Sistem Deteksi Kesalahan Penulisan Karya Ilmiah Dengan Metode Natural Language Processing Pada Ejaan KBBI

Scientific Writing Error Detection System Using Natural Language Processing Method on KBBI Spelling

Bagus Dwi Prasetyo*[1], Wibowo Harry Sugiharto[2], Muhammad Imam Ghozali[3]
Program Studi Teknik Informatika, Fakultas Teknik, Universitas Muria Kudus
E-mail : 202151074@std.umk.ac.id[1], wibowo.harrys@umk.ac.id[2] , imam.ghozali@umk.ac.id[3]*
**Corresponding author*
Received 2 December 2025; Revised 27 December 2025; Accepted 6 January 2026

Abstrak - Penulisan karya ilmiah yang sesuai dengan kaidah Bahasa Indonesia merupakan hal penting dalam dunia akademik. Namun, masih banyak ditemui kesalahan ejaan dan penggunaan kata tidak baku dalam dokumen ilmiah. Penelitian ini bertujuan untuk merancang dan mengembangkan sistem deteksi kesalahan penulisan karya ilmiah menggunakan metode *Natural Language Processing* (NLP) berbasis ejaan Kamus Besar Bahasa Indonesia (KBBI). Sistem dibangun menggunakan pendekatan *Waterfall* dengan implementasi berbasis web menggunakan Python dan library NLP seperti spaCy, Sastrawi, dan NLTK. Proses deteksi melibatkan tahapan *case folding*, tokenisasi, *stemming*, *Named Entity Recognition* (NER), pencocokan dengan database KBBI, serta koreksi ejaan menggunakan algoritma *Levenshtein Distance*. Evaluasi sistem dilakukan menggunakan dataset karya ilmiah yang telah dianotasi secara manual dan menghasilkan akurasi 92%, presisi 92%, recall 93%, dan F1-score 92%. Hasil tersebut menunjukkan bahwa sistem mampu mendeteksi kesalahan ejaan dengan baik dan memberikan saran perbaikan secara efektif. Penelitian ini diharapkan dapat membantu meningkatkan kualitas penulisan akademik di Indonesia.

Kata Kunci – Natural Language Processing, KBBI, Deteksi Ejaan, Karya Ilmiah

Abstract - *Writing scientific papers in accordance with Indonesian language rules is crucial in the academic world. However, many spelling errors and non-standard word usage remain in scientific documents. This study aims to design and develop a system for detecting errors in scientific papers using Natural Language Processing (NLP) methods based on the spelling of the Big Indonesian Dictionary (KBBI). The system was built using a Waterfall approach with a web-based implementation using Python and NLP libraries such as spaCy, Sastrawi, and NLTK. The detection process involves the following stages: case folding, tokenization, stemming, Named Entity Recognition (NER), matching against the KBBI database, and spelling correction using the Levenshtein Distance algorithm. System evaluation was conducted using a manually annotated scientific paper dataset, yielding 92% accuracy, 92% precision, 93% recall, and 92% F1-score. These results indicate that the system can detect spelling errors effectively and provide effective correction suggestions. This research is expected to help improve the quality of academic writing in Indonesia.*

Keywords – *Natural Language Processing, KBBI, Spelling Detection, Scientific Papers*

1. PENDAHULUAN

Penulisan karya ilmiah merupakan elemen penting dalam dunia akademik karena berfungsi sebagai sarana penyampaian hasil pemikiran, penelitian, dan inovasi secara sistematis dan terstruktur. Karya ilmiah seperti artikel, makalah, dan laporan penelitian harus disusun dengan mengikuti kaidah Bahasa Indonesia yang baik dan benar agar informasi yang disampaikan dapat dipahami secara tepat oleh pembaca. Berdasarkan laporan Indikator Iptek, Riset, dan

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15219

Inovasi Indonesia 2024 (IIRI 2024) yang diterbitkan oleh Badan Riset dan Inovasi Nasional (BRIN), Indonesia menghasilkan sebanyak 13.975 dokumen publikasi ilmiah internasional sepanjang tahun 2023 [1]. Tingginya jumlah publikasi tersebut meningkatkan potensi terjadinya kesalahan penulisan, khususnya pada aspek ejaan dan penggunaan kata baku.

