Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 10, Nomor 1, Februari 2025, pp 257-273
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

Analisis dan Deteksi Kemiripan Teks Berbasis Python dengan Algoritma Levenshtein Distance

Haris Sudarman[1], Yulhendri[2]
[1]Teknik Informatika, Fakultas Ilmu Komputer, Universitas Esa Unggul, Jakarta, Indonesia
[2]Sistem Informasi, Fakultas Ilmu Komputer, Universitas Esa Unggul, Jakarta, Indonesia
Email: harissudarman28@gmail.com[1], yulhendri@esaunggul.ac.id[2]

Abstract

Improvements in information technology have complicated the issue of plagiarism in academia, particularly in higher education. This project intends to create a plagiarism detection tool that examines the similarity of PDF files to established references utilizing the Levenshtein Distance method. The suggested system can effectively and precisely identify plagiarism by a series of procedures, such as text extraction, linguistic Preprocessing (tokenisation and stopword removal), and calculating the degree of similarity using the Levenshtein Distance method. Testing was carried out on various scenarios, including variations in document size and plagiarism levels. The experimental results show that the higher the level of similarity between the document and the reference, the longer the computing time required. However, this system can detect plagiarism with a fairly good success rate, even in documents with a low level of similarity. Black box testing confirms that this application can work according to the expected specifications, namely inputting PDF documents, detecting plagiarism, and providing accurate similarity percentage results. This research contributes to providing a plagiarism detection tool that can help maintain academic integrity, with the possibility of further development through integration with machine learning and user interface improvements.

Keywords: *Plagiarism, Levenshtein Distance, Plagiarism Detection*

Abstrak

Kemajuan teknologi informasi telah memperumit masalah plagiarisme di dunia akademis, khususnya pendidikan tinggi. Proyek ini bermaksud untuk membuat alat pendeteksi plagiarisme yang memeriksa kemiripan file PDF dengan referensi yang sudah ada menggunakan metode Levenshtein Distance. Sistem yang diusulkan dapat mengidentifikasi plagiarisme secara efektif dan tepat melalui serangkaian prosedur, seperti ekstraksi teks, Preprocessing linguistik (tokenisasi dan penghapusan stopword), serta menghitung derajat kemiripan menggunakan metode Levenshtein Distance. Pengujian dilakukan dengan berbagai skenario, termasuk variasi ukuran dokumen dan tingkat plagiarisme. Hasil percobaan menunjukkan bahwa semakin tinggi tingkat kemiripan antara dokumen dengan referensi maka waktu komputasi yang dibutuhkan semakin lama. Namun sistem ini dapat mendeteksi plagiarisme dengan tingkat keberhasilan yang cukup baik, bahkan pada dokumen yang tingkat kemiripannya rendah. Pengujian black box memastikan aplikasi ini dapat bekerja sesuai spesifikasi yang diharapkan yaitu penginputan dokumen PDF, mendeteksi plagiarisme, dan memberikan hasil persentase kemiripan yang akurat. Penelitian ini berkontribusi dalam menyediakan alat pendeteksi plagiarisme yang dapat membantu menjaga integritas akademik, dengan kemungkinan untuk dikembangkan lebih lanjut melalui integrasi dengan pembelajaran mesin dan peningkatan antarmuka pengguna.

Kata kunci: *Plagiarisme, Jarak Levenshtein, Deteksi Plagiarisme*

1. PENDAHULUAN

Menurut informasi dari Pangkalan Data Pendidikan Tinggi Kementerian Riset, Teknologi, dan Pendidikan Tinggi (PDDIKTI), terdapat 31.399 program studi yang ditawarkan oleh 4.523 perguruan tinggi di Indonesia [1]. Kompleksitas sistem pendidikan tinggi di Indonesia tercermin dari beragamnya universitas dan

pilihan gelar yang tersedia. Dalam kaitan ini, makalah akademik—seperti skripsi sarjana (S1) dan tugas akhir tingkat diploma (D3)—menjadi syarat kelulusan. Makalah akademis ini membantu mahasiswa mengembangkan keterampilan penelitian, yang penting bagi pendidikan tinggi di Indonesia, selain menunjukkan kemahiran akademis mereka di bidang studinya.

Oleh karena itu, meskipun jumlah perguruan tinggi dan program studi yang tersedia sangat banyak, namun penulisan ilmiah tetap menjadi komponen fundamental dalam proses penyelesaian studi dan pencapaian kompetensi akademik yang diharapkan sebagai syarat penyelesaian jenjang D3 dan S1. Melalui skripsi mahasiswa dituntut berpikir sistematis. Kemajuan teknologi tentunya diharapkan akan semakin memudahkan dalam menyelesaikan skripsi mahasiswa pada jenjang D3 dan S1 [2], Namun dengan adanya kemajuan teknologi tersebut seolah menjadi pedang bermata dua yang menjadikan alat informasi tidak hanya sekedar alat informasi saja. alat untuk membantu mahasiswa memperoleh referensi untuk skripsinya, namun juga menjadikannya sarana untuk menjiplak karya orang lain. Plagiarisme adalah tindakan mengambil gagasan orang lain, mengambil tulisan orang lain, dan mengambil naskah secara keseluruhan serta mengakuinya sebagai milik pribadi [3] sehingga plagiarisme dianggap sebagai tindakan tidak etis yang dapat merugikan kredibilitas akademik dan profesional. Tindakan plagiarisme banyak terjadi di institusi sekolah dan terus meningkat, tindakan plagiarisme dianggap biasa dilakukan oleh siswa, data yang disajikan dalam buku “Remaja: Mengubah Keyakinan dan Perilaku” menunjukkan bahwa 58,3% siswa pernah melakukan plagiarisme pada tugasnya dan dalam kurun waktu 20 tahun. tahun telah meningkat menjadi 97,5% [4]. Dengan permasalahan tersebut diperlukan upaya untuk dapat mencegah maraknya plagiarisme agar tidak terus menjamur dalam kehidupan masyarakat Indonesia. Upaya ini harus dilakukan sedini mungkin, khususnya pada sistem pendidikan dan moral masyarakat. Salah satu upaya untuk mencegah maraknya plagiarisme dalam kehidupan masyarakat adalah dengan melakukan penelitian dan memberikan inovasi teknologi yang dapat membantu mendeteksi praktik plagiarisme secara otomatis, salah satunya dengan membuat aplikasi pendeteksi plagiarisme. Untuk bisa mendapatkan suatu aplikasi yang mampu menjawab permasalahan tersebut maka diperlukan metode kecerdasan buatan.

