Analisis Penerapan Algoritma *Damerau Levenshtein Distance* dengan Ekspansi Sinonim pada Platform PUSDATA

Analysis of the Application of the Damerau-Levenshtein Distance Algorithm with Synonym Expansion on the PUSDATA Platform

Taslim[1], Nuralamsah Zulkarnaim*[2], Wawan Firgiawan[3]
[1,2,3]Teknik Informatika, Universitas Sulawesi Barat
E-mail: [1]taslimharuna21@gmail.com, [2]nuralamsah@unsulbar.ac.id, [3]wawanfirgiawan@unsulbar.ac.id*
**Corresponding author*
Received 25 March 2025; Revised 28 April 2025; Accepted 29 April 2025

Abstrak - Pencarian data pada platform PUSDATA masih terkendala kesalahan pengetikan (*typo*) dan perbedaan istilah yang digunakan pengguna. Hal ini menyebabkan pencarian gagal menemukan data yang relevan jika terjadi kesalahan kecil dalam penulisan kata kunci. Penelitian ini bertujuan menganalisis penerapan algoritma *Damerau Levenshtein Distance* dengan ekspansi sinonim untuk meningkatkan akurasi pencarian data. Metode yang digunakan meliputi *tokenisasi*, perhitungan jarak string untuk memperbaiki kesalahan ketik, dan penambahan sinonim guna memperluas pencarian. *Dataset* diperoleh dari *scraping* 500 judul *dataset* UCL *Repository* dan memperkaya makna kata dengan sinonim dari artikata.com. Pengujian dilakukan menggunakan kata kunci dengan variasi kesalahan penulisan. Hasil pengujian menunjukkan algoritma mampu memberikan rekomendasi kata yang relevan dengan akurasi 90% dari 40 data uji, di mana 36 data berhasil dikenali dengan benar dan 4 data tidak sesuai karena kemiripan karakter rendah. Penerapan algoritma ini efektif dalam meningkatkan kualitas pencarian data dengan mengoreksi berbagai kesalahan pengetikan dan menangani variasi sinonim kata kunci. Hasil ini menunjukkan sistem pencarian menjadi lebih fleksibel dan akurat dalam menemukan data meskipun terdapat kesalahan penulisan.

Kata kunci: *Damerau Levenshtein Distance*, Ekspansi Sinonim, Pencarian Data, PUSDATA,

Abstract - *The data search on the PUSDATA platform faces challenges caused by typographical errors (typos) and differences in terminology used by users. These issues result in the failure to find relevant data when minor errors occur in keyword input. This study aims to analyze the application of the Damerau-Levenshtein Distance algorithm combined with synonym expansion to improve search accuracy. The method includes tokenization, string distance calculation to correct typos, and synonym expansion to broaden search coverage. The dataset was obtained by scraping 500 dataset titles from the UCL Repository and expanding the meaning of words with synonyms from artikata.com. Testing was conducted by inputting keywords with various writing errors. The results showed that the algorithm effectively provided relevant word recommendations with an accuracy of 90% from 40 test data, where 36 were successfully recognized, and 4 failed due to low character similarity. The implementation of this algorithm effectively improves data search quality by correcting typing errors and handling synonym variations. These results indicate that the search system becomes more flexible and accurate in finding data even when writing errors occur.*

Keywords: *Damerau Levenshtein Distance, Synonym Expansion, Data Search, PUSDATA,*

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI :10.62411/tc.v24i2.12557

1. PENDAHULUAN

Dalam era digital saat ini, pengelolaan dan pengumpulan data menjadi aspek yang sangat penting dalam berbagai bidang, mulai dari riset akademis hingga industri teknologi. Platform pengumpulan *dataset* seperti PUSDATA memberikan kemudahan terhadap pengguna dalam mengakses berbagai sumber data yang diperlukan untuk analisis lebih lanjut. Namun, sistem pencarian di PUSDATA saat ini masih memiliki keterbatasan karena bergantung pada *query* manual yang sensitif terhadap kesalahan pengetikan. Jenis kesalahan pengetikan ini mencangkup penambahan huruf, pnenempatan huruf yang salah, penghapusan huruf dan juga modifikasi kata [1]. Kesalahan-kesalahan tersebut mengakibatkan data yang dicari tidak ditemukan karena tidak ada kecocokan anatara data yang ada dengan kata kunci pengguna [2]. Hal ini dapat membuat pengguna merasa tidak nyaman karena kesulitan menemukan data yang mereka cari, sehingga akhirnya mereka meninggalkan situs website [3]. Algoritma yang efektif dalam mendeteksi kesalahan pengetikan berdasarkan kemiripan penulisan antara teks yang dimasukkan dengan yang tersimpan dalam program akan sangat bermanfaat untuk mengatasi permasalahan ini [4].

Namun, pencarian berbasis teks secara literal juga memiliki keterbatasan lain, yaitu tidak mempertimbangkan sinonim atau variasi istilah dalam konteks semantik. Perbedaan istilah yang digunakan oleh pengguna memerlukan perhatian khusus agar hasil pencarian sesuai dengan tujuan yang diinginkan. Dengan menerapkan metode yang memperluas pencarian untuk mencakup sinonim dari istilah, pencarian tidak hanya akan terbatas pada istilah yang dimasukkan tetapi juga mencakup sinonimnya [5].