Beberapa penelitian menunjukkan bahwa kesalahan ejaan masih sering ditemukan dalam karya ilmiah mahasiswa. Khoyriyah et al. (2024) melaporkan bahwa banyak mahasiswa belum memperhatikan kaidah ejaan Bahasa Indonesia secara optimal, sehingga sering terjadi kesalahan penulisan kata tidak baku dan struktur kalimat yang kurang tepat [2]. Temuan ini sejalan dengan penelitian Tarigan et al. yang mengidentifikasi berbagai kesalahan penggunaan ejaan Bahasa Indonesia, terutama dalam pemilihan kata baku pada artikel ilmiah mahasiswa [3]. Kondisi tersebut menunjukkan perlunya solusi yang mampu membantu penulis dalam meningkatkan kualitas kebahasaan karya ilmiah. Seiring perkembangan teknologi, pemanfaatan *Natural Language Processing* (NLP) menjadi solusi yang relevan dalam mengatasi permasalahan tersebut. NLP merupakan sekumpulan metode yang memungkinkan sistem komputer untuk mengolah dan memahami bahasa alami manusia [4]. Dalam konteks penulisan akademik, NLP dapat dimanfaatkan untuk membangun sistem pendeteksi kesalahan penulisan secara otomatis guna meningkatkan efisiensi dan akurasi proses penulisan.

Penelitian terdahulu terkait deteksi kesalahan penulisan Bahasa Indonesia dapat diklasifikasikan berdasarkan pendekatan yang digunakan. Pendekatan yang paling umum adalah *dictionary lookup* dengan acuan Kamus Besar Bahasa Indonesia (KBBI). Ledjap et al. mengembangkan sistem pengecekan ejaan berbasis NLP dan KBBI yang mampu mencapai akurasi sebesar 81,64% [5]. Penelitian serupa dilakukan oleh Misbullah et al. (2022) yang menerapkan metode *dictionary lookup* untuk mendeteksi kata tidak baku pada tugas akhir mahasiswa. Hasil penelitian tersebut menunjukkan bahwa rata-rata kesalahan dalam setiap dokumen adalah sebesar 2,76%, namun presisi yang diperoleh masih relatif rendah, yaitu 28,71% [6]. Hal ini mengindikasikan bahwa pendekatan *dictionary lookup* efektif dalam mendeteksi kata tidak baku, tetapi masih memiliki keterbatasan dalam akurasi pendeteksian secara keseluruhan.

Selain *dictionary lookup*, beberapa penelitian memanfaatkan pendekatan berbasis kemiripan string (*string similarity*). Rohim dan Purmasari mengombinasikan algoritma *Soundex* dan *Damerau-Levenshtein* untuk pengembangan sistem koreksi ejaan, dengan hasil evaluasi menunjukkan nilai *recall* sebesar 100% dan *precision* sebesar 59,5% pada tahap deteksi kesalahan [7]. Wildan et al. (2024) menggunakan algoritma *Levenshtein Distance* dalam pengembangan fitur *spell checker* dan memperoleh tingkat akurasi deteksi dan koreksi ejaan antara 80% hingga 90% [8]. Sementara itu, Risang Agung Samudro et al. (2024) menerapkan algoritma *Jaro-Winkler Distance* dan berhasil memberikan skor koreksi lebih dari 0,94 pada data uji, menunjukkan kemampuan algoritma tersebut dalam memberikan saran perbaikan yang akurat [9].

Pendekatan lain yang juga digunakan adalah metode statistik berbasis *N-Gram*. Hartina (2020) mengembangkan sistem deteksi kesalahan pengetikan kata nonbaku menggunakan metode *N-Gram* dan melaporkan tingkat keberhasilan sebesar 80% pada dokumen karya ilmiah [10]. Dwi dan Qoiryah (2021) mengombinasikan *Jaro-Winkler Distance* dan *N-Gram* untuk mendeteksi serta memprediksi perbaikan kesalahan penulisan, dengan akurasi tertinggi mencapai 85,7%, meskipun performanya bervariasi tergantung pada jenis data uji [11].

Seiring berkembangnya teknik NLP, pendekatan deep learning berbasis konteks mulai diterapkan. Halim dan Nurhaida (2024) mengembangkan sistem deteksi dan koreksi kesalahan penulisan Bahasa Indonesia berbasis *Long Short-Term Memory* (LSTM) dan berhasil mencapai akurasi sebesar 93% pada proses pelatihan dan validasi [12]. Meskipun menunjukkan performa yang tinggi, pendekatan ini membutuhkan sumber daya komputasi dan dataset yang besar, sehingga kurang optimal untuk diimplementasikan pada sistem ringan berbasis web.