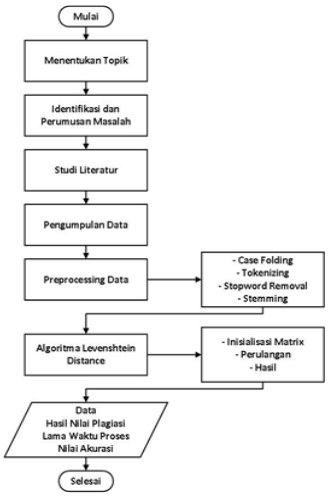
Berbagai penelitian telah dilakukan mengenai kecerdasan buatan yang digunakan dalam pendeteksian plagiarisme pada tahun 2017 hingga tahun 2023, pada tahun 2017 penelitian tersebut dilakukan oleh Alamsyah, penelitian ini menggunakan metode Winnowing dan Algoritma Rabin Karp, penelitian ini berhasil membandingkan kedua metode yang digunakan, kedua metode ini mempunyai kesamaan yaitu penggunaan hash yang menyebabkan ketergantungan terhadap hash dan berpotensi menghilangkan beberapa kesamaan namun memberikan hasil yang cukup memuaskan dengan tingkat kepuasan sebesar 32%. kesamaan [5]. 2023 yang berjudul Implementation of the Perceptual Hash Method for Plagiarism Detection for Software Modeling Coursework [6] menyatakan bahwa metode perceptual hash mampu mendeteksi plagiarisme dengan tingkat keberhasilan yang tinggi sebesar 70,7% dengan nilai ambang batas sebesar 75%

untuk tingkat kesamaan. Penelitian lain yang dilakukan M.Febbiawan, dkk. Tentang Sistem Deteksi Dini Plagiarisme Menggunakan Algoritma Levenshtein Distance [7], Penelitian ini merupakan proposal skripsi dan makalah publikasi. Pada uji kecepatan penghitungan, dokumen sumber memiliki jumlah kata sebanyak 4405 dengan 3 dokumen pembanding yang memiliki jumlah kata sebanyak 13465 sehingga menghasilkan durasi penghitungan selama 3,57 detik. Selain itu pada tahun 2016 dengan metode yang sama juga dilakukan penelitian mengenai Penerapan Pendeteksi Kemiripan Isi Teks Dokumen Menggunakan Metode Levenshtein Distance [8] dengan hasil nilai kemiripan yang tinggi yaitu diatas 77% hingga 100 % untuk dokumen dengan kemiripan tinggi. Sedangkan untuk dokumen yang tingkat kemiripannya rendah atau tidak ada plagiarisme maka nilai kemiripan yang dihasilkan dibawah 40%.

Menerapkan teknik Levenshtein Distance untuk mengidentifikasi plagiarisme dalam teks adalah tujuan dari penelitian ini. Secara khusus, penelitian ini bermaksud untuk mengevaluasi kecepatan komputasi algoritma Levenshtein Distance dalam mengidentifikasi plagiarisme pada dokumen dengan ukuran berbeda, mengukur plagiarisme dan kesamaan dokumen untuk mendeteksi plagiarisme menggunakan nilai kesamaan yang dihasilkan algoritma ini, dan mengevaluasi kemanjuran praktis algoritma sebagai pendeteksi plagiarisme yang dapat digunakan untuk mengidentifikasi kesamaan isi dokumen dengan cepat dan tepat [9].

2. METODOLOGI PENELITIAN

Dalam penelitian ini terdapat beberapa tahapan dan tes dalam melaksanakan proses penelitian. Tahapan yang dilakukan meliputi pengumpulan data, *Preprocessing* teks, pembuatan algoritma jarak Levenshtein, dan pengujian black box dalam bentuk aplikasi [10]. Pengujian black box ini dilakukan untuk mengetahui apakah aplikasi berjalan dengan baik dan mampu menampilkan persentase plagiarisme serta mendeteksi plagiarisme. Berikut langkah-langkah alur penelitian yang digambarkan melalui diagram alir pada Gambar 1.



Gambar 1. Alur Penelitian

Penelitian ini menggunakan pengumpulan data PDF link PDF yang dikumpulkan dalam bentuk JSON dengan tujuan untuk memudahkan akses dari berbagai sumber jurnal dan perpustakaan. Jadi dataset yang digunakan adalah kumpulan link PDF berformat .json dalam bentuk API (Application Programming Interface). Setelah data terkumpul maka akan dilakukan *Preprocessing*. Pada data *Preprocessing* akan terjadi perubahan seluruh data yang akan diubah menjadi bentuk string, dan akan dilakukan beberapa jenis *Preprocessing* data seperti yang terdiri dari casefolding, tokenizing, stopword, stemming. Selanjutnya ketika data sudah di preprocess maka data tersebut akan dibawa untuk melakukan proses selanjutnya yaitu algoritma Levenshtein distance yang selanjutnya akan menghasilkan matriks yang membandingkan data yang diupload dengan data API yang merupakan kumpulan jurnal yang telah dikumpulkan. proses pengumpulan data di awal.

2.1. Pengolahan Data

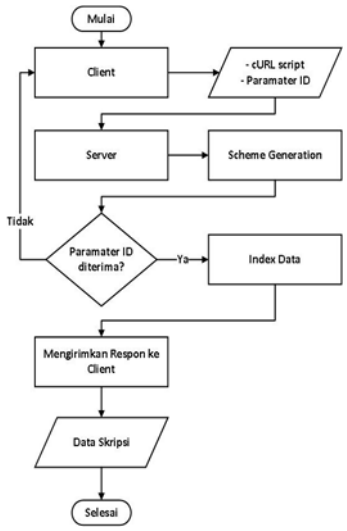
Pengolahan data dilakukan melalui beberapa langkah yaitu: input data, data input berupa PDF yang ingin diketahui persentase plagiatnya, kemudian akan dilakukan konversi PDF ke teks, sehingga dapat dilanjutkan ke tahap *Preprocessing* data kemudian dilanjutkan ke tahap pembuatan algoritma jarak Levenshtein untuk kemudian diperoleh hasil plagiarisme, akurasi waktu proses dan link dengan kemiripan paling tinggi [11].

