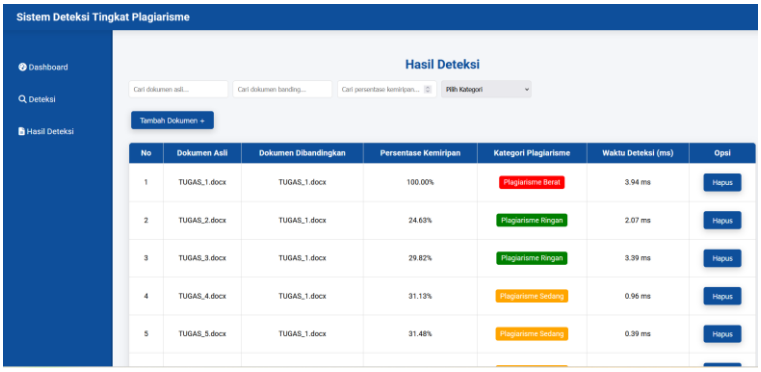
Halaman deteksi digunakan untuk menginputkan dokumen tugas mahasiswa yang ingin dicek persentase plagiarisme nya oleh pengguna. Tampilan halaman dashboard dapat dilihat pada gambar 3.



Gambar 3. Halaman Deteksi

c. Halaman Hasil Deteksi

Halaman hasil deteksi digunakan user untuk melihat semua persentase dari dokumen tugasmahasiswa yang sudah di deteksi plagiarisme nya. Tampilan halaman dashboard dapat dilihat pada gambar 4.



Gambar 4. Halaman Hasil Deteksi

d. *Preprocessing*

Text Preprocessing adalah proses yang sering digunakan untuk melakukan *text mining*. Tujuan dari *text preprocessing* adalah untuk mengembalikan teks menjadi bahasa yang alami. Adapun tahapan preprocessing meliputi *Question Seperator, Case folding, Punctuational Removal, Stopword Removal, Tokenizing, Steamming.*

- *Question Seperator*

```
question_pattern = re.compile(

r'^\d+\.\s|\b(apa itu|jelaskan|sebutkan|tuliskan|mengapa|bagaimana|berikan|hitung|buatlah|contohkan|urai|buktikan|bandingkan)\b',

re.IGNORECASE  # Mengabaikan huruf besar/kecil dalam pencocokan
```

“question_pattern = re.compile(“

Method digunakan untuk mendeteksi pola pertanyaan dalam sebuah teks. Tujuannya adalah untuk mengenali apakah suatu teks merupakan sebuah pertanyaan berdasarkan pola tertentu, seperti angka di awal diikuti dengan titik dan spasi (misalnya, "1. ") atau kata-kata tanya umum dalam bahasa Indonesia seperti "apa itu," "jelaskan," "sebutkan," "tuliskan," "mengapa," dan lainnya. Dengan menggunakan re.IGNORECASE, pencocokan dilakukan tanpa memperhatikan perbedaan huruf besar dan kecil, sehingga kata seperti "Mengapa" dan "mengapa" akan dianggap sama. Hal ini berguna dalam pemrosesan teks otomatis, seperti sistem deteksi pertanyaan dalam chatbot atau analisis dokumen.

```
# Membuat daftar baris yang cocok dengan pola pertanyaan

questions = [lines[i] for i in range(len(lines)) if
```

Method ini digunakan untuk mengekstrak baris-baris yang mengandung pertanyaan dari sebuah daftar teks. Dengan melakukan iterasi pada setiap elemen dalam daftar lines, method ini memeriksa apakah setiap baris cocok dengan pola pertanyaan yang telah ditentukan menggunakan question_pattern.search(). Jika sebuah baris sesuai dengan pola, maka baris tersebut akan dimasukkan ke dalam daftar questions. Dengan demikian, hanya baris-baris yang mengandung pertanyaan yang akan disimpan. Pendekatan ini berguna dalam berbagai aplikasi, seperti sistem ekstraksi pertanyaan dari dokumen yang perlu mengenali input berbentuk pertanyaan untuk memberikan respons yang sesuai.

```
    # Membuat daftar baris yang tidak cocok dengan pola pertanyaan (diasumsikan
                                sebagai jawaban)

            answers = [lines[i] for i in range(len(lines)) if not
                      question_pattern.search(lines[i])]
```