Penelitian ini tentunya juga tidak lepas dari penelitian terkait sebelumnya untuk memperkuat algoritma yang akan diterapkan dalam sistem pencarian. Penelitian yang dilakukan oleh Arsyta 2023 menghasilkan pengujian akurasi yang menunjukkan tingkat akurasi sebesar 100% yang menandakan bahwa algoritma tersebut dalam mengenali dan mengahasilkan saran pencarian yang relevan [6]. Penelitian serupa juga dilakukan oleh Christanti 2020 yang memperoleh hasil penerapan *Damerau Levenshtein Distance* dengan akurasi kata sebesar 88% [7]. Penelitian lain yang dilakukan oleh Soundex 2019 yang melakukan perbandingan algoritma *Damerau Levenshtein Distance* dan *Soundex Similarity* yang menunjukkan *Damerau Levenshtein Distance* berhasil mendapatkan akurasi sebesar 72% sedangkan *Soundex Similarity* memperoleh akurasi sebesar 68% [8]. Dengan *Damerau Levenshtein Distance* unggul dalam hal ketepatan hasil pencarian. Algoritma *Damerau Levenshtein Distance* berhasil diterapkan sehingga bisa memberi saran kata untuk ejaan yang salah [9]. Sebagaimana penelitian yang dilakukan oleh Indriyono 2020 bahwa Algoritma *Damerau Levenshtein Distance* memiliki kemampuan untuk mengoreksi kesalahan penulisan kata dan memberikan saran kata yang paling sesuai, dengan cara menghitung jarak paling dekat antara ejaan dalam dokumen dan ejaan dalam kamus bahasa Inggris yang memiliki tingkat kemiripan tinggi [10]. Penelitian sebelumnya juga menyebutkan bahwa Algoritma *Damerau-Levenshtein* memiliki sensitivitas sebesar 85% meskipun cakupannya hanya 74% untuk percobaan ini, dan algoritma ini juga bekerja dengan baik untuk karakter transposisi [11]. Selain itu, berdasarkan hasil penelitian yang telah dilakukan oleh Marcel 2020, dapat disimpulkan bahwa algoritma *Levenshtein Distance* berhasil diterapkan untuk mendeteksi kesalahan ejaan (*misspelled word*) dalam sistem pencarian lagu melayu karya Arif Putra. Dari hasil pengujian perangkat lunak yang dilakukan melalui kuesioner kepada 85 responden, diketahui bahwa 23,53% responden sangat setuju dan 60% responden setuju bahwa algoritma *Levenshtein Distance* yang digunakan mampu mengurangi kesalahan ejaan pada website pencarian lagu melayu tersebut [12].

Berdasarkan urain permasalahan diatas dan didukung oleh hasil beberapa penelitian sebelumnya, maka metode *Damerau Levenshtein Distance* dengan Ekspansi Sinonim dipilih kerangka kerja yang akan diterapkan pada sistem pencarian platform PUSDATA. Penelitian ini bertujuan untuk mengetahui kinerja pencarian data menggunakan kombinasi metode *Damerau Levenshtein Distance* dengan Ekspansi Sinonim yang dapat membantu meningkatkan hasil

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI :10.62411/tc.v24i2.12557

pencarian dengan memperbaiki kesalahan pengetikan serta menangani variasi istilah, sehingga mengahsilkan pencarian yang lebih relevan dan akurat.

2. METODE PENELITIAN

2.1 *Tokenisasi*

Tokenisasi merupakan proses memecah teks menjadi bagian-bagian kecil baik itu berupa kata, frasa, atau karakter tergantung pada level yang diinginkan [13]. Dalam penelitian ini, akan diterapkan tokenisasi pada tingkat kata untuk memproses teks, sehingga mempermudah perbandingan per kata serta mengurangi kompleksitas perhitungan. Berikut adalah contoh hasil tokenisasi pada tabel 1:

Tabel 1. Contoh *Tokenisasi* tingkat kata

Kalimat Asli	Hasil Tokenisasi
"Saya sedang belajar pemrograman."	Saya, sedang, belajar, pemrograman, .
"Tokenisasi mempermudah analisis teks."	Tokenisasi, mempermudah, analisis, teks, .
"Pemrosesan bahasa alami sangat menarik!"	Pemrosesan, bahasa, alami, sangat, menarik, !

2.2 Damerau Levenshtein Distance

Damerau-Levenstheins Distance, dalam ilmu komputer, adalah sebuah algoritma yang dikembangkan oleh Frederick J. Damerau dan Vladimir Levenshtein sebagai sebuah bentuk improvisasi dari algoritma *Levenshtein distance*. *Damerau Levenshtein Distance* menghitung nilai minimum yang diperlukan untuk mengubah satu string menjadi string lainnya [14]. Dalam algoritma awalnya (*Levenshtein*), terdapat 3 operasi yaitu *insertions* (insersi), *deletions* (penghapusan), dan *subtitutions* (subtitusi). Sedangkan dalam versi penyempurnaan yang diusulkan oleh Damerau, beliau menambahkan operasi *transposisi* dari dua karakter yang bersebelahan [15]. Dari hal tersebut, Algoritma *Damerau-Levenshtein Distance* dapat dirumuskan pada persamaan (1) berikut:

$$DL(a,b)(i,j) = min \begin{cases} 0 & jika\ i = j = 0 \\ DL(a,b)(i-1,j)+1 & jika\ i > 0 \\ DL(a,b)(i,j-1)+1 & jika\ j > 0 \\ DL(a,b)(i-1,j-1)+1\ (ai \neq bj) & jika\ i,j > 0 \\ DL(a,b)(i-2,j-2)+1 & jika\ i,j > 1\ dan\ a[i] = \text{b[j-1] dan a[i-1] = b[j]} \end{cases} \quad (1)$$

dengan a dan b merupakan 2 buah string yang akan dibandingkan. Masing masing kasus diatas berkorespondensi sebagai berikut:

$DL(a,b)\ (i-1,j) + 1$, *deletions*
$DL(a,b)\ (i,j-1) + 1$, *insertions* I
$DL(a,b)\ (i-1,j-1) + 1(ai \neq bj)$, *subtitutions*
$DL(a,b)\ (i-2,j-2) + 1$, *tranpositions*

Tabel 2, [16] merepresentasikan matriks *Damerau Levenshtein distance* m x n. Pada kasus ini, m dan n = 4. Kata kunci dan kata yang ditarget memiliki panjang karakter yang sama. Kesalahan ejaan hanya terdapat pada dua huruf yang posisinya terbalik. Pemeriksaan karakter dilakukan mulai s terhadap t. Isikan nilai dari setiap sel d[i, j] baris perbaris. Langkah ini akan terus berulang sampai semua matriks terisi (Tabel 2). Dari contoh kata "KACO" dan "KAOC" hanya memiliki satu jarak perbedaan nilai jarak. Pengoreksian untuk contoh ini hanya memperlukan satu operasi yaitu penukaran dua huruf yang berdekatan unuk merubah KAOC menjadi KACO.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI :10.62411/tc.v24i2.12557

