Implementasi Fitur Autocorrect pada Sistem Pencarian Judul Jurnal dengan Algoritma Levenshtein Distance

Tutuk Indriyani[1], Indah Maharoh[2], Nanang Fakhrur Rozi[3], Rinci Kembang Hapsari[4], S Nurmuslimah[5]*

[1,2,3,4,5]Institut Teknologi Adhi Tama Surabaya

**Penulis Korespondensi : tutuk@itats.ac.id*

ABSTRACT

When searching for a paper title, if a typing error occurs, often the desired title or abstract will not appear and we have to correct the text that has been entered as much as 15 words. Therefore, an autocorrect feature is needed in search engines to facilitate users if a typo occurs in the paper title or abstract. The autocorrect feature in the application functions as an automatic correction of incorrect words by immediately displaying the correct results. In this study, the Levenstein Distance Algorithm method was applied. The Levenshtein Distance algorithm is capable of producing quite high evaluation results, with 100% accuracy. The average response time for the three operations is approximately 0.22 seconds per operation, he said. This response time is quite fast, remaining less than 1 second.

Article History

Received : 09-06-2026
Revised : 17-07-2026
Accepted : 21-07-2026

Keywords

Fitur autocorrect
Proses pengecekan
Levenstein Distance

ABSTRAK

Pada pencarian judul makalah jika terjadi kesalahan pengetikan maka akan menyebabkan judul atau abstrak yang diinginkan sering kali tidak muncul dan kita harus mengkoreksi tulisan yang telah diinputkan dengan 15 kata. Proses pengecekan kesalahan pengetikan dengan cara manual akan membutuhkan suatu sumber pasti sebagai acuan bahwa kata tersebut memang salah dalam proses penulisannya. Oleh karena itu perlu adanya fitur autocorrect dalam mesin pencarian untuk memudahkan user apabila terjadi kesalahan dalam pengetikan judul atau abstrak makalah. Fitur autocorrect pada suatu aplikasi berfungsi sebagai otomatisasi pengoreksian kata yang salah dengan langsung menampilkan hasil yang benar. Peneliti menerapkan fitur autocorrect menggunakan metode Algoritma Levenstein Distance. Algoritma Levenshtein Distance ini mampu menghasilkan hasil evaluasi yang cukup tinggi yakni memiliki akurasi 100%, nilai rata-rata waktu respon ketiga operasinya adalah sekitar 0,22 detik per operasi katanya. Waktu respon ini termasuk cukup cepat karena masih < 1 detik.

PENDAHULUAN

Sistem pencarian makalah merupakan sarana utama bagi para peneliti, mahasiswa, dan akademisi untuk mengakses informasi terkini dengan cepat dan relevan. Pengertian cepat disini memiliki maksud bahwa mesin pencari bisa mengakses judul makalah ataupun jurnal yang benar-benar dicari oleh user dan tersusun berdasarkan nilai relevansi. Makalah-makalah tersebut biasanya ditampung dalam suatu wadah yang biasa disebut jurnal. Bagi para mahasiswa terutama mahasiswa baru yang berasal dari SMA sering kali masih asing mengenai metode-metode informatika. Pada bagian abstrak terkadang juga menggunakan bahasa inggris. Judul yang asing dan abstrak yang berbahasa inggris sering kali ada beberapa huruf yang terlewat sehingga ketika melakukan pencarian makalah yang diinginkan terkadang terjadi human error sehingga terdapat kesalahan pengetikan[1]. Kesalahan pengetikan itu biasanya disebabkan oleh beberapa faktor yaitu: letak huruf pada keyboard yang berdekatan, kesalahan karena kegagalan mekanis atau slip dari tangan atau jari, kesalahan yang disebabkan oleh ketidaksengajaan dan ketidaktahuan informasinya. Ketika pencarian judul makalah kesalahan pengetikan tersebut menyebabkan judul atau abstrak yang diinginkan sering kali tidak muncul dan kita harus mengkoreksi tulisan yang telah diinputkan[2].