Dari sisi implementasi sistem, beberapa penelitian berfokus pada pengembangan aplikasi. Saputra et al. (2021) merancang sistem berbasis web untuk mendeteksi kata tidak baku pada dokumen Word (.docx) dengan tahapan tokenisasi, *case folding*, *stemming*, dan *dictionary lookup* [13]. Fahriefazza mengembangkan aplikasi berbasis web untuk mendeteksi kesalahan penulisan

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15219

dalam naskah skripsi menggunakan metode *full-text indexing*, *tokenizing* dan *cleansing* [14], sedangkan Zulfadli Sultan merancang aplikasi serupa berbasis Android yang mampu mengenali kesalahan penghapusan, penyisipan, dan penggantian karakter [15]. Namun, sebagian besar penelitian tersebut lebih menitikberatkan pada aspek implementasi sistem daripada evaluasi mendalam terhadap performa algoritmik.

Pada ranah internasional, Desta dan Lehal (2023) mengembangkan sistem otomatis untuk deteksi dan koreksi kesalahan ejaan bahasa Tigrigna menggunakan pendekatan hybrid yang menggabungkan *Jaccard Bigram, Edit Distance,* dan probabilitas kata. Sistem ini menunjukkan performa yang sangat tinggi dengan nilai *F-measure* sebesar 98,85% untuk deteksi dan akurasi 95,36% untuk koreksi [16]. Meskipun hasilnya unggul, pendekatan tersebut belum tentu dapat langsung diadaptasi pada Bahasa Indonesia yang memiliki karakteristik morfologi yang berbeda.

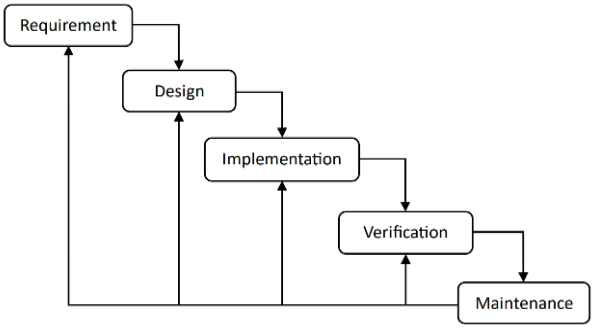
Berdasarkan kajian terhadap penelitian terdahulu, dapat disimpulkan bahwa sebagian besar sistem pendeteksi kesalahan penulisan Bahasa Indonesia masih berfokus pada satu pendekatan tertentu, seperti *dictionary lookup, edit distance* atau model pembelajaran mesin secara terpisah. Penelitian ini bertujuan mengintegrasikan pendekatan NLP berbasis KBBI ke dalam aplikasi web yang dirancang khusus untuk konteks penulisan karya ilmiah. Sistem ini dilengkapi dengan tahapan prapemrosesan yang disesuaikan dengan karakteristik Bahasa Indonesia formal, serta difokuskan pada pendeteksian kesalahan ejaan dan penggunaan kata tidak baku dalam dokumen akademik. Dengan demikian, sistem yang dikembangkan diharapkan lebih relevan dan aplikatif bagi mahasiswa dan peneliti di lingkungan pendidikan tinggi di Indonesia.

2. METODE PENELITIAN

Penelitian ini menerapkan pendekatan rekayasa perangkat lunak aplikatif dengan fokus pada perancangan sistem pendeteksi kesalahan penulisan karya ilmiah berbasis *Natural Language Processing* (NLP) sesuai kaidah ejaan Kamus Besar Bahasa Indonesia (KBBI). Penelitian ini menggunakan metode kualitatif, yang menitikberatkan pada proses perancangan, implementasi, dan analisis sistem. Evaluasi sistem dilakukan secara kuantitatif menggunakan dataset karya ilmiah yang telah dianotasi secara manual. Kinerja sistem diukur melalui *precision, recall,* dan *F1-score*, guna menilai akurasi, kelengkapan, serta konsistensi deteksi kesalahan ejaan. Dataset uji bersifat independen dan tidak digunakan dalam proses pengembangan, sehingga validitas evaluasi dapat terjaga.

2.1. Pengembangan Sistem

Penelitian ini menggunakan metode pengembangan *Waterfall*, yang merupakan model klasik pengembangan perangkat lunak dengan pendekatan linier dan sistematis. Setiap tahap dilaksanakan secara berurutan, dimulai dari analisis kebutuhan, perancangan, implementasi, pengujian, hingga penyebaran dan pemeliharaan sistem [17]. Model ini dipilih untuk memastikan proses perancangan berjalan efektif dan terstruktur.