Tabel 2. Contoh matriks *Damerau Levenshtein Distance* 4 x 4

S/T		**K**	**A**	**C**	**O**
	0	1	2	3	4
K	1	0	1	2	3
A	2	1	0	1	2
O	3	2	1	1	1
C	4	3	2	1	1

2.3 Kueri Ekspansi Sinonim

Kueri ekspansi artinya memperluas kueri dengan cara menambahkan kata-kata yang paling berkaitan dengan kata kunci kueri [17]. Salah satu metode dalam kueri ekspansi adalah ekspansi dengan sinonim, yaitu menambahkan kata-kata yang berbeda tetapi memiliki arti yang sama atau mirip dengan kata kunci yang dimasukkan oleh pengguna. Metode ini didasarkan pada hubungan semantik antar kata, sehingga memungkinkan sistem pencarian untuk menemukan informasi yang relevan meskipun tidak menggunakan kata kunci yang persis sama.

Salah satu metode dalam kueri ekspansi adalah ekspansi dengan sinonim yang memiliki hubungan antara istilah-istilah yang berbeda dari kata inputan tetapi memiliki makna yang sama. Kueri ekspansi ini dapat digunakan untuk meningkatkan efektivitas pencarian dalam mesin pencari [18].

Contoh penerapan kueri ekspansi sinonim dalam sistem adalah sebagai berikut: Misalnya, pengguna memasukkan kata kunci "dokter", namun dalam database yang tersedia, istilah yang digunakan adalah "medis". Dalam kondisi normal, pencarian kata "dokter" tidak akan menemukan data tersebut. Namun dengan kueri ekspansi, sistem akan menampilkan kata "medis", atau "kedokteran" ke dalam daftar kata yang dicocokkan. Setelah kata-kata sinonim ini ditambahkan, algoritma Damerau-Levenshtein Distance akan menghitung jarak atau perbedaan karakter antar kata dari kueri.

Dalam sistem pencarian, tidak semua kondisi membutuhkan algoritma atau kueri ekspansi. Jika pengguna mengetik kata kunci dengan benar dan lengkap, serta kata tersebut memang ada dalam dokumen dengan ejaan yang sama, maka sistem cukup menggunakan pencocokan langsung. Artinya, sistem bisa langsung menemukan hasil yang sesuai tanpa perlu melakukan proses tambahan seperti pengecekan kesalahan atau perluasan kata

Namun, jika pengguna salah mengetik kata, misalnya huruf tertukar atau ada huruf yang kurang, maka sistem perlu menggunakan algoritma seperti Damerau-Levenshtein Distance. Algoritma ini berguna untuk mengenali kata yang mirip dengan apa yang dimaksud pengguna meskipun terjadi kesalahan penulisan. Contohnya, saat pengguna mengetik "doter" padahal maksudnya "dokter", sistem bisa memahami maksud sebenarnya dan tetap menampilkan hasil yang relevan.

Selain itu, sistem juga bisa menambahkan sinonim atau kata-kata yang mempunyai arti mirip ke dalam pencarian. Ini disebut kueri ekspansi dengan sinonim. Tujuannya adalah agar pencarian lebih luas dan tidak terbatas pada satu kata saja. Misalnya, jika pengguna mengetik "dokter", sistem juga bisa mencari kata seperti "medis" atau "kedokteran" agar hasilnya lebih lengkap. Sistem bisa menggunakan ketiga cara ini secara bersamaan, mencocokkan kata secara langsung, memperbaiki kesalahan ketik dengan algoritma, dan menambahkan sinonim untuk memperluas pencarian. Dengan begitu, hasil pencarian jadi lebih akurat dan sesuai dengan apa yang dicari pengguna.

2.4 Perhitungan Nilai Evaluasi

Pada evaluasi ini akan menguji kinerja algoritma dalam memberikan rekomendasi kata yang sesuai dengan keinginan pengguna [19]. Evaluasi dilakukan dengan cara menghitung seberapa banyak hasil rekomendasi algoritma yang benar dibandingkan dengan jumlah total pengujian yang dilakukan.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI :10.62411/tc.v24i2.12557

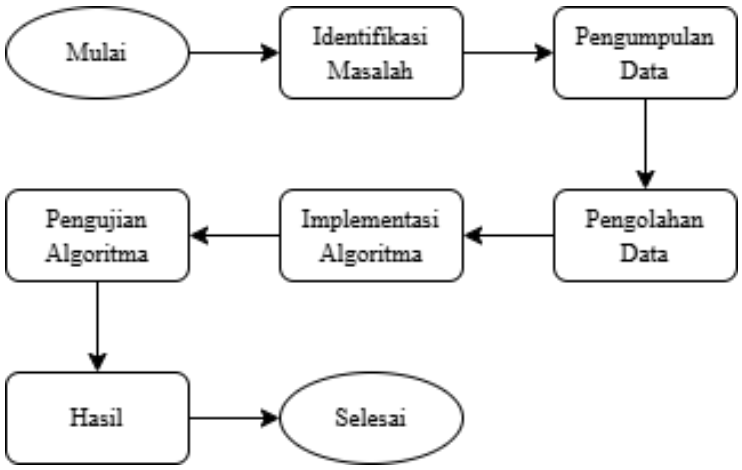
Untuk mengukur tingkat keberhasilan algoritma, digunakan rumus akurasi sebagai berikut:

$$Akurasi = \left(\frac{Data\ yang\ benar}{Total\ Data\ Uji}\right) \times 100\%$$

Jumlah Data yang Benar merupakan total kata yang berhasil direkomendasikan oleh algoritma dan sesuai dengan kata yang dimaksud oleh pengguna. Artinya, kata yang dihasilkan oleh sistem sudah tepat dan relevan dengan target kata yang diinginkan. Sedangkan Total Data Uji adalah keseluruhan data yang digunakan dalam proses pengujian, yaitu jumlah semua kata uji yang diinput untuk mengukur seberapa baik algoritma bekerja. Dengan membandingkan jumlah data yang benar dan total data uji, kita bisa mengetahui seberapa besar tingkat keberhasilan atau akurasi dari algoritma yang diterapkan.