ISSN: 2774-4345, DOI: 10.31284/j.kernel.2026.v7i1.9059

Proses pengecekan kesalahan pengetikan dengan cara manual akan membutuhkan suatu sumber pasti sebagai acuan bahwa kata tersebut memang salah dalam proses penulisannya. Oleh karena itu perlu adanya fitur *autocorrect* dalam mesin pencarian untuk memudahkan user apabila 77terjadi kesalahan dalam pengetikan judul atau abstrak makalah. Fitur autocorrect pada suatu aplikasi berfungsi sebagai otomatisasi pengoreksian kata yang salah dengan langsung menampilkan hasil yang benar[3]. Ada beberapa metode atau algoritma yang digunakan sebagai autocorrect yaitu metode Cosine Similarity, KMP, Jaro-Winkler, dan Algoritma Levenshtein Distance. Metode cosine similarity adalah metode untuk menghitung kemiripan antar dua objek yang diukur menggunakan keyword (kata kunci) dari dokumen serta dinyatakan dalam dua vector [4]. Algoritma KMP adalah algoritma untuk pencocokan string dengan membandingkan karakter pattern dan karakter teks yang diolah sampai terjadi ketidak cocokan untuk melakukan pergeseran. Algoritma Jaro Winkler Distance adalah sebuah algoritma yang digunakan untuk mengukur tingkat kemiripan antara dua string yang biasanya diterapkan dalam pendeteksian duplikat. Pengukurannya dilihat dari semakin tinggi nilai Jaro Winkler Distance untuk dua string maka semakin mirip tingkat kemiripan dua string tersebut [5]. Algoritma Levenshtein Distance adalah algoritama untuk mencari jarak antara dua kata yakni kata yang diketikkan oleh user dengan kata yang tersimpan pada database dengan menghitung jarak perbedaan antara kedua string dalam bentuk matriks. Cara kerja algoritma Levenshtein Distance yaitu menghitung jarak antara kedua string kemudian mencari jumlah minimum dari operasi perubahan untuk merubah dari string A menjadi string B. Adapun perhitungan direpsentasikan dalam bentuk tabel perhitungan Levenshtein Distance, di mana nilai terakhir pada pojok kanan bawah pada matriks merupakan nilai akhir dari jarak kedua string[6].

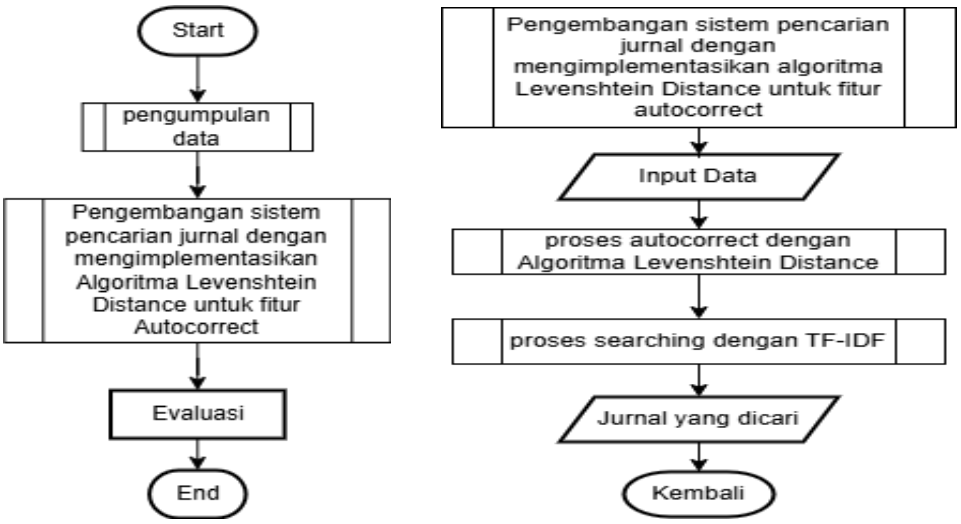
Pada penelitian terkait yaitu tentang pengoreksian kesalahan penulisan kalimat menyatakan bahwa Algoritma Levenshtein Distance menghasilkan nilai akurasi sebeser 93.5% sedangkan hasil untuk algoritma cosine similarity mendapatkan hasil sebesar 77.9% pada pengoreksian penulisan tersebut dengan 12 pengujian. Hal ini menyatakan bahwa Algoritma Levenshtein Distance lebih baik daripada algoritma Cosine Similarity[6]. Adapun penelitian yang terkait yaitu penelitian mengenai perbandingan Knuth Morris Pratt (KMP) dan Algoritma Levenshtein Distance menyatakan bahwa algoritma levenshtein distance lebih direkomendasikan daripada algoritma KMP dalam hal kecepatan dan akurasi pencarian data member. Pada penelitian terkait lainnya yakni tentang perbandingan Algoritma Levenshtein Distance dan Jaro Winkler pada pencarian dokumen didapatkan bahwa hasil rata-rata perhitungan Jaro Winkler adalah 0,0052 detik, sedangkan untuk perhitungan Levensthein Distance adalah 0,00028[4]. Hal ini menujukkan Algoritma Levenshtein Distance lebih cepat waktu penanganannya dari pada Algoritma Jaro Winkler.

Adapula penelitian terkait lainnya yaitu tentang penerapan levenshtein distance pada kamus istilah militer TNI menyatakan bahwa dapat mempermudah proses pencarian istilah militer TNI apabila terjadi kesalahan pengetikan pada kolom pencarian. Penelitian terkait lainnya yaitu menyimpulkan bahwa hasil responden yang berjumlah 85 orang menyatakan 23,53% sangat setuju dan 60% setuju bahwa algoritma Levenshtein Distance yang diterapkan dapat meminimalisir kesalahan ejaan pada pencarian lagu melayu karya Arif Putra[7]. Penelitian terkait lainnya yakni tentang autocomplete untuk pencarian judul skripsi menggunakan levenshtein distance menyatakan bahwa untuk pencarian single target memiliki keakuratan 75% dan multi target dengan 2 kata memiliki keakuratan 50%[8]. Adapun penelitian terkait lainnya yaitu menunjukkan bahwa dengan menggunakan algoritma Levenshtein Distance single target sebesar 75% dan multi target sebesar 87%. Pada penelitian yang menyatakan bahwa hasil pengujian implementasi algoritma pada drugs e-dictionary dihasilkan nilai akurasi, recall dan precision sebesar 90%[9].