Gambar 1 Alur metode *waterfall*

a. Analisis kebutuhan (*Requirement*)

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15219

Pada tahap ini dilakukan kajian literatur dan pengumpulan data. Kajian literatur mencakup konsep *Natural Language Processing* (NLP), metode deteksi kesalahan penulisan, serta kaidah ejaan Bahasa indonesia berdasarkan KBBI. Sementara itu, pengumpulan data melibatkan pemanfaatan database KBBI yang diperoleh dari repositori GitHub, serta penyusunan dataset mandiri berupa kalimat berbahasa Indonesia yang dianotasi untuk keperluan pengujian sistem.

b. Perancangan Sistem (*Design*)

Pada tahap perancangan menggunakan metode *unified modeling language* (UML), uml yang digunakan antara lain *usecase diagram, class diagram, sequence diagram* dan *activity diagram*. *Usecase* digunakan untuk menggambarkan hubungan interaksi pengguna dan sistem. *Class diagram* digunakan untuk memodelkan kelas di dalam sistem. *Sequence diagram* menggambarkan interaksi antar objek dalam sistem. Serta *activity diagram* digunakan untuk memodelkan alur kerja dalam sistem.

c. Implementasi (*Implementation*)

Sistem ini dibangun menggunakan bahasa pemrograman Python untuk sisi *backend*, HTML dan CSS untuk frontend guna menghasilkan antarmuka yang responsif. MySQL digunakan sebagai basis data utama, dengan integrasi database KBBI yang diambil dari repositori GitHub sebagai sumber acuan ejaan Bahasa Indonesia. Serta menggunakan library NLP antara lain *lowercase*, spaCy, Sastrawi dan NLTK.

d. Pengujian (*Verification*)

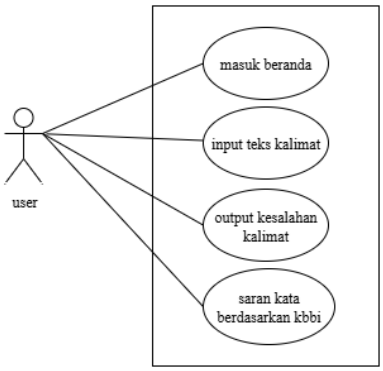
pada tahap pengujian sistem menggunakan Metode *white box testing* dan *black box testing*. *White box* digunakan untuk menguji logika dan struktur kode program dengan memastikan alur kontrol, kondisi serta proses berjalan tanpa kesalahan. *Black box* digunakan untuk menguji fungsionalitas sistem dengan memastikan semua fitur berjalan sesuai kebutuhan.

e. Pemeliharaan (*Maintenance*)

Tahap terakhir adalah tahap pemeliharaan berfungsi untuk memastikan sistem berjalan dengan optimal. Aktivitas pemeliharaan meliputi perbaikan kesalahan (*bug fixing*), penambahan fitur sesuai kebutuhan pengguna, serta pembaruan data KBBI apabila terdapat perubahan standar ejaan.

2.2. Rancangan Sistem

a. Use case

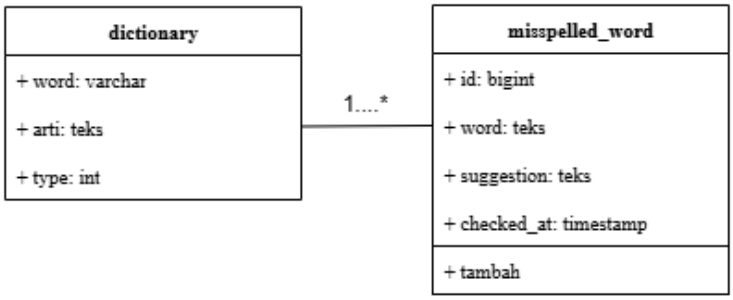


Gambar 2 *usecase diagram*

Berdasarkan gambar *usecase* di atas, dapat dilihat bahwa sistem hanya memiliki satu aktor saja yaitu aktor pengguna. Aktor pengguna hanya memiliki akses untuk menggunakan serta memanfaatkan fitur sistem dalam mendeteksi kesalahan ejaan.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15219