Method ini digunakan untuk mengekstrak baris-baris yang tidak mengandung pertanyaan dari sebuah daftar teks, yang diasumsikan sebagai jawaban. Dengan melakukan iterasi pada setiap elemen dalam daftar lines, method ini memeriksa apakah sebuah baris tidak cocok dengan pola pertanyaan yang telah ditentukan menggunakan question_pattern.search(). Jika suatu baris tidak mengandung pola pertanyaan, maka baris tersebut akan dimasukkan ke dalam daftar answers. Pendekatan ini berguna dalam berbagai aplikasi, seperti pemrosesan teks otomatis untuk memisahkan pertanyaan dari jawaban dalam dokumen, chatbot, atau sistem analisis teks lainnya.

```
  # Menggabungkan semua jawaban menjadi satu string, dipisahkan oleh newline (\n)

    # Mengembalikan teks jawaban dan daftar pertanyaan
```

“return '\n'.join(answers), questions”

Method ini digunakan untuk menggabungkan semua jawaban yang telah diekstrak ke dalam satu string yang dipisahkan oleh newline (\n). Proses ini dilakukan dengan menggunakan method join(), yang menghubungkan setiap elemen dalam daftar *answers* menjadi satu teks yang lebih mudah dibaca. Setelah itu, method mengembalikan dua nilai, yaitu teks jawaban yang telah digabungkan dan daftar pertanyaan yang sebelumnya telah diekstrak.

- *Case Folding*

```
        def preprocess_text(text: str, stop_words: List[str]) -> List[str]:

                                    # Case Folding
```

Method ini digunakan untuk mengubah semua karakter dalam string text menjadi huruf kecil. Tujuannya adalah untuk mengurangi keragaman atau variasi yang disebabkan oleh perbedaan antara huruf besar dan kecil.

- *Punctuational Removal*

```
  # Remove punctuation

    text_without_punctuation = case_folded_text.translate(str.maketrans('', '',
string.punctuation))
```

Method ini menghapus tanda baca seperti titik, koma, tanda tanya, dan sebagainya dari teks. Ini dilakukan dengan menggunakan translate() dan str.maketrans(), yang menggantikan setiap tanda baca dengan karakter kosong. Langkah ini penting untuk membersihkan teks agar analisis selanjutnya lebih akurat tanpa terpengaruh oleh tanda baca yang tidak relevan.

- *Stopword Removal*

```
# Split words and remove stopwords

filtered_words = [word for word in words if word not in stop_words]
```

Method ini menghapus kata-kata umum yang tidak memberikan informasi penting dalam analisis teks.

Kata-kata seperti "dan", "atau", "adalah", dan lainnya termasuk dalam stop_words, yang dihilangkan agar fokus analisis lebih pada kata-kata bermakna.

- *Tokenizing*

```
words = text_without_numbers.split()
```

Method ini memecah teks yang sudah tidak mengandung angka (jika ada) menjadi daftar kata-kata (token). Fungsi split() memisahkan teks berdasarkan spasi, sehingga setiap kata dalam teks akan menjadi elemen dalam daftar words untuk analisis lebih lanjut

- *Steamming*

```
# Apply stemming

stemmed_words = [sastrawi_stemmer.stem(word) for word in filtered_words]
```

“stemmed_words = [sastrawi_stemmer.stem(word) for word in filtered_words]”

Method ini mengubah kata-kata dalam daftar *filtered_words* menjadi bentuk dasarnya menggunakan algoritma Sastrawi. Proses *stemming* ini menghilangkan imbuhan pada kata, sehingga kata seperti "berlari" menjadi "lari". Langkah ini berguna untuk menyederhanakan teks dan meningkatkan kualitas analisis teks.