Tujuan dari penelitian ini adalah untuk mengembangkan fitur autocorrect dengan Algoritma Levenshtein Distance pada pencarian jurnal dan mengukur fitur autocorrect dengan Algoritma Levenshtein Distance pada pencarian jurnal. Berdasarkan hal yang sudah dibahas sebelumnya, peneliti menerapkan menggunakan metode Algoritma Levenstein Distance pada sistem mesin pencarian jurnal dengan harapan untuk mempermudah user ketika melakukan pencarian judul jurnal atau abstrak yang diinginkan meskipun mungkin secara tidak sengaja ada sedikit kesalahan pengetikan ketika memasukan penginputannya.

METODE

Pada tahapan ini peneliti akan membuat sistem pencarian jurnal yang mana user akan melakukan proses input data. Pada proses penginputan data ini diharapkan mampu melakukan autocorrect pada kata yang salah ketik. Lalu mengklik tombol searching dan akan memunculkan jurnal yang dicari jika kata kunci yang diinputkan terdapat pada database. Pengembangan fitur autocorrect dengan algoritma Levenshtein Distance ini yang akan membenarkan setiap kata yang diinputkan oleh user agar ketika melakukan proses searching maka jurnal yang diinginkan tetap akan dimunculkan. Pembuatan diagram alir yang benar ditunjukkan seperti pada Gambar 1, yaitu Flowchart Alur Sitem Penelitian, b. Implementasi Sistem Autocorrect.

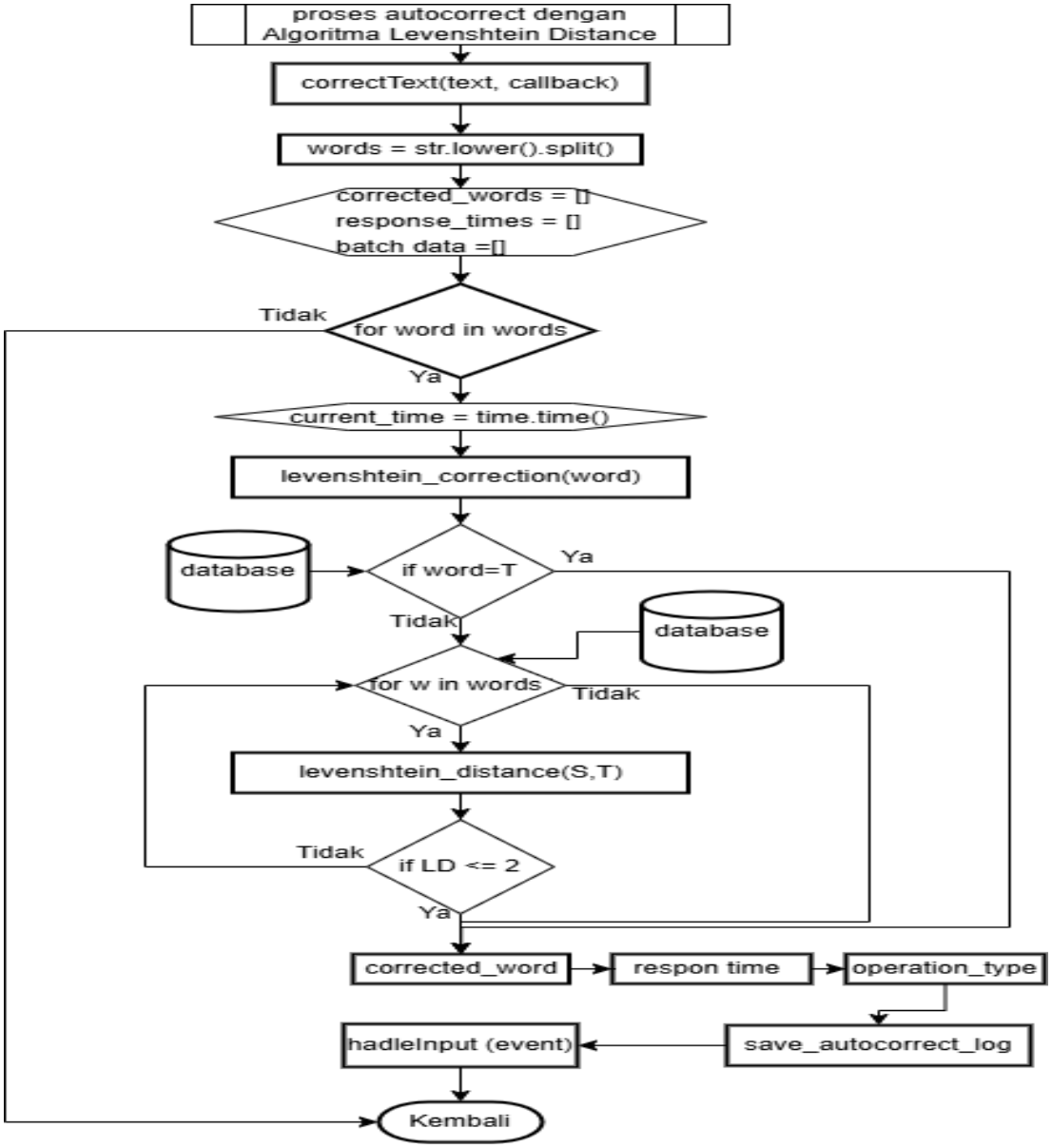


(a) (b)