2.1.1. Pengumpulan Data

Studi ini juga menggunakan metode cURL sebagai instrumen kunci dalam proses pengumpulan kumpulan data. cUR, sebuah utilitas baris perintah yang fleksibel, dimaksudkan untuk mengirim data ke atau menerima data dari server dengan mengakomodasi beberapa protokol seperti HTTP, HTTPS, FTP, dan lainnya. Dalam penelitian ini, cURL digunakan untuk berinteraksi dengan Application Programming Interface (API) yang menyediakan data terkait sesuai penelitian. persyaratan.. API bertindak sebagai perantara atau jembatan komunikasi antara aplikasi pengguna dan server, memungkinkan pengambilan data secara otomatis, cepat dan efisien. Untuk mendukung proses analisis data, penelitian ini menerapkan mekanisme dimana hasil setiap permintaan API yang berhasil dikembalikan oleh server disimpan dalam format file JSON (JavaScript Object Notation). File JSON ini memiliki struktur data yang terorganisir dengan baik, sehingga mudah dibaca dan digunakan kembali dalam berbagai proses lanjutan, seperti analisis, validasi, dan pemrosesan lebih lanjut. Format JSON juga memungkinkan data diakses secara fleksibel, karena dataset yang disimpan tidak memerlukan pengambilan ulang dari API, sehingga mempercepat proses kerja dan menghemat sumber daya. Pendekatan penyimpanan ini memberikan sejumlah keuntungan strategis. Pertama, data yang telah dikumpulkan tetap aman dan dapat diakses kapan saja, meskipun ada perubahan pada API endpoint atau pembatasan akses di kemudian hari. Kedua, penggunaan file JSON membantu menjaga konsistensi kumpulan data, yang sangat penting untuk memastikan validitas dan keakuratan hasil penelitian.

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 10, Nomor 1, Februari 2025, pp 257-273
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

Dengan strategi ini, penelitian berhasil menghemat waktu, mengurangi risiko kehilangan data, dan meningkatkan efisiensi proses pengembangan sistem secara keseluruhan. Pendekatan yang diterapkan dalam penelitian ini tidak hanya memperkuat ketahanan data, namun juga memastikan sistem dapat terus berkembang dengan dukungan data yang andal, terstruktur, dan dapat digunakan kembali kapan pun diperlukan. Berikut alur pengumpulan data yang digambarkan pada Gambar 2:



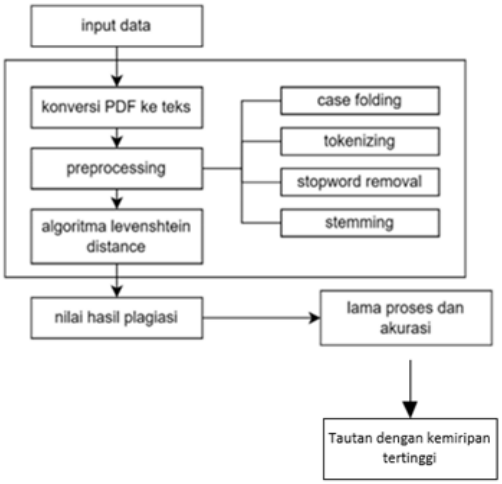
Gambar 2. Alur Pengumpulan Data

Proses pengumpulan data dimulai dari sisi client, dengan mengajukan permintaan data menggunakan script cURL yang dilengkapi dengan parameter ID. Script diteruskan ke server untuk melalui tahap pembuatan skema yaitu pengecekan skema akses. Jika parameter ID valid maka server akan memproses data melalui tahap pengindeksan untuk menyiapkan data yang diminta. Jika tidak, jika parameter ID tidak valid, permintaan akan ditolak, dan klien harus mengirimkan ulang. Data yang berhasil diproses dikirim kembali ke klien dalam bentuk respon cURL, yang kemudian diubah menjadi file berformat JSON agar mudah digunakan dan dianalisis lebih lanjut.

2.1.2. Data Pemrosesan Awal

Preprocessing teks merupakan serangkaian langkah untuk menyiapkan teks mentah agar siap untuk dianalisis. Proses ini sangat krusial dalam Natural Language Processing (NLP), dengan tujuan utama mengurangi elemen yang tidak relevan dan menyederhanakan teks[12], sehingga algoritma atau analisis lanjutan dapat berjalan lebih efisien. Dalam penelitian ini, seluruh tahapan prapemrosesan teks dilakukan dengan menggunakan pustaka Natural Language Toolkit (NLTK) untuk menyederhanakan pemrosesan data teks[13][14]. Dalam penelitian ini dilakukan serangkaian *Preprocessing* data seperti yang digambarkan pada Gambar 3:

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 10, Nomor 1, Februari 2025, pp 257-273
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik



Gambar 3. Data Pemrosesan Awal

Preprocessing data pada penelitian ini memiliki beberapa tahapan diantaranya casefolding, tokenizing, stopword removal dan stemming. Kegunaan tahapan tersebut adalah sebagai berikut:

a. *Lipat Kasus*

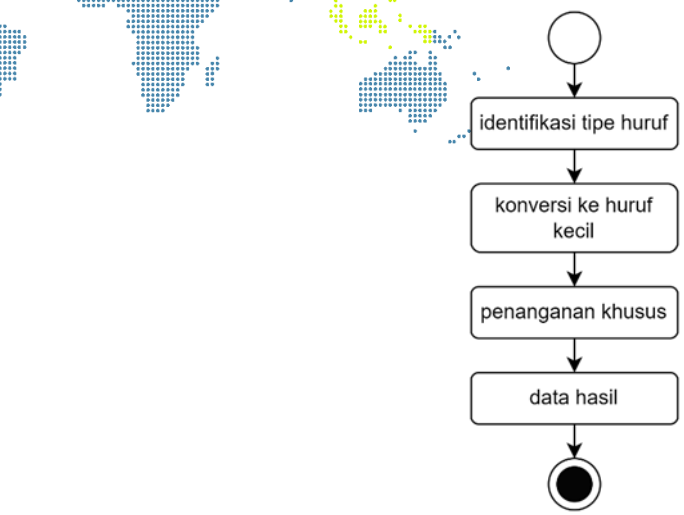
Case Folding merupakan salah satu tahapan dalam proses *Preprocessing* teks yang berfungsi mengubah seluruh huruf pada dokumen menjadi huruf kecil. Tujuan utama dari casefolding adalah untuk menyamakan format teks agar tidak terjadi perbedaan antara huruf besar dan kecil yang dapat mempengaruhi hasil analisis atau pengolahan teks. Hal ini penting dalam analisis teks, seperti pencocokan kata, karena sistem komputer akan memperlakukan huruf besar dan kecil sebagai entitas yang berbeda. Misalnya, kata "Data", "data", dan "DATA" akan dianggap sebagai tiga entitas berbeda jika pelipatan kasus tidak diterapkan. Dengan menerapkan case flip, seluruh kata tersebut akan diubah ke dalam format yang konsisten yaitu “data”. Hal ini membantu mengurangi variasi data dan meningkatkan keakuratan proses analisis seperti klasifikasi teks, pengelompokan, atau pencarian kata kunci.[15][16]. Contoh penerapan case lipat:

Tabel 1. Lipat Kasus

Teks Nyata	Setelah Pelipatan Kasus
Pemrosesan Bahasa Alami itu MENYENANGKAN dan Menantang!	pemrosesan bahasa alami itu menyenangkan dan menantang!

Pada contoh ini, semua huruf kapital diubah menjadi huruf kecil, sehingga teks menjadi lebih seragam dan siap untuk tahap *Preprocessing* teks selanjutnya, seperti tokenisasi atau penghapusan stopword. Untuk mendapatkan data hasil *Preprocessing* dari casefolding diperlukan alur sebagai berikut seperti pada Gambar 4:

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 10, Nomor 1, Februari 2025, pp 257-273
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik



Gambar 4. Alur Pelipatan Kasus

Pada alur case lipat digambarkan perlu dilakukan identifikasi terlebih dahulu jenis huruf apakah huruf besar atau kecil, kemudian diubah menjadi huruf kecil, setelah itu pengolahan data dengan case lipat akan dilanjutkan ke tahap penanganan khusus, tahapan ini merupakan tahapan khusus untuk simbol. Simbol aritmatika atau bahasa tertentu seperti karakter "ß" (Eszett) dalam bahasa Jerman dapat diubah menjadi "ss".

b. *Tokenisasi*

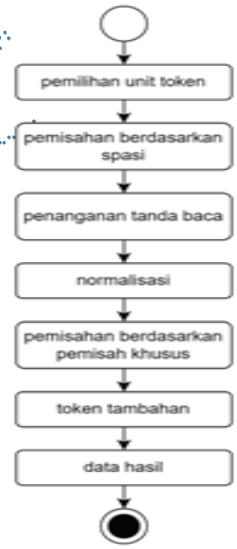
Tokenisasi adalah suatu proses dalam natural bahasa pemrosesan (NLP) yang memecah teks atau kalimat menjadi unit-unit kecil yang disebut token. Token ini dapat berupa kata, tanda baca, atau entitas penting lainnya dalam teks.[17][18] Contoh tokenisasi:

Tabel 2. Tokenisasi

Teks Nyata	**Setelah Tokenisasi**
Saya kuliah di Universitas Esa Unggul.	["Saya", "belajar", "di", "universitas", "Esa", “Luar Biasa”].
Halo, apa kabarmu?	["Halo, apa kabar", "?"

Tokenisasi penting karena memungkinkan sistem komputer menganalisis dan memproses teks dengan lebih mudah, seperti dalam analisis sentimen atau pemahaman konteks. Berikut langkah-langkah algoritma yang digunakan untuk *Preprocessing* data Tokenizing yang digambarkan pada gambar 5

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 10, Nomor 1, Februari 2025, pp 257-273
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik



Gambar 5. Tokenisasi

Pada proses Tokenizing, dipilih unit token yang digunakan untuk menentukan unit yang diinginkan sebagai token, dapat berupa kata, frase, simbol, atau aturan khusus lainnya tergantung kebutuhan aplikasi. Selanjutnya dilakukan pemisahan berdasarkan spasi. Tokenisasi sering kali dimulai dengan memisahkan string berdasarkan spasi atau karakter spasi. lainnya.

c. *Penghapusan Kata Penghenti*

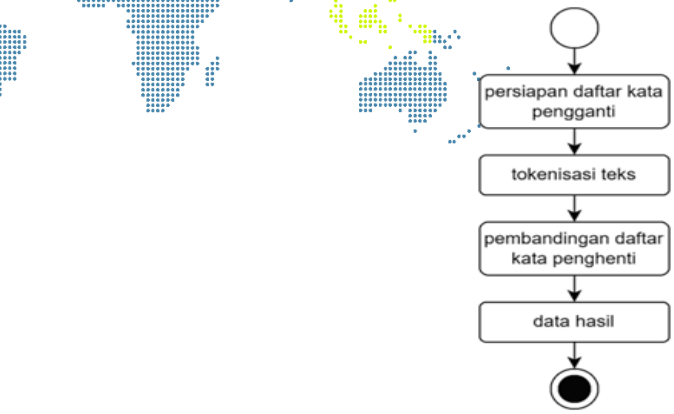
Stopword removal merupakan salah satu teknik dalam pemrosesan bahasa alami (NLP) yang digunakan untuk menghilangkan kata-kata yang dianggap tidak memiliki arti penting atau kontribusi signifikan terhadap analisis teks. Kata-kata yang termasuk dalam kategori stopwords biasanya merupakan kata-kata umum seperti "dan", "atau", "adalah", "dalam", "ke", "dengan", dll, yang sering muncul di hampir semua teks, namun tidak tidak memberikan informasi spesifik untuk analisis lebih lanjut. Tujuan dari penghapusan stopword adalah untuk mengurangi kompleksitas teks dan fokus pada kata-kata yang lebih relevan sehingga dapat memberikan informasi yang lebih bermakna [19][20]. Misalnya saja lihat contoh Tabel 3:

Tabel 3. Penghapusan Kata Penghenti

Teks Nyata	Setelah Penghapusan Stopword
Saya sedang belajar di universitas	"belajar", "universitas"