2.5 Metode Penelitian

Metode penelitian mencakup tahapan dalam pelaksanaan penelitian [20]. Pada gambar 2 merupakan tahap penelitian yang akan dilakukan.



Gambar 1. Digram Metode Penelitian

1. Identifikasi masalah

   Pada tahap ini, permasalahan utama dalam penelitian diidentifikasi secara jelas. Tantangan yang dihadapi adalah bagaimana memastikan pencarian data tetap memberikan hasil yang relevan meskipun terdapat kesalahan ketik (*typo*) dan variasi sinonim dalam teks pencarian. Identifikasi dilakukan dengan menganalisis sistem pencarian PUSDATA secara mendalam untuk menemukan akar masalah, mulai dari sensitivitas terhadap kesalahan penulisan hingga kurangnya pemahaman sistem terhadap makna kata yang berbeda namun memiliki arti sama. Dampak dari masalah ini menyebabkan pengguna kesulitan mendapatkan data yang dicari dan berpotensi menurunkan tingkat kepercayaan pengguna terhadap platform.
2. Pengumpulan data

   Tahap ini bertujuan untuk mengumpulkan informasi dan data yang relevan untuk mendukung penelitian. Data yang dikumpulkan meliputi korpus teks yang akan digunakan sebagai basis pencarian, daftar sinonim untuk ekspansi kata, serta *dataset* uji untuk

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI :10.62411/tc.v24i2.12557

mengevaluasi kinerja algoritma. Proses pengumpulan data dilakukan melalui *scraping* 500 judul dataset dari UCL Repository, dipilih karena keragamannya yang mewakili berbagai konteks kata kunci. Selain itu, daftar sinonim dari artikata.com dikumpulkan untuk mendukung proses ekspansi kata kunci, memastikan sistem mampu mengenali berbagai variasi kata dengan makna yang sama dalam pencarian.

3. Pengolahan data

   Setelah data terkumpul, tahap ini melibatkan pembersihan, analisis, dan penyusunan ulang data agar dapat digunakan dalam implementasi algoritma. Pembersihan data mencakup penghapusan karakter khusus, normalisasi teks, serta penghapusan stopword yang tidak relevan. Pengolahan data menjadi tahap krusial agar data terstruktur dan siap diolah algoritma. Normalisasi dilakukan untuk menyeragamkan data, menghindari perbedaan akibat huruf kapital, simbol, atau spasi berlebih. *Tokenisasi* memecah kalimat menjadi per kata untuk memudahkan perhitungan jarak string dan memfasilitasi pencocokan data dengan sinonim. Penghapusan stopword bertujuan menghilangkan kata-kata yang tidak berdampak pada hasil pencarian seperti 'yang', 'dan', 'atau'.

4. Implementasi algoritma

   Tahapan selanjutnya adalah implementasi algoritma yang telah dipilih dalam penelitian ini yaitu algoritma *Damerau Levenshtein Distance* dengan Ekspansi Sinonim. Implementasi dimulai dengan menghitung jarak antara input pengguna dan dataset menggunakan *Damerau Levenshtein Distance* untuk mendeteksi dan memperbaiki kesalahan ketik. Sistem kemudian memperluas pencarian dengan menambahkan daftar sinonim dari input pengguna, sehingga sistem dapat menghubungkan kata yang berbeda namun memiliki makna serupa. Proses ini meningkatkan fleksibilitas sistem dalam memahami variasi kata dan *typo* secara bersamaan.

5. Pengujian algoritma

   Selanjutnya, tahap ini akan dilakukan pengujian algoritma yang telah diterapkan sebelumnya, pengujian dengan menguji hasil algoritma *Damerau Levenshtein Distance* dengan ekspansi sinonim dalam memberikan rekomendasi kata yang diinginkan pengguna. Pengujian dilakukan dengan skenario pengujian terkontrol menggunakan kata kunci yang sengaja dibuat salah penulisannya, seperti penambahan huruf, pengurangan huruf, atau pertukaran posisi huruf. Evaluasi setiap hasil output dilakukan secara manual untuk memastikan apakah rekomendasi sistem benar-benar sesuai dengan target kata kunci. Pengujian ini sekaligus menguji seberapa efektif kombinasi metode dalam menangani *typo* dan variasi kata.

6. Hasil

   Tahap terakhir adalah penyajian hasil dari pengujian dan analisis data yang telah dilakukan. Hasil ini mencakup evaluasi efektivitas algoritma dalam meningkatkan kualitas pencarian, termasuk seberapa baik sistem dapat mengenali kesalahan ketik dan memperluas pencarian menggunakan sinonim. Pada tahap ini, selain menghitung akurasi, dilakukan juga analisis terhadap kekuatan dan kelemahan dari algoritma yang digunakan. Evaluasi bertujuan untuk mengukur sejauh mana algoritma dapat memperbaiki kesalahan ketik dan memperkaya hasil pencarian dengan memanfaatkan sinonim yang relevan. Hasil analisis ini akan menjadi dasar rekomendasi pengembangan sistem di masa depan.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI :10.62411/tc.v24i2.12557

3. HASIL DAN PEMBAHASAN

3.1 Sumber Data

Sumber data dalam penelitian ini dilakukan melalui proses *scraping* atau pengambilan data secara otomatis dari sebuah website, yaitu UCL Repository. UCL Repository dipilih karena memiliki banyak koleksi *dataset* dari berbagai bidang ilmu, sehingga cocok digunakan sebagai sumber data penelitian. Proses *scraping* dilakukan dengan bantuan *tools* yang dapat menarik data dari website secara cepat dan terstruktur. Dari proses ini, berhasil dikumpulkan 500 judul *dataset* yang masing-masing mewakili berbagai macam topik dan kata kunci yang bervariasi.