Gambar 1. (a) Flowchart Alur Sitem Penelitian, (b) Implementasi Sistem Autocorrect.

Pada tahap (a) dilakukan proses pengumpulan data. Pengumpulan data disini adalah melakukan scraping data jurnal Komponen data yang discraping yaitu data judul, data abstrak dan url dari kedua jurnal tersebut. Setelah mendapatkan komponen data tadi, barulah melakukan pre-prosesing data pada data judul dan data abstraknya untuk dijadikan kata kunci. Kata kunci inilah yang akan dijadikan sebagai sumber data target (acuan) dari metode Levenshtein Distance,kemudian dilakukan implementasi system autocorrect dan di evaluasi hasilnya. Pada bagian (b) yaitu input data merupakan langkah awal pada tahapan ini adalah user menginputkan data. Data disini bisa berupa kata atau beberapa kata atau kalimat. Proses search ini menggunakan searching pada query ditambah dengan pembobotan TF-IDF. Setelah itu melakukan proses pencarian judul jurnal yang diinginkan proses autocorrect pada setiap kata yang telah diinputkan dapat ditunjukkan pada Gambar 2, yaitu Flowchart Proses Autocorrect

ISSN: 2774-4345, DOI: 10.31284/j.kernel.2026.v7i1.9059



Gambar 2. Flowchart Proses Autocorrect.

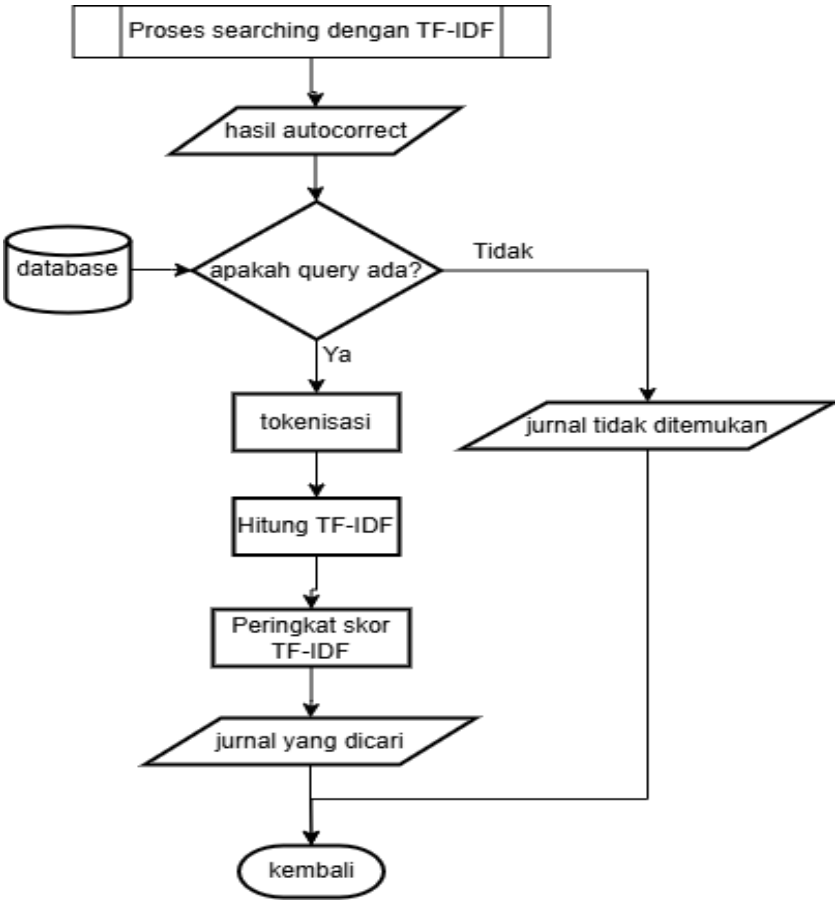
Pada tahap kedua ini melakukan proses Autocorrect dengan contoh teks yang digunakan adalah clasifikasi naïve bayess maka langkah- langkah yang dilakukan sesuai dengan algoritma tersebut adalalah:

Words = str.lower().split() tahapan ini memecah inputan menjadi huruf kecil dan membuatnya menjadi token yang akan disimpan dalam array. Contoh : Kata ke-1 : clasifikasi , Kata ke-2 : naiv, Kata ke-3 : bayess, maka hasil array = [clasifikasi, naiv, bayess]. Kemudian dilakukan:

1. Inisialisasi array, Inisialisasi array corrected_word untuk menyimpan hasil kata yang telah diautocorrect, respon_time untuk menyimpan aktu repson yag diperlukan untuk proses autocorrect, batch data untuk menyimpan hasil proses autocorrect seperti input, hasil, respon time, cuurect_time yang nantinya akan disimpan di autocorrect log.
2. For w in words, melakukan looping inputan mulai dari kata pertama hingga kata inputan terakhir untuk melakukan proses function autocorrect yaitu proses 4. Apabila kondisi for sudah tidak memenuhi maka proses selesai (dilanjutkan ke tahap TF-IDF).
3. Current_time = time.time(), menginisialisasikan waktu mulai sebelum dilakukan proses autocorrect.

4. Melakukan function levenshtein_correction(word), menginisialisasikan waktu mulai sebelum dilakukan proses autocorrect.

Proses search ini menggunakan searching pada query ditambah dengan pembobotan TF-IDF dengan library TFidfVectorizer dan sklearn dapat ditunjukkan pada Gambar 3, yaitu Flowchart proses searching dengan TF-IDF.



Gambar 3. Flowchart proses searching dengan TF-IDF

Pada tahap ketiga ini melakukan proses searching dengan TF-IDF yaitu Ambil query yang sudah diautoccrect dengan Contoh : 'klasifikasi naïve bayes', langkah langkah yang dilakukan:

1. Membandingkan query dengan database yaitu melakukan proses searching jika ada query yang sama dengan database maka jurnal akan ditambahkan. Jika ada maka lanjut ke tahap selanjutnya. Jika tida ada maka akan menghasilkan output data jurnal tidak ditemukan.
2. Tokenisasi yaitu melakukan pembagian kata pada setiap data jurnal yang telah ditemukan.
   Contoh :
   a. D1 = [analisa, perbandingan, algoritma, dt, naïve, bayes, dalam prediksi, penerimaan, kredit, motor]
   b. D2 = [klasifikasi, naïve, bayes, pada, analisis, sentiment, atas, penolakan, dibukanya, larangan, ekspor, benih, lobster].
   c. D3 = [analisis, sentiment, tanggapan, masyarakat, terhadap pembelajaran, daring, di, masa, pandemic, covid, melalui, social, media, twitter, menggunakan, klasifikasi,naïve bayes]
3. Menghitung pembobotan TF-IDF yang ditunjukkan pada persamaan (1).

$$Wdt = TFdt \times IDFt \quad (1)$$

ISSN: 2774-4345, DOI: 10.31284/j.kernel.2026.v7i1.9059

Dimana:
Wdt = bobot dokumen ked terhadap kata ket
TFdt = banyaknya kata yang dicari pada sebuah dokumen
IDFt = Inversed Document Frequency (log (N/df)
N = total dokumen
df = banyak dokumen yang mengandung kata yang dicari.

Pada tahapan keempat yaitu melakukan proses evaluasi. Proses ini dilakukan untuk mengevaluasi seberapa tinggi tingkat akurasi pada fitur autocorrect yang telah dibangun. Ada beberapa skenario pengujian dalam penelitian ini:

1. Pengujian untuk operasi penghapusan atau delete, pada skenario ini digunakan untuk melihat respon time pada sistem untuk autocorrect apabila terjadi penghapusan karakter pada kata.
2. Pengujian untuk operasi penambahan atau insert, pada skenario ini digunakan untuk melihat respon time pada sistem untuk autocorrect apabila terjadi penambahan karakter pada kata.
3. Pengujian untuk operasi pengubahan atau substitute, pada pengujian ini digunakan untuk melihat respon time pada sistem untuk autocorrect apabila terjadi pengubahan karakter pada kata.

Pada setiap operasi nantinya akan dihitung waktu respon dengan rumus seperti pada persamaan 2[10].

$$rata - rata\ waktu\ respon = \frac{\sum_{k=1}^{n} respon\ time}{total\ data} \quad (2)$$

Dengan :
n = total data pengujian = total data
respon time = waktu respon autocorrect yang didapat dari waktu selesai – waktu mu

Kemudian setelah didapatkan masing-masing rata-rata waktu respon tiap operasinya maka dilakukan lagi perhitungan rata-rata waktu respon semua operasi. Rata-rata waktu respon inilah yang akan digunakan untuk mengukur fitur autocorrect system ini yang ditunjukkan pada persamaan 3[11].

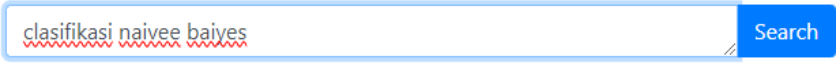
$$rata - rata\ waktu\ respon\ fitur\ autocorrect = \frac{\sum_{i=1}^{j} rata{-}rata\ respon\ time}{total\ operasi\ outocorrect} \quad (3)$$

Dengan :
j = jumlah dari rata-rata respon time semua operasi autocorrect
total operasi autocorrect = 3 yaitu hapus, tambah, rubah.