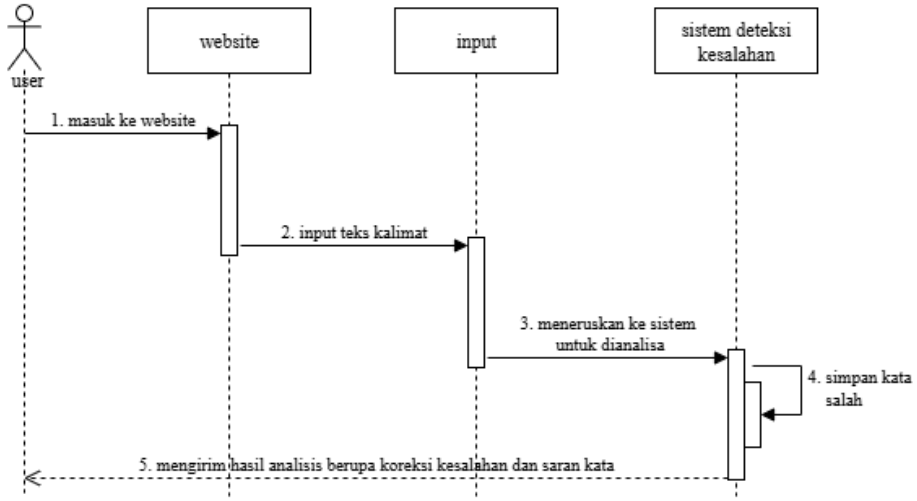
b. Class diagram



Gambar 3 *class diagram*

Class diagram di atas menggambarkan struktur dua kelas dalam sistem deteksi kesalahan ejaan, yaitu *dictionary* dan *misspelled_word*. Kelas *dictionary* berfungsi sebagai kamus utama yang menyimpan daftar kata baku. kelas *misspelled_word* digunakan untuk menyimpan data kata-kata salah ejaan yang terdeteksi oleh sistem. Relasi antara kedua kelas ini bersifat *one-to-many*, artinya satu kata dalam *dictionary* dapat memiliki banyak kemunculan sebagai saran koreksi untuk berbagai kata yang salah dalam *misspelled_word*.