Dengan menghilangkan stopwords, proses analisis teks menjadi lebih efisien dan lebih fokus pada kata-kata yang dapat membantu dalam tugas-tugas seperti klasifikasi teks, pengambilan informasi, atau analisis sentimen.
Berikut alur yang digunakan untuk membuat program stopword removal seperti pada Gambar 6:

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 10, Nomor 1, Februari 2025, pp 257-273
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik



Gambar 6. Penghapusan Kata Penghenti

Mengumpulkan istilah-istilah umum yang tidak memberikan banyak konteks adalah langkah pertama dalam membuat daftar stop word, yang selanjutnya akan dihilangkan dari teks. Teks tersebut kemudian dibagi menjadi bagian-bagian yang lebih kecil, seperti kata atau frasa, menggunakan metode tokenisasi. Setiap token dibandingkan dengan daftar kata-kata berhenti yang telah dibuat sebelumnya setelah teks diberi token. Token dihilangkan jika muncul dalam daftar kata-kata berhenti, tetapi disimpan untuk pemeriksaan tambahan jika tidak. Prosedur ini membantu mempersempit analisis menjadi kata-kata yang lebih relevan dan bermakna.

d. *berasal*

Stemming adalah prosedur dalam pemrosesan bahasa alami (NLP) yang berupaya memotong kata menjadi bentuk dasar atau kata dasar. Proses ini menghilangkan imbuhan atau sufiks pada kata sehingga kata tersebut kembali ke bentuk dasarnya yang lebih sederhana. Misalnya kata “run”, “run”, dan “runner” dapat diolah sehingga menghasilkan bentuk dasar yang sama yaitu “run”. Stemming penting karena dalam analisis teks, bentuk kata yang berbeda sering kali memiliki arti yang serupa. Misalnya kata “cinta”, “cinta”, dan “dicintai” pada dasarnya mempunyai makna inti yang sama, sehingga dengan melakukan stemming, sistem dapat lebih fokus pada makna dasar kata tersebut dibandingkan berbagai bentuknya. Teknik ini membantu dalam tugas-tugas seperti klasifikasi teks, pengambilan informasi, dan analisis sentimen, di mana fokus utamanya adalah mengidentifikasi makna inti kata-kata dalam teks [21][22]. Berikut alur yang digunakan untuk membuat program stemming seperti pada gambar 7:

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 10, Nomor 1, Februari 2025, pp 257-273
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

Gambar 7. Berasal

Proses *stemming* diawali dengan penyusunan daftar kata dasar yang berisi kata-kata yang telah diidentifikasi sebelumnya dan dapat membantu menentukan imbuhan mana yang sebaiknya dihilangkan dari suatu kata. Selanjutnya, teks tersebut diberi token menjadi unit-unit yang lebih kecil seperti kata atau frasa. Setelah itu, diterapkan aturan atau metode stemming untuk menghilangkan imbuhan, misalnya prefiks atau sufiks, pada setiap kata. Misalnya saja lihat Tabel 4.

Tabel 4. Berasal

Teks Nyata	**Setelah Steming**
"Berlari"	"Berlari"
"Lebih baik"	"Lebih baik"

2.2. *Penerapan Algoritma Jarak Levenshtein*

Edit jarak adalah nama yang lebih populer untuk jarak Levenshtein. Menemukan jumlah mutasi titik terkecil yang diperlukan untuk mengubah satu string menjadi string lainnya adalah gagasan di balik jarak Levenshtein. Penyisipan, substitusi, dan penghapusan adalah contoh mutasi titik [9]. Jarak Levenshtein merupakan matriks string yang berfungsi untuk mengukur selisih atau jarak antara dua string. Nilai jarak ini ditentukan oleh banyaknya operasi perubahan yang digunakan untuk melakukan transformasi dari satu string ke string lainnya. Operasi ini adalah penyisipan, penghapusan atau substitusi. Algoritma Levenshtein Distance dapat digunakan untuk mendeteksi kemiripan antara dua string yang berpotensi melakukan tindakan plagiarisme [10]. Algoritma Levenshtein Distance bekerja dari sisi kiri atas sebuah array, pada dua matriks yang berisi string A dan string B. Berikut algoritma Levenshtein Distance untuk mendapatkan nilai jarak [11].

Levenshtein Distance menghitung bobot kemiripan setelah mendapatkan nilai jarak dari kedua dokumen yang dibandingkan. Kemudian digunakan persamaan untuk menentukan bobot kemiripan yaitu:

Bobot kesamaan = = $\left(1 - \frac{d\,[m,n]}{Maks\ (s,T)}\right) * 100\%$ (1)

Dimana d[m,n] adalah nilai jarak yang terletak pada baris ke-m dan kolom ke-n, kemudian s adalah panjang string awal, kemudian T adalah panjang string target, dan Max(s,T) adalah panjang string target. panjang string terbesar antara string awal dan string target.

2.3. *Perhitungan Akurasi dan Persentase Tertinggi*

Karena hasil akhir aplikasi akan ditampilkan, penting untuk mengetahui keakuratan dan % plagiarisme dalam penelitian ini. Teknik Levenshtein Distance digunakan untuk menilai persentase plagiarisme guna memastikan tingkat kemiripan antara teks referensi dengan konten yang dikirimkan. Dengan mengubah satu teks menjadi teks lain melalui proses penyisipan, penghapusan, dan penggantian karakter, metode ini menghitung jarak edit antara dua teks. Fungsi Levenshtein.ratio menghasilkan nilai kesamaan yang memiliki rentang 0 hingga 1. Nilai ini kemudian diubah menjadi persentase menggunakan rumus berikut:

$$\%\text{plagiarisme} = Levenshtein.ratio(teks1, teks2) * 100 \quad (2)$$

Hasil ini mewakili sejauh mana kemiripan teks yang diuji dengan teks perbandingan. Nilai 100% menunjukkan bahwa teks yang diuji identik dengan dokumen pembanding. Sedangkan akurasi dalam konteks ini diperhitungkan sebagai pelengkap persentase plagiarisme. Semakin kecil nilai persentase plagiarisme maka semakin tinggi tingkat keunikan teks yang diuji. Rumus yang digunakan adalah sebagai berikut:

$$\text{Akurasi} = 100 - persentase\ plagiarisme \quad (3)$$

Nilai akurasi yang tinggi menunjukkan bahwa teks yang diuji mempunyai tingkat keunikan yang tinggi dan tidak banyak mengandung kemiripan dengan teks referensi. Keakuratan ini penting untuk menilai keabsahan suatu teks sebagai karya orisinal.