- Implementasi *Algoritma Knuth Morris Pratt*

Implementasi Algoritma *Knuth Morris Pratt* untuk digunakan pada perhitungan persentase kemiripan kata, dengan menggunakan bahasa pemrograman *Pyhton.*

```
    lps = [0] * len(pattern)
    length = 0
    i = 1
    while i < len(pattern):
        if pattern[i] ==
pattern[length]:
            length += 1
            lps[i] = length
            i += 1
        else:
if length != 0:
                length = lps[length - 1]
                  else:
                lps[i] = 0
                i += 1
       return lps
     def kmp_search_with_table(text: str,
pattern: str) -> List[int]:
    lps = compute_lps_array(pattern)
       i = 0
       j = 0
    matches = []
while i < len(text):
        if pattern[j] == text[i]:
            i += 1
            j += 1
        if j == len(pattern):
            matches.append(i - j)
            j = lps[j - 1]
```

Proses Algoritma *Knuth Morris pratt:*

”compute_lps_array(pattern: str) -> List[int]”

Fungsi ini digunakan untuk membangun array LPS (*Longest Prefix Suffix*), yang membantu menentukan posisi pola untuk melanjutkan pencocokan saat terjadi ketidaksesuaian karakter.

“kmp_search_with_table(text: str, pattern: str) -> List[int]”

Fungsi ini digunakan untuk mencari pola dalam teks menggunakan algoritma *Knuth Morris Pratt* dan mengembalikan daftar indeks posisi awal kecocokan pola dalam teks.

Hasil Pengujian

Pengujian ini dilakukan untuk memastikan bahwa algoritma *Knuth Morris Pratt* dapat berfungsi dengan baik dalam proses pencocokan string dan deteksi kemiripan dokumen. Langkah-langkah pengujian mencakup penerimaan input dokumen, pencarian substring menggunakan algoritma *Knuth Morris Pratt*, serta perhitungan tingkat kemiripan dokumen menggunakan *Jaccard Similarity*.

Berikut adalah hasil pengujian algoritma *Knuth Morris Pratt* yang diterapkan pada sistem deteksi plagiarisme.

Tabel 1. Hasil pengujian

No	Dokumen tugas	Dokumen tugas	Persentase Plagiarisme	Tingkat Plagiarisme
1	Tugas 1	Tugas 2	24.63%	Plagiarisme Ringan
2	Tugas 1	Tugas 3	29.82%	Plagiarisme Ringan
3	Tugas 1	Tugas 4	31.13%	Plagiarisme Sedang
4	Tugas 1	Tugas 5	31.48%	Plagiarisme Sedang
5	Tugas 1	Tugas 6	30.09%	Plagiarisme Sedang
6	Tugas 1	Tugas 7	30.91%	Plagiarisme Sedang
7	Tugas 1	Tugas 8	24.32%	Plagiarisme Ringan
8	Tugas 1	Tugas 9	29.46%	Plagiarisme Ringan
9	Tugas 1	Tugas 10	29.46%	Plagiarisme Ringan
10	Tugas 1	Tugas 11	25.66%	Plagiarisme Ringan
11	Tugas 1	Tugas 12	28.95%	Plagiarisme Ringan
12	Tugas 1	Tugas 13	26.05%	Plagiarisme Ringan
13	Tugas 1	Tugas 14	28.83%	Plagiarisme Ringan
14	Tugas 1	Tugas 15	30.43%	Plagiarisme Sedang
15	Tugas 1	Tugas 16	25.93%	Plagiarisme Ringan
16	Tugas 1	Tugas 17	23.70%	Plagiarisme Ringan
17	Tugas 1	Tugas 18	28.45%	Plagiarisme Ringan
18	Tugas 1	Tugas 19	27.03%	Plagiarisme Ringan
19	Tugas 1	Tugas 20	27.59%	Plagiarisme Ringan