c. Sequence diagram

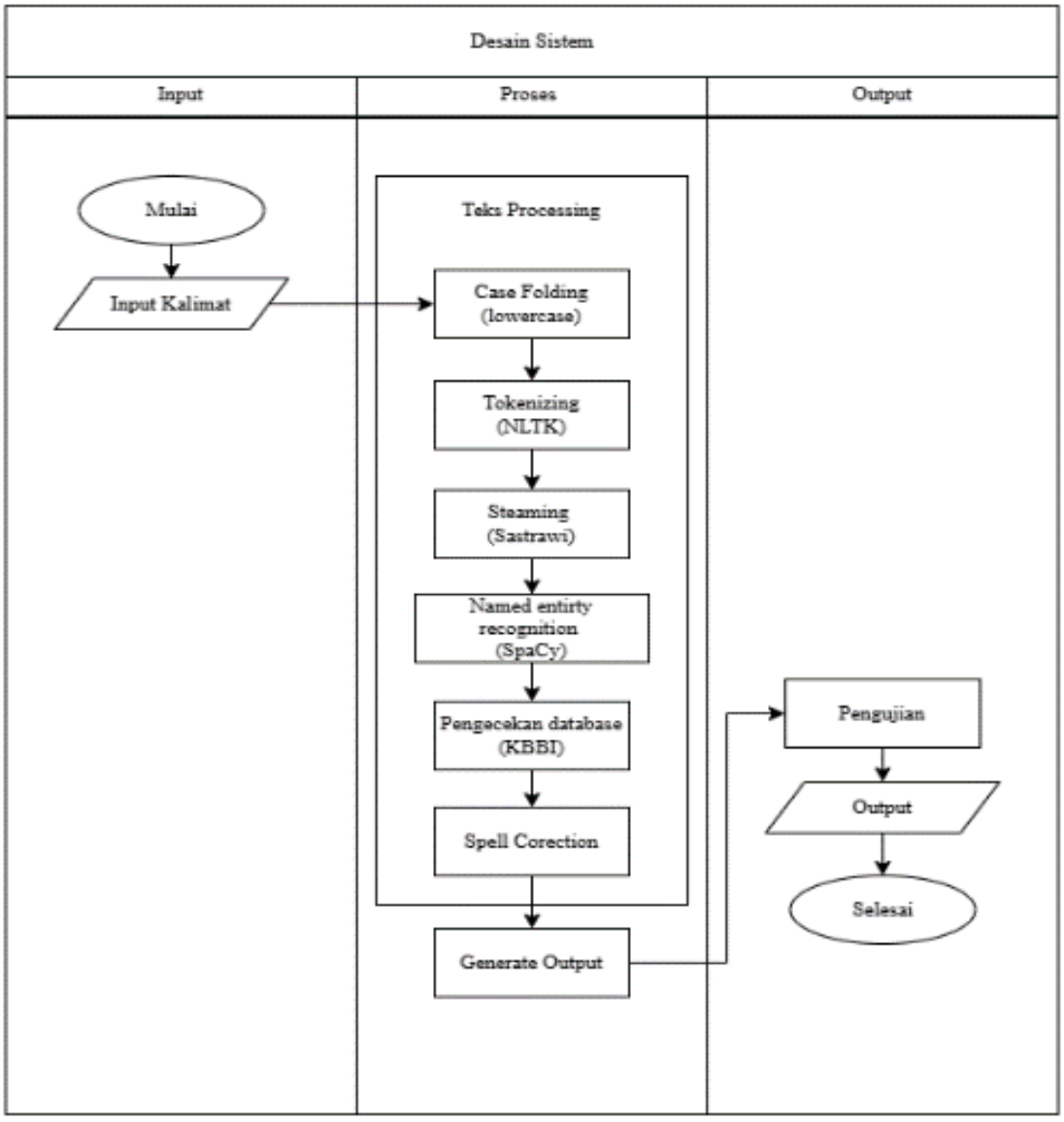


Gambar 4 *sequence diagram*

Gambar *sequence diagram* di atas menunjukkan alur interaksi antara user dan komponen-komponen sistem. Pada langkah pertama, pengguna mengakses halaman website. Langkah kedua, pengguna memasukkan teks ke dalam kolom input yang disediakan. Pada langkah ketiga, sistem meneruskan input tersebut ke modul deteksi kesalahan untuk dianalisis. Langkah keempat, kata yang salah akan di simpan di database khusus. Setelah proses selesai, langkah terakhir sistem mengirimkan hasil berupa koreksi kesalahan dan saran kata untuk ditampilkan kepada pengguna.

d. Activity diagram

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15219



Gambar 5 *activity diagram*

Penerapan metode NLP diatas secara umum menggunakan *rulebased* model. Tetapi juga menggunakan basis model lain seperti *lexicon based* sebagai sumber data berupa file KBBI. KBBI disini berfungsi sebagai daftar kata baku (*word list*) serta sumber validasi kata. Input kalimat Proses diawali dengan pengguna memasukkan kalimat atau teks yang ingin diperiksa. Teks yang dimasukkan dapat berupa kalimat biasa, paragraf, atau potongan artikel yang akan dideteksi kesalahan ejaannya. Selanjutnya akan dilakukan berbagai macam *teks processing*.

Tahap awal dalam pemprosesan teks adalah *case Folding,* pada proses ini Sistem melakukan konversi semua huruf dalam teks menjadi huruf kecil (*lowercase*). Hal ini bertujuan untuk Standarisasi data agar proses berikutnya tidak terganggu oleh perbedaan huruf kapital dan kecil. Dengan mengubah seluruh huruf menjadi *lowercase*, variasi penulisan seperti “Data”, “DATA”, dan “data” akan dianggap sama, sehingga dapat mengurangi pengulangan proses dan meningkatkan efisiensi dalam analisis teks. Menurut Studi oleh Fournet et al tahun 2022 menunjukkan bahwa penggunaan *lowercase* memberikan tingkat akurasi identifikasi kata yang lebih tinggi dibandingkan *uppercase* dalam berbagai tugas pemrosesan kata dan kalimat, terutama karena huruf kecil lebih familiar bagi pembaca dan lebih mendukung pengenalan kata secara cepat [18]. Sedangkan format lain seperti *title case* dan *sentence case* tidak digunakan dalam *case*

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15219

folding karena keduanya masih menggabungkan huruf kapital dengan huruf kecil. Hal ini menyebabkan potensi perbedaan token yang tidak relevan secara semantik, yang justru dapat menurunkan performa sistem.

Tabel 1 Hasil proses *lowercase*

Input	Output
Indonesia Adalah Negara Kepulauan	indonesia adalah negara kepulauan
JAKARTA IBUKOTA INDONESIA	jakarta ibukota indonesia
pRESIDEN pRABOWO pERGI kE bELANDA	presiden prabowo pergi ke belanda

Setelah standarisasi huruf selesai, proses dilanjutkan dengan pemecahan kata (tokenisasi). Proses dilakukan menggunakan *library Natural Language Toolkit* (NLTK) yang sudah banyak digunakan untuk pemrosesan bahasa alami. Tokenisasi sering menjadi metode pilihan dibanding teknik lain seperti *character-level* atau *subword-level tokenization*. Tokenisasi kata sederhana sangat cocok untuk Bahasa Indonesia yang cukup terpisah oleh spasi, sedangkan metode sub-kata seperti BPE, *WordPiece,* dan *SentencePiece* lebih dipakai dalam model *transformer* khususnya untuk menangani kata jarang atau bahasa tanpa spasi [19].

Gambar 6 Cara kerja tokensasi

Selanjutnya, setiap token yang terbentuk dikembalikan ke bentuk dasarnya melalui proses stemming, dengan menggunakan *library Sastrawi*, yang khusus dikembangkan untuk Bahasa Indonesia. Stemming berguna untuk menghilangkan imbuhan seperti awalan, akhiran, atau sisipan. Studi oleh Mustikasari et al tahun 2021 menunjukkan bahwa algoritma *stemming Sastrawi* mampu mengembalikan 95,2% kata berimbuhan ke *root word* yang benar, lebih unggul daripada *Rule-Based Stemming* (algoritma Nazief-Adriani dan algoritma Arifin-Setiono) [20].