2.4. Peningkatan Akurasi Jarak Levenshtein melalui Teknik Pemrosesan Awal Teks

Keakuratan Levenshtein Distance yang dikombinasikan dengan pelipatan kasus, tokenisasi, penghapusan stopword, dan stemming dapat bervariasi bergantung pada kumpulan data dan aplikasi tertentu[23][24][25][26][27][28]. Namun berdasarkan kinerja umum: Rentang Akurasi: Akurasi Dasar: Jarak Levenshtein sendiri memiliki rentang akurasi 40-70%, bergantung pada kasus penggunaan. Dengan Peningkatan Pra-pemrosesan:
Pelipatan Kasus: Mengurangi variabilitas yang disebabkan oleh perbedaan kasus, meningkatkan akurasi sebesar 5-10%.

Tokenisasi, Penghapusan Stopword, dan Stemming: Teknik-teknik ini semakin menyempurnakan masukan, berpotensi menambah peningkatan hasil sebesar 10-15%. Perkiraan Akurasi:
Dengan kombinasi teknik ini, keakuratan Levenshtein Distance dapat dicapai **60-85%,** tergantung pada kompleksitas kumpulan data dan tugas kesamaan teks.

Namun rentang ini masih kurang pemahaman semantiknya, karena Levenshtein Distance hanya berfokus pada edit jarak tanpa mempertimbangkan makna atau konteks.

2.5. *Skenario Penelitian*

Tujuan dari proyek ini adalah untuk menciptakan alat pendeteksi plagiarisme yang mampu menganalisis file PDF dan menilai kemiripannya dengan dokumen asli melalui penerapan algoritma Jarak Levenshtein. Kajian ini didasari oleh tingginya permintaan akan teknologi otomatis yang mampu mengidentifikasi plagiarisme secara akurat dan meningkatkan ketersediaan teknologi digital. sumber daya. Penelitian dilakukan melalui beberapa tahapan yaitu pengumpulan dataset berupa dokumen asli dan dokumen dengan tingkat modifikasi plagiarisme yang bervariasi (0%, 25%, 50%, 75%, dan 100%), pengembangan sistem dengan tahap *Preprocessing* teks. seperti casefolding, tokenization, stopword removal, dan stemming, serta implementasi algoritma Levenshtein Distance[29].

Sistem ini diuji melalui beberapa skenario seperti pengujian akurasi dan waktu pemrosesan berdasarkan ukuran dokumen (kecil, sedang, besar). Variabel yang diuji meliputi variabel independen (ukuran dokumen dan jumlah dokumen referensi), variabel terkontrol (algoritma, teks tahapan prapemrosesan, dan spesifikasi perangkat keras), serta variabel terikat (tingkat deteksi akurasi dan waktu pemrosesan sistem). Data yang dikumpulkan meliputi persentase terdeteksinya plagiarisme, waktu pemrosesan sistem, dan tingkat akurasi berdasarkan perbandingan hasil sistem dengan evaluasi manual. Analisis dilakukan secara deskriptif dan statistik untuk mengevaluasi keandalan dan efisiensi sistem. Penelitian ini diharapkan menghasilkan sebuah aplikasi pendeteksi plagiarisme yang memiliki tingkat keberhasilan di atas 85% dan dapat memberikan kontribusi nyata dalam menyediakan alat pendeteksi plagiarisme di berbagai bidang pendidikan dan profesi.

2.6. *Pengujian Kotak Hitam*

Pengujian black box merupakan metode evaluasi perangkat lunak yang menguji fungsi sistem berdasarkan spesifikasi tanpa memperhatikan implementasi internalnya. Dalam konteks sistem pendeteksi plagiarisme berbasis Levenshtein Distance, pengujian ini memastikan bahwa masukan berupa dokumen PDF diproses dengan benar melalui ekstraksi teks, prapemrosesan linguistik, dan penghitungan persentase plagiarisme, sehingga menghasilkan keluaran berupa persentase kesamaan, akurasi, dan waktu pemrosesan yang memenuhi persyaratan. Pengujian dilakukan pada berbagai skenario, seperti validasi masukan, pengujian fungsionalitas metode ekstraksi dan algoritma deteksi, keandalan hasil, dan penanganan kesalahan, untuk memastikan sistem bekerja sesuai spesifikasi dan memberikan hasil yang konsisten [30].

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 10, Nomor 1, Februari 2025, pp 257-273
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

2.7. *Penerapan Web*

Web Deployment adalah proses menempatkan web aplikasi atau sistem perangkat lunak di lingkungan yang dapat diakses oleh pengguna lain Render: Cloud Application Platform adalah cloud yang saya pilih untuk aplikasi saya Render memungkinkan pengembang dengan mudah menjalankan dan menyebarkan aplikasi web, API, backend layanan, dan database tanpa perlu mengelola infrastruktur secara manual.

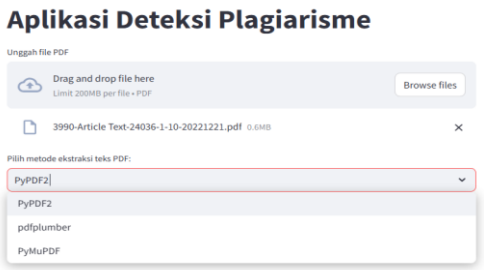
3. HASIL DAN PEMBAHASAN

3.1. *Hasil Implementasi*

Aplikasi *online* untuk deteksi plagiarisme dikembangkan sebagai bagian dari proyek untuk membangun dan menjalankan sistem deteksi plagiarisme. Dengan bantuan alat ini, pengguna dapat mengunggah file PDF, yang kemudian dibandingkan dengan database dokumen PDF yang disimpan dalam API. Dengan menggunakan data berformat JSON dari repositori, proses perbandingan mempermudah identifikasi dan investigasi potensi situasi plagiarisme bagi akademisi secara akurat.