20	Tugas 1	Tugas 21	21.74%	Plagiarisme Ringan
21	Tugas 1	Tugas 22	36.52%	Plagiarisme Sedang
22	Tugas 1	Tugas 23	27.59%	Plagiarisme Ringan
23	Tugas 1	Tugas 24	48.21%	Plagiarisme Sedang
24	Tugas 1	Tugas 25	32.20%	Plagiarisme Sedang
25	Tugas 1	Tugas 26	28.07%	Plagiarisme Ringan
26	Tugas 1	Tugas 27	27.35%	Plagiarisme Ringan
27	Tugas 1	Tugas 28	43.36%	Plagiarisme Sedang
28	Tugas 1	Tugas 29	43.37%	Plagiarisme Sedang
29	Tugas 1	Tugas 30	50.67%	Plagiarisme Sedang
30	Tugas 1	Tugas 31	47.22%	Plagiarisme Sedang

Hasil perhitungan manual berdasarkan hasil dari data testing sebelumnya dengan menggunakan perhitungan sistem dapat dilihat pada tabel 2.

Tabel 2. Akurasi manual dan sistem

Hasil persentase manual	Hasil persentase sistem
24,44%	24,63%

A1 = 24,44

A2 = 24,63

Akurasi $= \frac{24,44}{24,63} \times 100$

$= 0.9922 \times 100$

$= 99,22$

5. KESIMPULAN DAN SARAN

Berdasarkan penelitian yang telah dilakukan, dapat disimpulkan bahwa algoritma *Knuth Morris Pratt* efektif dalam mendeteksi tingkat plagiarisme pada tugas mahasiswa. Penelitian ini berhasil menerapkan algoritma *Knuth Morris Pratt* pada sistem deteksi plagiarisme, yang difokuskan pada tugas mahasiswa berbentuk esai dalam bahasa Indonesia berupa dokumen Word dan PDF tanpa gambar atau tabel. Sistem ini mampu mendeteksi tingkat kemiripan antar dokumen dengan rata-rata akurasi mencapai 98%, dan mengelompokkan hasil deteksi plagiarisme dalam kategori yang tepat, seperti Plagiarisme Ringan, Plagiarisme Sedang, dan Plagiarisme Berat. Sistem deteksi plagiarisme yang

dibangun mampu memberikan asil yang hampir sama dengan perhitungan manual. Selain itu, sistem ini juga mampu mengurangi kebutuhan pengecekan manual yang memakan waktu. Dengan demikian, penelitian ini berhasil mencapai tujuan untuk menerapkan algoritma *Knuth Morris Pratt* dalam deteksi plagiarisme pada tugas mahasiswa, memberikan manfaat berupa peningkatan akurasi dan kemampuan dalam mendeteksi plagiarisme di lingkungan akademik.

DAFTAR REFERENSI

Arafat, M., Trimarsiah, Y., Susantho, H., & Redaksi, D. (2022). Rancang bangun sistem informasi pemesanan online percetakan Sriwijaya Multi Grafika berbasis website. *JURNAL INTECH, 3*(2), 6–11.

Asnawi, M. F., & Abidin, Z. (2021). Sistem pendeteksi kemiripan pada proposal pengajuan tugas akhir menggunakan algoritma Rabin-Karp. *JAMI: Jurnal Ahli Muda Indonesia, 2*(1), 73–82. https://doi.org/10.46510/jami.v2i1.61

Dzaky Alfaro, S. (Tanpa tahun). Aplikasi algoritma KMP (Knuth-Morris-Pratt) dalam pengecekan plagiarisme. *[Online]* Tersedia di: https://www.google.com/url?sa=i&url=https%3A%2F%2Fggwp.id%2Fm

Ilham, M., & Mirza, A. H. (2020). Pengarsipan dokumen pada SMA Plus Negeri 17 Palembang. *[Online]* Tersedia di: https://journal-computing.org/index.php/journal-sea/index

Irmayanti. (2023). Perancangan sistem informasi penyewaan Thermoking di PT. Moderen Prima dengan Flask Python. *Jurnal Sistem dan Teknologi Informasi Cendekia, 1*(1), 19–28.