HASIL DAN PEMBAHASAN

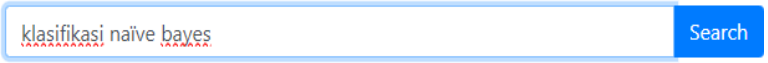
Pembahasan Proses Input data, Autocorrect dan Searching.

Input data , langkah awal pada tahapan ini adalah user menginputkan data. Data disini bisa berupa kata atau beberapa kata atau kalimat. Tampilan inputan : clasifikasi naivee baiyes ditunjukkan pada Gambar 4, yaitu tampilan Inputan,



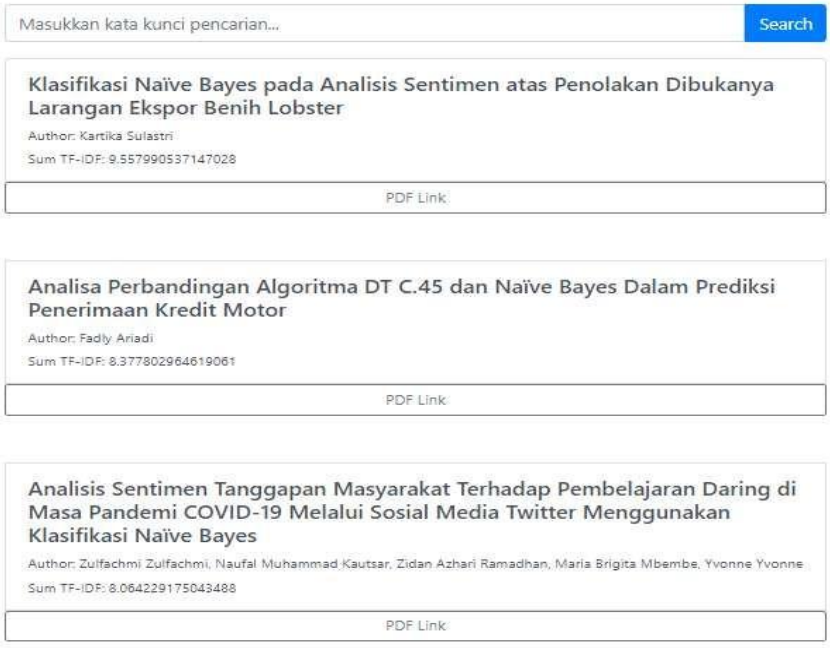
Gambar 4 Tampilan Inputan

Proses autocorrect dengan algoritma Levenshtein Distance, pada bagian ini akan dilakukan proses autocorrect pada setiap kata yang telah diinputkan.Tampilan hasil autocorrect ditunjukan pada Gambar 5, yaitu Tampilan hasil Autocorr



Gambar 5 Tampilan Hasil Autocorrec

Proses searching, membuat proses searching untuk inputan yang telah dibenarkan. Proses search ini menggunakan searching pada query ditambah dengan pembobotan TF-IDF agar bisa menampilkan jurnal yang sesuai dengan inputan serta beberapa jurnal yang relevan. Pada bagian ini digunakan untuk menampilkan data_jurnal sesuai hasil searching dengan TF-IDF mulai dari judul, abstrak, author, bobot w (tfidf_score) dan link_pdf. Untuk tampilan hasil searching dapat dilihat pada Gambar 6, yaitu Tampilan Hasil Searching dengan TF-IDF



Gambar 6. Tampilan Hasil Searching dengan TF-IDF

Menghasilkan judul jurnal yang relevan sesuai dengan inputan yang telah dibenarkan. Pada inputan yang telah diberikan ternyata ada tujuh jurnal yang relevan namun 3 yang paling mirip adalah hasil pada gambar 6 di atas.

Pengujian dan Evaluasi

Pada tahapan keempat yaitu melakukan proses pengujian dan evaluasi. Proses ini dilakukan untuk melakukan beberapa pengujian dan mengevaluasi seberapa tinggi akurasi pada fitur autocorrect yang telah dibangun. Ada beberapa hal yang perlu dilakukan yakni melakukan 3 skenario pengujian yaitu berdasarkan operasi autocorrectnya.

Ada 3 pengujian dalam penelitian ini:

1. *Pengujian untuk operasi penghapusan atau delete*

Pada pengujian ini digunakan untuk melihat respon time pada sistem untuk autocorrect apabila terjadi penghapusan karakter pada kata. Ditunjukkan pada Tabel 1. Pada hasil penghapusan teks memiliki akurasi 100% sesuai, dengan merujuk pada persamaan 2 dan 3, maka pada Table 1 dapat

ISSN: 2774-4345, DOI: 10.31284/j.kernel.2026.v7i1.9059

diketahui bahwa rata-rata respon timenya adalah 6,7 sehingga dapat diketahui bahwa rata-rata waktu responnya adalah sekitar 0,22 detik per kata pada operasi penghapusan