Judul-judul *dataset* yang berhasil dikumpulkan ini kemudian dimasukkan ke dalam platform PUSDATA untuk digunakan sebagai data utama dalam penelitian. Data ini akan menjadi bahan uji coba untuk melihat bagaimana sistem dapat memperbaiki kesalahan ketik dan mengenali sinonim kata saat melakukan pencarian. Selain itu, data ini juga akan diproses lebih lanjut melalui beberapa tahapan seperti *tokenisasi* dan perhitungan jarak string menggunakan algoritma *Damerau Levenshtein Distance.*

Agar sistem pencarian tidak hanya terpaku pada kata kunci yang sama persis, maka dilakukan juga proses ekspansi sinonim. Proses ini bertujuan untuk memperluas arti atau makna dari setiap kata dalam dataset. Dengan begitu, sistem pencarian akan tetap dapat menemukan data yang relevan meskipun pengguna mengetikkan kata yang berbeda tapi memiliki arti yang sama. Misalnya, jika di dalam dataset ada kata "mobil", maka dengan adanya sinonim, sistem juga bisa mengenali kata seperti "kendaraan" atau "otomobil".

Untuk mendapatkan daftar sinonim, penelitian ini menggunakan data dari situs artikata.com, yaitu sebuah website yang menyediakan kumpulan sinonim dalam bahasa Indonesia. Setiap kata penting dalam dataset dicocokkan dengan daftar sinonim dari situs tersebut, lalu ditambahkan ke dalam data. Proses ini dilakukan secara bertahap hingga semua kata kunci penting memiliki pasangan sinonimnya masing-masing.

Dengan adanya proses penambahan sinonim ini, sistem pencarian akan menjadi lebih pintar dan fleksibel. Artinya, sistem tidak hanya cocok ketika pengguna mengetik kata yang sama persis, tetapi juga tetap bisa menemukan data yang relevan meskipun terjadi perbedaan dalam penulisan atau penggunaan kata. Selain itu, proses ini juga membantu sistem dalam mengatasi kesalahan ketik (*typo*) yang sering terjadi ketika pengguna memasukkan kata kunci pencarian.

Secara keseluruhan, proses pengumpulan data ini menjadi bagian penting dalam penelitian karena memastikan bahwa data yang digunakan cukup beragam dan kaya akan variasi kata. Dengan begitu, sistem yang dibangun benar-benar diuji dalam kondisi yang mirip dengan situasi nyata, di mana pengguna sering kali salah mengetik kata atau menggunakan istilah yang berbeda dalam melakukan pencarian data.

3.2 Proses Pengolahan Algoritma

Setelah data berhasil dikumpulkan dan dimasukkan ke dalam platform PUSDATA, langkah selanjutnya adalah proses pengolahan data menggunakan algoritma yang telah ditentukan. Proses ini bertujuan untuk menguji kemampuan sistem dalam mengenali kesalahan penulisan kata dan menyarankan hasil pencarian yang relevan, baik berdasarkan kemiripan ejaan maupun makna kata melalui sinonim. Untuk mencapai tujuan tersebut, sistem melakukan serangkaian tahapan mulai dari menerima input pengguna hingga menghasilkan output dalam bentuk saran atau hasil pencarian yang relevan. Tahapan-tahapan ini dijelaskan sebagai berikut:

1. Input kata kunci oleh pengguna

Pengguna memasukkan kata kunci ke dalam sistem pencarian pada platform PUSDATA. Kata kunci ini bisa saja mengandung kesalahan pengetikan (*typo*) atau

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI :10.62411/tc.v24i2.12557

menggunakan istilah berbeda dari yang ada di dalam dataset, sehingga sistem perlu memprosesnya secara cermat.

Contoh : Pengguna ingin mencari informasi tentang kecerdasan buatan, tetapi mengetik "kecardasan buatan", terjadi kesalahan ketik pada kata "kecerdasan".

2. *Tokenisasi* kata kunci

Setelah pengguna memasukkan kata kunci, sistem akan memecah kata tersebut melalui proses *tokenisasi*. *Tokenisasi* dilakukan untuk memudahkan sistem membandingkan setiap kata dengan data dalam dataset. Pemecahan ini juga membantu dalam perhitungan jarak antar *string* secara lebih efisien.

Contoh : Kata kunci "kecardasan buatan" dipecah menjadi dua token "kecardasan", "buatan" Setiap token akan diproses secara terpisah.

3. Pencocokan kata dengan *Dataset*

Setiap token akan dibandingkan dengan seluruh kata dalam dataset menggunakan algoritma *Damerau Levenshtein Distance*. Algoritma ini menghitung jarak edit antara dua kata. Semakin kecil nilai skor yang diperoleh, maka semakin tinggi tingkat kemiripan antara kata pengguna dengan kata yang ada di dataset. Jika jarak berada di bawah ambang batas tertentu, maka sistem akan menganggap kata tersebut relevan.

Contoh : "kecardasan" dibandingkan dengan "kecerdasan" jarak edit = 1 dan "buatan" cocok langsung dengan data "buatan" jarak edit = 0 keduanya dianggap relevan dan akan diproses lebih lanjut.

4. Ekspansi sinonim

Jika kata kunci yang diberikan tidak ditemukan secara langsung, sistem akan memperluas pencarian den gan menambahkan sinonim dari kata tersebut. Daftar sinonim diambil dari situs artikata.com. Sinonim yang diperoleh akan diproses kembali seperti kata asli dan dibandingkan dengan data yang ada, sehingga peluang pencocokan menjadi lebih luas.

Contoh : Jika pengguna mengetik "cerdas" dan tidak ditemukan di dataset, maka sistem mengambil sinonim seperti "kecerdasan buatan" . Kemudian sinonim ini diproses kembali untuk pencocokan.