Islamiyati, D. S., & Fikri, A. (2022). Penerapan algoritma Knuth-Morris-Pratt dalam mendeteksi tingkat kemiripan judul skripsi berbasis web. *Journal of Information System Research (JOSH), 3*(2), 58–63. https://doi.org/10.47065/josh.v3i2.1168

Johar, A., Setiawan, S., Supratman, J. W., & Indonesia, A. (2019). Implementasi metode string matching untuk pencarian berita utama pada portal berita berbasis Android (Studi kasus: Harian Rakyat Bengkulu). *[Online]* Tersedia di: http://www.ejournal.unib.ac.id/index.php/pseudocode

Kambey, G. E. I., Sengkey, R., & Jacobus, A. (Tanpa tahun). Penerapan clustering pada aplikasi pendeteksi kemiripan dokumen teks Bahasa Indonesia. *Jurnal Teknik Informatika, 15*(2), 75–82.

Krisbiantoro, D., Rohim, S. F., & Santiko, I. (2021). Perbandingan algoritma N-gram dan algoritma Knuth Morris Pratt untuk mengukur tingkat akurasi plagiarisme pada dokumen abstrak skripsi berbasis website. *JITU: Journal Informatic Technology and Communication, 5*(1), 30–39. https://doi.org/10.36596/jitu.v5i1.390

Muliadi, M., Said, M. R., & Sofyan, E. (2021). Sistem pendeteksi kemiripan judul skripsi menggunakan algoritma Winnowing. *Journal Peqguruang: Conference Series*, *3*(1), 144. https://doi.org/10.35329/jp.v3i1.1261

Nugraha, D. (2021). Penerapan algoritma cosine similarity pada aplikasi bank soal.

Pratama, R. P., Faisal, M., & Hanani, A. (2019). Deteksi plagiarisme pada dokumen jurnal menggunakan metode cosine similarity. *SMARTICS Journal*, *5*(1), 22–26. https://doi.org/10.21067/smartics.v5i1.2848

Ramli, M. S., Cokrowibowo, S., & Rustan, M. F. (2021). Uji plagiarism pada tugas mahasiswa menggunakan algoritma Winnowing. *Journal of Applied Computer Science and Technology*, *2*(2), 108–112. https://doi.org/10.52158/jacost.v2i2.177

Rijoly, M. E., Pramudita, W., Tomasouw, B. P., & Leleury, Z. A. (2021). Perancangan sistem deteksi plagiarisme skripsi (judul dan abstrak) berbasis Matlab menggunakan algoritma Winnowing. *Tensor: Pure and Applied Mathematics Journal*, *2*(2), 67–76. https://doi.org/10.30598/tensorvol2iss2pp67-76

Siswanto, E., & Giap, Y. C. (2020). Dan cosine similarity untuk pendeteksi plagiarisme pada dokumen. *JURNAL ALGOR*, *1*(2). *[Online]* Tersedia di: https://jurnal.buddhidharma.ac.id/index.php/algor/index

Tugas, J., Berbasis, A., Daniel, W., & Susanti, W. (2021). Penerapan algoritma cosine similarity pada sistem pengajuan. *Prosiding Seminar Nasional Informatika (SENATIKA).*

Utomo, S., Subroto, I. M. I., & Riansyah, A. (2022). Deteksi plagiat tugas akhir dengan metode Jaccard similarity. *Jurnal Transistor Elektro dan Informatika (TRANSISTOR EI)*, *4*(2).