Table 1. Pengujian Operasi Penghapusan Teks

id	inputan	hasil	response_time (s)	operation_type	created_at
1	peranncangan	perancangan	0,298	delete n	02/11/2024 19:37:42
2	pointt	point	0,266	delete t	02/11/2024 19:37:42
3	hurruf	huruf	0,140	delete r	02/11/2024 19:37:42
4	anndroid	android	0,031	delete n	02/11/2024 19:37:42
5	sisstem	sistem	0,440	delete s	02/11/2024 19:37:42
6	pembayyaran	pembayaran	0,383	delete y	02/11/2024 19:37:42
7	sekolahh	sekolah	0,408	delete h	02/11/2024 19:37:42
8	baiyes	bayes	0,047	delete i	02/11/2024 19:37:42
9	senttimen	sentimen	0,491	delete t	02/11/2024 19:37:42
10	atass	atas	0,073	delete s	02/11/2024 19:37:42
11	penolakann	penolakan	0,390	delete n	02/11/2024 19:37:42
12	eksspor	ekspor	0,156	delete s	02/11/2024 19:37:42
13	bennih	benih	0,077	delete n	02/11/2024 19:37:42
14	lobsteer	lobster	0,235	delete e	02/11/2024 19:37:42
15	sisteem	sistem	0,378	delete e	02/11/2024 19:37:42

2. *Pengujian untuk operasi penambahan (insert).*

Pada pengujian operasi penambahan ini digunakan untuk melihat respon time pada sistm untuk autocorrect apabila terjadi penambahan karakter pada kata dapat ditunjukkan pada Tabel 2.

Tabel 2. Pengujian penambahan Teks

Id	inputan	hasil	response_time (s)	operation_type	created_at
1	analis	analisa	0,038	insert a	02/11/2024 19:37:42
2	suplier	supplier	0,389	insert p	02/11/2024 19:37:42
3	weigting	weighting	0,550	insert h	02/11/2024 19:37:42
4	resorce	resource	0,420	insert u	02/11/2024 19:37:42
5	maitreawira	maitreyawira	0,223	insert y	02/11/2024 19:37:42
6	serqual	servqual	0,374	insert v	02/11/2024 19:37:42
7	satisfation	satisfaction	0,439	insert c	02/11/2024 19:37:42
8	implemenasi	implementasi	0,140	insert t	02/11/2024 19:37:42
9	programing	programming	0,457	insert m	02/11/2024 19:37:42
10	aplication	application	0,077	insert p	02/11/2024 19:37:42
11	ligtweight	lightweight	0,279	insert h	02/11/2024 19:37:42
12	acces	access	0,031	insert s	02/11/2024 19:37:42
13	surveilence	surveillence	0,456	insert l	02/11/2024 19:37:42
14	terhadp	terhadap	0,406	insert a	02/11/2024 19:37:42
15	suppot	support	0,296	insert r	02/11/2024 19:37:42

Pada hasil penambahan teks memiliki akurasi 100% sesuai, dengan merujuk pada persamaan 2 dan 3, maka pada Table 2 dapat diketahui bahwa rata-rata respon timenya adalah 8,3 sehingga

dapat diketahui bahwa rata-rata waktu responnya adalah sekitar 0,27 detik per kata pada operasi penambahan data.

3. *Pengujian untuk operasi pengubahan atau substitute*

Pada pengujian ini digunakan untuk melihat respon time pada sistem untuk autocorrect apabila terjadi pengubahan karakter pada teks dapat ditunjukkan pada Tabel 3.

Tabel 3. Pengujian Operasi Perubahan Teks.

id	inputan	hasil	response_time (s)	operation_type	created_at
1	clasifikasi	klasifikasi	0,194	substitute c -> k	02/11/2024 19:37:42
2	divukanya	dibukanya	0,156	substitute v -> b	02/11/2024 19:37:42
3	selexsi	seleksi	0,333	substitute x -> k	02/11/2024 19:37:42
4	propinsi	provinsi	0,687	substitute p -> v	02/11/2024 19:37:42
5	awaal	awal	0,048	substitute a -> l	02/11/2024 19:37:42
6	manejemen	manajemen	0,313	substitute e -> a	02/11/2024 19:37:42
7	multimeria	multimedia	0,367	substitute r -> d	02/11/2024 19:37:42
8	fikih	fiqih	0,106	substitute k -> q	02/11/2024 19:37:42
9	peneravan	penerapan	0,39	substitute v -> p	02/11/2024 19:37:42
10	directori	directory	0,126	substitute i -> y	02/11/2024 19:37:42
11	pengujisn	pengujian	0,406	substitute s -> a	02/11/2024 19:37:42
12	boundari	boundary	0,08	substitute i -> y	02/11/2024 19:37:42
13	maturiti	maturity	0,281	substitute i -> y	02/11/2024 19:37:42
14	objec.	object	0,204	substitute . -> t	02/11/2024 19:37:42
15	artivicial	artificial	0,047	substitute v -> f	02/11/2024 19:37:42

Dengan merujuk pada persamaan 2 dan 3, maka pada Table 3 dapat diketahui bahwa rata-rata respon timenya adalah 5,9 sehingga dapat diketahui bahwa rata-rata waktu responnya adalah sekitar 0,2 detik per kata pada operasi perubahan teks. Dapat diketahui bahwa operasi substitusi mempunyai waktu respon yang lebih cepat dari pada 2 operasi autocorrect yang lain yaitu sekitar 0,2 detik serta pencarian datanya 100% ditemukan, jika data kata tersebut terdapat dalam database. Hal ini dikarenakan ketika proses substitusi maka perhitungannya hanya langsung mengambil dari diagonal kiri atasnya pada matriks tanpa perlu menghitung nilai minimal dari ketiga matriks terdekat.