a. Tampilan Awal

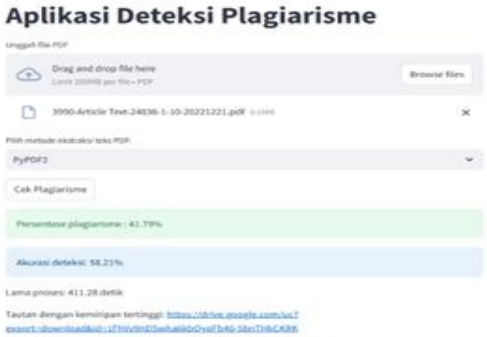
Berikut tampilan awal yang digambarkan seperti pada gambar:



Gambar 8. Tampilan Awal Implementasi

b. Tampilan Deteksi Plagiarisme

Berikut tampilan Deteksi Plagiarisme saat digunakan:



Gambar 9. Tampilan Deteksi Plagiarisme

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 10, Nomor 1, Februari 2025, pp 257-273
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

3.2. *Hasil Eksperimen*

Dalam penelitian ini dilakukan percobaan terhadap 3 dokumen yang menghasilkan hasil sebagai berikut:

Tabel 5. Hasil Eksperimen

Tingkat Kesamaan	Kecepatan Komputasi	Persentase Hasil Plagiarisme
100%	877,58 Detik	100%
>50%	773,56 Detik	80%

Setelah dilakukan perbandingan tersebut diketahui bahwa setelah dilakukan perbandingan diperoleh bahwa semakin tinggi tingkat kemiripan antara dokumen yang diuji dengan dokumen sumber maka semakin lama waktu komputasi yang diperlukan dan semakin tinggi persentase plagiarisme yang terdeteksi. . Untuk dokumen dengan tingkat kemiripan 100%, waktu komputasi mencapai 877,58 detik dengan hasil plagiarisme 100%. Sebaliknya, pada dokumen dengan kemiripan lebih dari 50%, waktu komputasinya berkurang menjadi 683,56 detik dengan terdeteksi adanya plagiarisme sekitar 80%, dan pada dokumen dengan kemiripan kurang dari 50%, waktu komputasinya menjadi 411,28 detik dengan persentase plagiarisme terdeteksi sekitar 41%. Dibandingkan dengan penelitian sebelumnya, temuan ini menunjukkan pola yang konsisten antara tingkat kemiripan, waktu komputasi, dan persentase plagiarisme. Penelitian sebelumnya mungkin menunjukkan tren serupa, namun dalam penelitian ini, pengurangan waktu komputasi yang signifikan pada tingkat kesamaan yang lebih rendah menunjukkan peningkatan efisiensi dalam deteksi plagiarisme. Hal ini menunjukkan bahwa sistem yang digunakan dalam penelitian ini lebih efisien dalam mengidentifikasi plagiarisme pada dokumen yang tingkat kemiripannya rendah, meskipun deteksi plagiarisme yang teridentifikasi juga lebih sedikit.

3.3. *Hasil Pengujian*

Berikut tabel hasil pengujian black box hasil implementasi algoritma *Levenshtein distance:*

Tabel 6. Hasil Pengujian

No	Skenario Pengujian	Kasus Uji	Hasil yang Diharapkan	Hasil Pengujian
1.	Masukkan PDF yang akan terdeteksi plagiarisme	Masukkan File PDF	Jika di bawah ukuran yang ditentukan maka aplikasi dapat menginput PDF dengan lancar	Sesuai
2.	Periksa plagiarisme	Panggilan cURL, deteksi plagiarisme, perhitungan persentase	Aplikasi dapat memanggil cURL yang terdapat pada API yang telah dibuat, mampu mendeteksi plagiarisme dengan metode yang telah ditentukan dan memberikan catatan akhir persentase plagiarisme terbesar	Sesuai

Tabel 7. Perbandingan Algoritma Kesamaan Teks

Algoritma	Ketepatan	Keuntungan	Kekurangan
Jarak Levenshtein	40-70%	Sederhana, cocok untuk modifikasi kecil	Tidak mempertimbangkan konteks
Kesamaan Kosinus	70-80%	Cocok untuk teks panjang dan kumpulan kata-kata	Tidak memahami sinonim atau konteks
Kesamaan Jaccard	60-75%	Mudah diterapkan, cocok untuk teks pendek	Tidak memahami sinonim atau konteks
Smith-Waterman	80-90%	Baik untuk plagiarisme parsial dalam substring	Lambat untuk teks yang panjang
Jarak Penggerak Kata	85-95%	Memahami sinonim dan makna teks	Membutuhkan model Word2Vec
Kesamaan berbasis BERT	90-98%	Akurat untuk kesamaan semantik yang kompleks	Membutuhkan komputasi yang tinggi

4. SIMPULAN

Pada penelitian ini telah dikembangkan aplikasi pendeteksi plagiarisme yang mampu menganalisis tingkat kemiripan dokumen PDF dengan efisien dan akurat. Aplikasi ini memungkinkan pengguna untuk mengunggah dokumen, memilih metode ekstraksi teks seperti PyPDF2, pdfplumber, atau PyMuPDF, dan memulai proses deteksi dengan algoritma Levenshtein. Hasil analisisnya meliputi persentase plagiarisme, akurasi deteksi, dan waktu pemrosesan, memberikan informasi yang relevan bagi mahasiswa, peneliti, dan akademisi profesional untuk menjaga orisinalitas karyanya. Pengujian menunjukkan aplikasi ini efektif mendeteksi plagiarisme, meski hasilnya bergantung pada metode ekstraksi dan kualitas teks. Untuk pengembangan lebih lanjut, disarankan untuk meningkatkan algoritma pendeteksian dengan model pembelajaran mesin, meningkatkan antarmuka pengguna agar lebih intuitif dan menarik, dan memperluas database referensi untuk mencakup lebih banyak sumber akademis. Selain itu, integrasi dengan sistem pembelajaran, pengujian yang lebih mendalam, penyediaan dokumentasi, dan pembaruan rutin diharapkan dapat meningkatkan fungsionalitas dan relevansi aplikasi. Dengan langkah-langkah tersebut, aplikasi diharapkan mampu memberikan kontribusi yang lebih signifikan dalam menjaga integritas akademik dan memenuhi kebutuhan pengguna yang semakin kompleks.

DAFTAR PUSTAKA

[1] Diktiristek, "Statistik Pendidikan Tinggi 2021 (Final)," Direktorat Jenderal Pendidikan Tinggi, Riset, dan Teknologi, 2021. .