5. Pemilihan hasil dan output

Setelah seluruh proses pen cocokan selesai, baik dari kata asli maupun sinonim, sistem akan memilih hasil dengan jarak terkecil untuk ditampilkan kepada pengguna. Hasil pencarian yang muncul merupakan kata atau frasa yang paling mendekati maksud dari input pengguna, baik secara ejaan maupun makna. Dengan tahapan ini, sistem dapat mengatasi *typo* dan variasi kata, serta memberikan hasil pencarian yang lebih akurat dan bermanfaat.

Contoh: Sistem menampilkan "kecerdasan buatan" sebagai hasil pencarian dari input "kecardasan buatan" atau "cerdas", karena memiliki kemiripan makna dan ejaan yang tinggi.

3.3 Hasil Pengujian

Setelah tahap implementasi selesai, dilakukan proses pengujian untuk melihat seberapa baik algoritma *Damerau Levenshtein Distance* yang dipadukan dengan ekspansi sinonim dalam memperbaiki kesalahan pengetikan dan menemukan kata yang sesuai. Pengujian dilakukan dengan memasukkan beberapa kata kunci yang sengaja dibuat salah penulisannya (*typo*) dengan berbagai variasi, seperti penambahan huruf, pengurangan huruf, pertukaran posisi huruf, hingga penulisan yang jauh berbeda dari kata aslinya.

Setiap kata kunci yang diinput akan diproses oleh algoritma, lalu dibandingkan apakah hasil rekomendasi kata dari sistem sesuai dengan kata yang dimaksud oleh pengguna. Jika hasilnya sesuai, maka validasi dianggap benar, namun jika tidak sesuai, maka dianggap salah. Pengujian ini bertujuan untuk mengukur kemampuan sistem dalam memberikan rekomendasi kata yang relevan meskipun terjadi kesalahan ketik.

Berikut ini adalah hasil pengujian yang telah dilakukan dan dirangkum dalam tabel berikut:

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI :10.62411/tc.v24i2.12557

Tabel 3. Hasil Pengujian Algoritma *Damerau Levensthein Distance*

Pengujian	**Kata kunci yang diinputkan**	**Kata yang dimaksud**	**Validasi**
1	Iras	Iris	Benar
2	Adulsss	Adult	Benar
3	Gisekke	Gisette	Salah
4	Hpatitis	Hepatitis	Benar
5	Hepatutis	Hepatitis	Benar
6	Heppatulis	Hepatitis	Benar
7	Spaambas	Spambase	Benar
8	Sapambasssee	Spambase	Benar
9	Spammbassuu	Spambase	Benar
10	Anneallingss	Annealing	Benar
11	Kannealling	Annealing	Benar
12	Avian	Avila	Salah
13	Alvian	Avila	Salah
14	Gise	Gisette	Benar
15	isette	Gisette	Benar
16	Farm add	Farm ads	Benar
17	Famm dds	Farm ads	Benar
18	NoisOffice	NoisyOffice	Benar
19	MaedReminder	MaskReminder	Benar
20	widiaAHE	Wiki4HE	Benar

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI :10.62411/tc.v24i2.12557

21	Covetype	Covertype	Benar
22	Cvoertype	Covertype	Benar
23	Covertyype	Covertype	Benar
24	Raisi	Raisin	Benar
25	Raiin	Raisin	Benar
26	Arhythmia	Arrhythmia	Benar
27	Arryhthmia	Arrhythmia	Benar
28	Arrhytyymia	Arrhythmia	Benar
29	Fertillit	Fertility	Benar
30	Fertikal	Fertility	Benar
31	yeasy	Yeast	Benar
32	yeasstt	Yeast	Benar
33	Parkinnsons	Parkinsons	Benar
34	Parkinsonss	Parkinsons	Benar
35	Fllagggssss	Flags	Salah
36	Hepatis	Hepatitis	Benar
37	Abalonee	Abalone	Benar
38	Movei	Movie	Benar
39	Exercise	Physical Therapy Exercises	Benar
40	Ecol	Ecoli	Benar

Tabel 4. Hasil Pengujian Algoritma *Damerau Levensthein Distance*
Dengan beberapa skenario pengujian kesalahan ketik dan sinonim

No	**Jenis kesalahan**	**Kata kunci**	**Kata yang dimaksud**	**Deskripsi**
1	Insertions	Parkinnsons	Parkinsons	Terjadi penambahan huruf '**n**' ganda di

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI :10.62411/tc.v24i2.12557

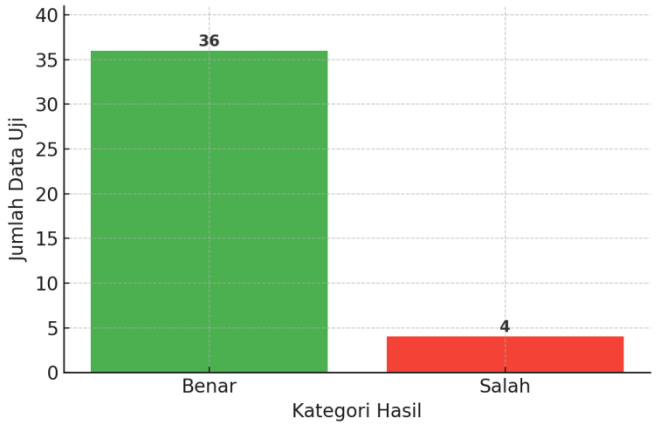
				tengah kata.
2	Deletions	Hepatis	Hepatitis	Huruf **'i'** di bagian tengah kata hilang akibat salah ketik.
3	Substitutions	Abalonee	Abalone	Huruf **'e'** ditambahkan sebagai pengganti di akhir kata.
4	Transposisi	Movei	Movie	Huruf **'e'** dan **'i'** tertukar posisi di akhir kata.
5	Sinonim	Exercise	Physical Therapy Exercises	Kata **'exercise'** adalah istilah umum yang sering digunakan untuk merujuk pada aktivitas seperti **'physical therapy exercises'**.