KESIMPULAN

Pada Sistem pencarian jurnal yang dilengkapi fitur autocorrect menggunakan algoritma Levenshtein Distance dapat mempermudah user apabila terjadi kesalahan inputannya sehingga judul jurnal yang diinginkan akan tetap muncul jika datanya ada di database. Algoritma Levenshtein Distance ternyata mempunyai respon time fitur autocorrect yang cukup cepat apabila terjadi operasi substitusi atau pengubahan yakni hanya sekitar 0,2 detik. Hal ini dikarenakan operasi perhitungannya hanya langsung mensubstitusi diagonal kiri atas pada matriks tanpa perlu mencari nilai minimal matriks terdekat. Algoritma Levenshtein Distance ini mampu menghasilkan hasil evaluasi yang cukup tinggi yakni memiliki akurasi pencarian data dengan 100% ketemu, nilai rata-rata waktu respon ketiga operasinya adalah sekitar 0,22 detik per operasi katanya. Waktu respon ini termasuk cukup cepat karena masih < 1 detik.

DAFTAR PUSTAKA

[1] S. Chen, M. E. Asar, J. Yagoobi, and C. Shao, "RLGBS: Reinforcement Learning-Guided Beam Search for process optimization in a paper machine dryer section," *Comput. Ind.*, vol. 172, Nov. 2025, doi: 10.1016/j.compind.2025.104351.

[2] L. Q. Ma and J. Wang, "Creating captivating and searchable titles in technical writing: Practical guides and sample analyses," Jan. 01, 2026, *Elsevier B.V.* doi: 10.1016/j.seh.2025.100188.

[3] C. Moukrim, T. Abderrahim, E. H. Benlahmer, and A. Tarik, "An innovative approach to autocorrecting grammatical errors in Arabic texts," *Journal of King Saud University - Computer and Information Sciences*, vol. 33, no. 4, pp. 476–488, May 2021, doi: 10.1016/j.jksuci.2019.02.005.

[4] S. Hajbi, O. Amezian, N. El Moukhi, R. Korchiyne, and Y. Chihab, "Moroccan Arabizi-to-Arabic conversion using rule-based transliteration and weighted Levenshtein algorithm," *Sci. Afr.*, vol. 23, Mar. 2024, doi: 10.1016/j.sciaf.2024.e02073.

[5] V. Junnila, T. Laihonen, and T. Lehtilä, "Levenshtein's sequence reconstruction problem and results for larger alphabet sizes," *Theor. Comput. Sci.*, vol. 1045, Aug. 2025, doi: 10.1016/j.tcs.2025.115279.

[6] H. C. Lind-Combs, T. Bent, R. F. Holt, C. G. Clopper, and E. Brown, "Comparing Levenshtein distance and dynamic time warping in predicting listeners' judgments of accent distance," *Speech Commun.*, vol. 155, Nov. 2023, doi: 10.1016/j.specom.2023.102987.

[7] D. Wahyud1, R. Rosnelly, and R. Dewi, "Penerapan Algoritma Levenshtein Pada Aplikasi Kamus Istilah Militer Tni Berbasis Android."

[8] S. Ahmmed, M. R. H. Mondal, M. R. Mia, M. Adibuzzaman, A. S. M. L. Hoque, and S. I. Ahamed, "A novel approach for standardizing clinical laboratory categorical test results using machine learning and string distance similarity," *Heliyon*, vol. 9, no. 11, Nov. 2023, doi: 10.1016/j.heliyon.2023.e21523.

[9] Y. N. Daniati, I. A. Zukarnain, and K. Nurfitri, "PADA SISTEM PENCARIAN DATA BUKU BERBASIS WEB," 2022. [Online]. Available: http://studentjournal.umpo.ac.id/index.php/komputek

[10] T. Indriyani, M. Kurniawan, G. Yuliastuti, C. Prabiantissa, and R. Kembang, "An Improve KNN Method for Classification of Sexually Transmitted Diseases," *2023 Sixth International Conference on Vocational Education and Electrical Engineering (ICVEE)*, vol. 3, pp. 315–319, Sep. 2023.

[11] T. Indriyani, S. Nurmuslimah, A. Taufiqurrahman, R. K. Hapsari, C. N. Prabiantissa, and A. Rachmad, "Steganography on Color Images Using Least Significant Bit (LSB) Method," 2023, pp. 39–48. doi: 10.2991/978-94-6463-174-6_5.