[2] K. Johnson, M. Lee, and T. Robinson, "Comprehensive text *Preprocessing* techniques for efficient natural language processing," *Adv. Comput. Sci. Text Anal.*, 2023 [3] X. Yao, M. H. Yap, and Y. Zhang, "An Empirical Study to Evaluate Structural Similarity for Source Code Translation," 2019, doi: 10.1109/TIMES-iCON47539.2019.9024512.

[4] V. Z. Riadi, "Vector Space Model dan Clustering untuk Deteksi Kesamaan Dokumen," *Skripsi*, Universitas Komputer Indonesia, 2019

[5] J. Nangi, I. B. G. P. Asmara, M. I. Sarita, L. M. G. Jaya, H. T. Mokui, dan L. Tajidun, "Perbandingan Algoritma Winnowing dan Algoritma Rabin-Karp

pada Aplikasi Pendeteksi Kesamaan Dokumen Skripsi," *Jurnal Sistem Informasi Bisnis*, vol. 14, no. 2, hlm. 131–142, 2024.

[6] S. Kurniati, R. Yulianto, and T. Harlina, "Implementasi Metode Perceptual Hash Untuk Deteksi Plagiarisme Tugas Mata Kuliah Software Modeling," Jikom J. Inform. dan Komput., 2023, doi: 10.55794/jikom.v12i2.84.

[7] M. H. Febiawan, A. Setiawan, and A. Primadewi, "Sistem Pendeteksi Dini Plagiarisme Menggunakan Algoritma Levenshtein Distance," J. Komtika (Komputasi dan Inform., 2020, doi: 10.31603/komtika.v3i1.3464.

[8] A. Prasetyo, B. Santoso, dan D. W. Susilo, "Implementasi Algoritma Levenshtein Distance untuk Deteksi Plagiarisme pada Dokumen Teks Bahasa Indonesia," *Jurnal Informatika*, vol. 14, no. 2, hlm. 45–52, 2020.

[9] P. Sunilkumar and A. P. Shaji, "A Survey on Semantic Similarity," 2019, doi: 10.1109/ICAC347590.2019.9036843.

[10] G. Alfian, M. Syafrudin, M. F. Ijaz, M. A. Syaekhoni, N. L. Fitriyani, dan J. Rhee, "Improving efficiency of RFID-based traceability system for perishable food by utilizing IoT sensors and machine learning model," *Food Control*, vol. 110, hlm. 107016, 2020.

[11] Muhammad Romzi and B. Kurniawan, "JTIM : Jurnal Teknik Informatika Mahakarya," JTIM J. Tek. Inform. Mahakarya, 2020.

[12] A. Widodo, "Analyzing consumer journey using the operation edit distance approach," University of Indonesia Thesis, 2023.

[13] K. Johnson, M. Lee, and T. Robinson, "Comprehensive text *Preprocessing* techniques for efficient natural language processing," Adv. Comput. Sci. Text Anal., 2023.

[14] A. Patel, S. Gupta, and L. Roy, "The impact of text *Preprocessing* on machine learning models in NLP," Int. J. Data Sci. Appl., 2023

[15] Tahaei, M., Li, T., & Vaniea, K. (2022). Understanding Privacy-Related Advice on Stack Overflow. Proceedings on Privacy Enhancing Technologies. https://doi.org/10.2478/popets-2022-0038

[16] M. Davis, L. Brown, and R. Wilson, "The significance of *Case Folding* in text *Preprocessing* for natural language processing," J. Text Anal. NLP, 2023

[17] A. Pratama, B. Santoso, & D. W. Susilo. (2023). Implementasi Sistem Deteksi Plagiarisme Dokumen Bahasa Indonesia Menggunakan Metode Vector Space Model. Jurnal Teknologi Informasi, 2023.

[18] H. Taylor, C. Martinez, and P. Nguyen, "Tokenizing techniques and their applications in natural language processing," Int. J. Comput. Linguist. NLP, 2023

[19] Yao, X., Yap, M. H., & Zhang, Y. (2019). An Empirical Study to Evaluate Structural Similarity for Source Code Translation. TIMES-ICON 2019 - 2019 4th Technology Innovation Management and Engineering Science International Conference. https://doi.org/10.1109/TIMES-iCON47539.2019.9024512

[20] A. Smith, J. Doe, and K. Johnson, "The impact of stopword removal on text analysis and classification," Int. J. Nat. Lang. Process., 2023.

[21] Sunilkumar, P., & Shaji, A. P. (2019). A Survey on Semantic Similarity. 2019 6th IEEE International Conference on. https://doi.org/10.1109/ICAC347590.2019.9036843

[22] J. Brown, A. Green, and M. White, "The role of stemming in natural language processing tasks," Int. J. Data Sci. NLP, 2023.

[23] J. Brown, A. Smith, and P. Taylor, "Advancements in Levenshtein Distance applications for text similarity and natural language processing," Int. J. Comput. Linguist., 2023.

[24] A. Patel, S. Gupta, and R. Sharma, "The role of advanced *Preprocessing* in enhancing NLP tasks," J. Data Sci. NLP, 2023.

[25] Jurafsky, D., & Martin, J. H. (2021 *"Speech and Language Processing (3rd Edition)"*A comprehensive resource on natural language processing, covering tokenization, stemming, and distance metrics like Levenshtein.

[26] A. Kurniawan, R. Santoso, and B. Putra, "The impact of *Preprocessing* techniques on text mining performance," Int. J. Comput. Sci. Inf. Technol., 2023.

[27] I. Setiawan, T. Widodo, and L. Saputra, "The influence of text *Preprocessing* techniques on classification accuracy: A comparative analysis," J. Appl. Math. Comput. Sci., 2023.

[28] R. Andika, Y. Pratama, and K. Susilo, "Advancements in word embedding techniques for document similarity measures," Int. Conf. Mach. Learn. Appl., 2023.

[29] Rinaldi Munir Plagiarism Detection Using Levenshtein Distance With Dynamic Programming https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/Makalah/Makalah-Stima-2019-071.pdf?utm.

[30] N. Munir, R. Kusnadi, and S. Halim, "Development of a plagiarism detection system using Levenshtein Distance and *Preprocessing* techniques," Int. J. Comput. Sci. Res., 2023.