Berdasarkan hasil pengujian yang telah dilakukan terhadap 40 data uji dengan berbagai variasi kesalahan penulisan kata kunci, diperoleh hasil bahwa algoritma *Damerau Levenshtein Distance* dengan ekspansi sinonim mampu memberikan tingkat akurasi yang sangat baik dalam menemukan kata yang dimaksud oleh pengguna meskipun terdapat kesalahan ketik (*typo*).

Dari total 40 pengujian, sebanyak 36 pengujian berhasil memberikan output yang benar dan sesuai dengan kata yang dimaksud, sementara 4 pengujian menghasilkan output yang tidak sesuai. Hal ini menunjukkan bahwa algoritma memiliki tingkat keberhasilan sebesar 90%, dihitung menggunakan rumus akurasi sebagai berikut:

$$Akurasi = \left(\frac{36}{40}\right) \times 100\% = 90\%$$

Selain ditampilkan dalam bentuk tabel, hasil pengujian ini juga divisualisasikan melalui grafik batang untuk memperjelas perbandingan antara pencarian yang berhasil dan yang tidak berhasil dalam menemukan kata yang dimaksud oleh pengguna. Grafik berikut memberikan gambaran tingkat keberhasilan algoritma dalam melakukan koreksi terhadap input yang salah.



Grafik 1. Hasil Pengujian Algoritma *Damerau Levensthein Distance*

Beberapa kasus seperti "Iras" yang diubah menjadi "Iris" dan "Hpatitis" yang dikenali sebagai "Hepatitis" menunjukkan bahwa algoritma berhasil menangani kesalahan penambahan, pengurangan, serta transposisi huruf dengan baik. Bahkan pada input dengan banyak kesalahan

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI :10.62411/tc.v24i2.12557

seperti "Spaambas", "Sapambasssee", dan "Spammbassuu", sistem tetap mampu merekomendasikan kata "Spambase" secara tepat.

Namun demikian, terdapat beberapa kata yang tidak berhasil dikenali atau dipetakan dengan benar oleh algoritma. Contohnya adalah kata "Gisekke" yang tidak dikenali sebagai "Gisette", serta "Avian" dan "Alvian" yang gagal dipetakan ke "Avila". Selain itu, pada kasus "Fllagggssss", algoritma tidak berhasil mengidentifikasi kata yang dimaksud, yaitu "Flags", karena terlalu banyak karakter tambahan yang menyebabkan jarak edit menjadi terlalu besar. Kasus-kasus ini menunjukkan adanya batasan pada algoritma, terutama ketika tingkat kemiripan karakter antar kata sangat rendah atau ketika jumlah kesalahan dalam satu input terlalu banyak. Dalam situasi seperti ini, kemampuan algoritma untuk menghasilkan hasil yang relevan menjadi terbatas, sehingga diperlukan pendekatan tambahan atau kombinasi dengan metode lain untuk meningkatkan akurasi.

Secara keseluruhan, pengujian ini membuktikan bahwa penerapan algoritma *Damerau Levenshtein Distance* yang dipadukan dengan ekspansi sinonim mampu meningkatkan efektivitas pencarian data dengan menangani berbagai macam variasi kesalahan ketik, sehingga memberikan hasil pencarian yang lebih relevan dan akurat.

4. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian dan pengujian yang telah dilakukan terhadap 40 data uji, penerapan algoritma *Damerau Levenshtein Distance* dengan ekspansi sinonim pada platform PUSDATA terbukti mampu meningkatkan efektivitas pencarian data, khususnya dalam menangani kesalahan ketik (*typo*) dan variasi sinonim pada kata kunci. Dari total pengujian tersebut, algoritma berhasil memberikan hasil yang benar sebanyak 36 kali, dengan tingkat akurasi mencapai 90%, sementara 4 pencarian tidak sesuai dengan kata yang dimaksud.

Kelebihan dari metode ini terletak pada kemampuannya dalam mengoreksi berbagai jenis kesalahan penulisan, termasuk penambahan huruf, pengurangan huruf, transposisi huruf, dan penggantian karakter. Selain itu, fitur ekspansi sinonim terbukti efektif dalam memperluas cakupan pencarian, sehingga pengguna tetap bisa mendapatkan hasil yang relevan meskipun menggunakan istilah berbeda. Hal ini sangat membantu ketika pengguna tidak mengetahui istilah teknis yang tepat atau mengalami kesalahan dalam mengetik kata kunci.

Namun, terdapat pula beberapa kelemahan yang perlu diperhatikan. Algoritma ini masih memiliki keterbatasan dalam mengenali kata yang memiliki tingkat kemiripan karakter sangat rendah atau ketika jumlah kesalahan terlalu banyak. Contohnya terlihat pada kata "Gisekke" yang tidak berhasil dipetakan ke "Gisette", serta "Avian" dan "Alvian" yang gagal dikenali sebagai "Avila". Begitu pula dengan kata "Fllagggssss" yang terlalu banyak mengandung karakter tambahan sehingga tidak dikenali sebagai "Flags". Hal ini menunjukkan bahwa meskipun algoritma cukup andal dalam banyak kasus, tetap ada batasan yang harus diatasi melalui pengembangan lebih lanjut.

Saran untuk penelitian selanjutnya adalah memperkaya database *sinonim* agar cakupan pencarian semakin luas dan akurat. Selain itu, pengujian sebaiknya dilakukan dengan jumlah data yang lebih besar dan lebih beragam untuk memperoleh gambaran yang lebih menyeluruh terhadap kinerja sistem. Diperlukan pula pendekatan tambahan yang memungkinkan sistem mengenali kata yang sangat berbeda secara penulisan namun memiliki makna yang sama, misalnya melalui integrasi pembelajaran mesin atau analisis semantik. Dengan pengembangan lebih lanjut, diharapkan sistem pencarian ini dapat semakin efektif, adaptif, dan memberikan hasil yang bermanfaat bagi pengguna dalam berbagai kondisi pencarian.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI :10.62411/tc.v24i2.12557

DAFTAR PUSTAKA

[1] R. A. S. Susi Rianti, "Perbandingan Algoritma Edit Distance, Levensthein Distance, Hamming Distance, Jaccard Similarity dalam Mendeteksi String Matching," vol. 10, no. 2, pp. 71–76, 2019.

[2] E. D. Oktaviyani, S. Christina, and D. Ronaldo, "Keywords Search Correction Using Damerau Levenshtein Distance Algorithm," *Conf. Senat. STT Adisutjipto Yogyakarta*, vol. 5, pp. 167–176, 2019, doi: 10.28989/senatik.v5i0.344.

[3] H. T. Sadiah, L. D. Iryani, T. A. Zuraiyah, Y. Wahyuni, and C. Zaddana, "Implementation of Levenshtein Distance Algorithm for Product Search Query Suggestions on Koro Pedang Edutourism E-Commerce," *J. Adv. Res. Appl. Sci. Eng. Technol.*, vol. 42, no. 2, pp. 188–196, 2024, doi: 10.37934/araset.42.2.188196.

[4] B. I. Muhammad, R. Nitia, and .L.B. Roland, "Penerapan Algoritma Levenshtein Distance untuk Mengoreksi Kesalahan Pengejaan pada Editor Teks," pp. 1–4, 2006.

[5] Wildannissa Pinasti and Lya Hulliyyatus Suadaa, "Named Entity Recognition in Statistical Dataset Search Queries," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 13, no. 3, pp. 171–177, 2024, doi: 10.22146/jnteti.v13i3.11580.

[6] F. W. Arsyta and R. E. Putra, "Penerapan Algoritma Damerau Levenshtein Distance Pada Pencarian Arsip Desa Jerukseger Pendukung ISO," *J. Informatics Comput. Sci.*, vol. 4, no. 4, pp. 423–435, 2023.

[7] V. Christanti Mawardi, F. Augusfian, J. Pragantha, and S. Bressan, "Spelling Correction Application with Damerau-Levenshtein Distance to Help Teachers Examine Typographical Error in Exam Test Scripts," *E3S Web Conf.*, vol. 188, 2020, doi: 10.1051/e3sconf/202018800027.

[8] D. Soundex, S. Pada, P. Ejaan, and K. Otomatis, "Analisa Perbandingan Algoritma Damerau-Levenshtein Distance," 2019.

[9] L. H. Iramani, "Implementasi Algoritma Damerau-Levenshtein Distance Untuk Pengoreksian Ejaan Bahasa Melayu Kampar (Ocu)," *J. Ekon. Vol. 18, Nomor 1 Maret201*, vol. 2, no. 1, pp. 1–73, 2021.

[10] B. V. Indriyono, "Kombinasi Damerau Levenshtein dan Jaro-Winkler Distance Untuk Koreksi Kata Bahasa Inggris," *J. Tek. Inform. dan Sist. Inf.*, vol. 6, no. 2, pp. 162–173, 2020, doi: 10.28932/jutisi.v6i2.2493.

[11] B. Irmawati, R. Dwiyansaputra, P. Studi, T. Informatika, F. Teknik, and U. Mataram, "Damerau-Levenshtein Distance Dan N-Gram," vol. 6, no. 1, pp. 257–263, 2024.

[12] K. J. B. Marcel Rino Batisya, "Implementasi Algoritma Levenshtein Distance Untuk Misspelled Word Pada Pencarian Lagu Melayu Marcel," *J. GEEJ*, vol. 7, no. 2, 2020.

[13] R. C. Rivaldi, T. D. Wismarini, J. T. Lomba, and J. Semarang, "Analisis Sentimen Pada Ulasan Produk Dengan Metode Natural Language Processing (NLP) (Studi Kasus Zalika Store 88 Shopee)," vol. 17, no. 1, pp. 120–128, 2024.

[14] A. Nuraminah and A. Ammar, "Damerau-Levenshtein Distance Algorithm Based on Abstract Syntax Tree to Detect Code Plagiarism," *Sci. J. Informatics*, vol. 11, no. 1, pp. 11–20, 2023, doi: 10.15294/sji.v11i1.48064.

[15] M. Ridwan and H. Arifin, "Aplikasi Algoritma Damerau – Levenshtein Distance dalam Mendeteksi Potensi Penipuan," 2019.

[16] A. P. Wibawa, P. Yuliawati, P. Santoso, R. Shalahuddin, and I. M. Wirawan, "Damerau Levenshtain Distance dengan Metode Empiris untuk Koreksi Ejaan Bahasa Indonesia," *Ilk. J. Ilm.*, vol. 12, no. 3, pp. 176–182, 2020, doi: 10.33096/ilkom.v12i3.600.176-182.

[17] S. Jain, K. R. Seeja, and R. Jindal, "A fuzzy ontology framework in information retrieval using semantic query expansion," *Int. J. Inf. Manag. Data Insights*, vol. 1, no. 1, p. 100009, 2021, doi: 10.1016/j.jjimei.2021.100009.

[18] L. Massai, "Evaluation of semantic relations impact in query expansion-based retrieval systems," *Knowledge-Based Syst.*, vol. 283, no. June 2023, p. 111183, 2024, doi:

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI :10.62411/tc.v24i2.12557

10.1016/j.knosys.2023.111183.

[19] R. Ilham, "Implementasi Algoritma Levenshtein Distance Untuk Fitur Pencarian Kata Pada Pengembangan Kamus Bahasa Aceh Berbasis Web," pp. 1–68, 2023.

[20] C. M. Suprapto, W. Syaifullah, J. Saputra, and P. S. Informatika, "J-Icon : Jurnal Informatika dan Komputer Prediksi Hasil Panen Budidaya Ikan Lele dari Mitra Panen J-Icon : Jurnal Informatika dan Komputer," vol. 12, no. 2, 2024, doi: 10.35508/jicon.v12i2.13187.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI :10.62411/tc.v24i2.12557