Tabel 2 Kata diubah ke bentuk dasar

Input	Output
makanan	makan
berlari	lari
tertulis	tulis

Setelah itu untuk mencegah kesalahan koreksi pada kata khusus dilakukan proses *Named Entity Recognition* (NER). Pada proses ini sistem mendeteksi entitas penting dalam teks, seperti nama orang, nama tempat, atau organisasi menggunakan *SpaCy*. *Library spacy* menyediakan modul *named entity recognition* (NER) yang sangat efisien dan berbasis *neural network* [21]. Entitas yang terdeteksi biasanya dikecualikan dari proses koreksi ejaan karena merupakan kata khusus. Pemilihan *SpaCy* didasari oleh fakta bahwa ia menawarkan model NER pra-latih yang cepat dan akurat, serta mendukung pipeline multibahasa dan penggunaan dalam produksi secara mulus. Studi perbandingan kinerja NER oleh Anna Kiefer n.d menunjukkan bahwa *SpaCy* memiliki waktu eksekusi sangat rendah (~0,004 ms per kalimat) dengan *F1-score* mencapai ~0,967, meskipun *Flair* memiliki akurasi lebih tinggi sedikit—tetapi *Flair* secara signifikan lebih lambat (sekitar 75–90 kali lebih lama) [22]. Dalam evaluasi lain terhadap berbagai alat NER (termasuk *Stanford* NER, NLTK, *Polyglot, GATE, Flair, DeepPavlov*), *SpaCy* terbukti sebagai salah satu alat paling cepat, sementara *Stanford* dan *Flair* berjalan lebih lambat hingga dua urutan magnitudo [23].

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15219

Tabel 3 Hasil setelah proses *spacy*

Input	Output
presiden prabowo pergi ke belanda selama beberapa hari	"prabowo" (*person*)
	"belanda" (*location*)

Kata yang bukan entitas kemudian diperiksa dalam basis data KBBI. KBBI berisi kumpulan kata-kata baku Bahasa Indonesia. Jika kata ditemukan dalam KBBI, maka kata dianggap benar/baku. Jika kata tidak ditemukan dalam KBBI, kata dianggap salah ejaan atau tidak baku, sehingga akan masuk ke proses koreksi ejaan (*Spell Correction*). Dataset KBBI didapat dari platform github, diunggah pada september 2024 oleh pengguna dyazincahya dengan total data sebanyak 115.978 Kata [24]. Sumber data diambil dari KBBI VI daring yang diterbitkan oleh Badan Pengembangan dan Pembinaan Bahasa, yang merupakan bagian dari Kementerian Pendidikan Dasar dan Menengah Republik Indonesia [25].

Jika kata tidak terdaftar dalam KBBI, sistem akan menerapkan proses koreksi ejaan menggunakan algoritma *Levenshtein Distance* atau *library SpellChecker*. *Levenshtein Distance* adalah algoritma untuk menghitung jumlah minimum operasi yang diperlukan untuk mengubah satu kata menjadi kata lain, dengan operasi Penyisipan karakter, Penghapusan karakter dan Substitusi karakter. Semakin kecil nilai *Levenshtein Distance*, semakin mirip dua kata tersebut. Studi yang dilakukan oleh Yanfi et al tahun 2023 yang membandingkan *Levenshtein* dengan *Jaro–Winkler* dan metode *n-gram* mencatat bahwa untuk kata dengan panjang ≥4 karakter, *Levenshtein* mencapai akurasi hampir setara dengan *Jaro–Winkler* (~99,5 %) [26].

$D(s,t) = min\ D(s-1,t) + 1$ (penghapusan) (1)

$D(s,t) = min\ D(s,t-1) + 1$ (penyisipan) (2)

$D(s,t) = min\ D(s-1,t-1) + 1, sj \neq ti$ (penggantian) (3)

$D(s,t) = min\ D(s-1,t-1), sj = ti$ (tidak ada perubahan) (4)

keterangan:

$s = String$ sumber $s(j) =$ karakter string sumber ke-j

$t = String$ target $t(i) =$ karakter string target ke-i

$D =$ jarak edit *levenshtein distance*

Saat akan mengubah kata salah "Indonsia" menjadi kata benar "Indonesia" di dapatkan proses sebagai berikut:

Tabel 4 Cara Kerja *Levenshtein distance*

	"	I	N	D	O	N	E	S	I	A
"	0	1	2	3	4	5	6	7	8	9
I	1	0	1	2	3	4	5	6	7	8
N	2	1	0	1	2	3	4	5	6	7
D	3	2	1	0	1	2	3	4	5	6
O	4	3	2	1	0	1	2	3	4	5
N	5	4	3	2	1	0	1	2	3	4
S	6	5	4	3	2	1	1	1	2	3
I	7	6	5	4	3	2	2	1	1	2
A	8	7	6	5	4	3	3	2	1	1

Baris pertama dan kolom pertama diisi angka 0, 1, 2, 3, dst. ini mewakili skenario jika semua karakter dihapus atau disisipkan. Langkah perhitungan setiap sel dihitung dengan, jika huruf sama maka ikuti nilai diagonal kiri-atas (D(s-1, t-1)).Jika huruf berbeda maka ambil nilai minimum dari Atas (D(s-1, t) + 1) berupa penghapusan huruf, Kiri (D(s, t-1) + 1) berupa penyisipan huruf dan Diagonal kiri-atas (D(s-1, t-1) + 1) berupa penggantian huruf. Kesimpulannya Nilai akhir di pojok kanan bawah = 1. Artinya, perlu 1 perubahan (penyisipan huruf 'E') untuk mengubah "Indonsia" menjadi "Indonesia".

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15219

SpellChecker adalah *library Python* yang mempermudah proses koreksi ejaan dengan basis data kata baku. *SpellChecker* secara otomatis menghitung jarak *Levenshtein* dan memberikan saran kata terdekat. Menurut studi oleh Elmobark tahun 2025 yang membandingkan *SpellChecker* dengan *Hunspell, JamSpell,* dan *Autocorrect*. *SpellChecker* menunjukkan akurasi ~97,1% untuk kata umum, hampir setara dengan *Hunspell* namun lebih mudah diintegrasikan [27]. Setelah semua Langkah selesai akan dilakukan *Generate Output,* sistem akan menghasilkan output berupa teks yang sudah diperiksa dan dikoreksi ejaannya. Teks ini dapat ditampilkan kepada pengguna atau disimpan dalam sistem. Sebelum teks ditampilkan kata yang salah akan terlebih dahulu disimpan ke database *misspelled_word*.

Terpisah dari proses utama, tahap selanjutnya yaitu evaluasi model dilakukan untuk mengukur kinerja sistem dalam mendeteksi kesalahan penulisan pada teks karya ilmiah. Evaluasi ini menggunakan *Confusion Matrix*, yaitu sebuah tabel yang berfungsi untuk menilai performa model klasifikasi dengan cara membandingkan hasil prediksi sistem terhadap data uji yang telah diberi label secara manual [28].

		Nilai Aktual	
		Positive	Negative
Nilai Prediksi	Positive	TP (True Positive)	FP (False Positive)
	Negative	FN (False Negative)	TN (True Negative)

Gambar 7 *confusion matrix*

Pertama, metrik akurasi digunakan untuk mengukur proporsi prediksi yang benar (baik positif maupun negatif) terhadap seluruh data. Akurasi dihitung menggunakan persamaan berikut:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (5)$$

Selanjutnya, metrik presisi digunakan untuk mengetahui tingkat ketepatan model dalam memprediksi kata salah yang benar-benar salah. Presisi dirumuskan sebagai:

$$precision = \frac{TP}{TP+FP} \quad (6)$$

Kemudian, metrik *recall* (sensitivitas) diperlukan untuk mengetahui seberapa baik model mampu menemukan kembali seluruh kasus positif yang sebenarnya ada. Rumusnya ditunjukkan pada persamaan berikut:

$$Recall = \frac{TP}{TP+FN} \quad (7)$$

Terakhir, *F1-Score* merupakan rata-rata harmonis antara presisi dan *recall* sehingga memberikan penilaian yang lebih seimbang, terutama pada kondisi ketidakseimbangan kelas [21]. Rumusnya:

$$F1-Score = 2\,x\,\frac{Precision\ x\ Recal}{Precision+recall} \quad (8)$$

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15219

3. HASIL DAN PEMBAHASAN

3.1. Evaluasi Model

Pada tahap evaluasi, digunakan kalimat yang diambil dari berbagai karya ilmiah dengan beragam topik. Dataset yang digunakan terdiri atas 300 kalimat dengan jumlah kata kurang lebih 3.000 kata. Komposisi dataset dibagi secara seimbang, yaitu 50% kalimat benar dan 50% kalimat yang mengandung kesalahan. Kalimat yang dikategorikan mengandung kesalahan meliputi penggunaan kata singkatan yang tidak baku, kesalahan penulisan kata, serta penggunaan istilah asing yang tidak tercantum dalam KBBI. Dataset yang digunakan merupakan data uji murni dan tidak dilibatkan dalam proses pengembangan maupun pelatihan sistem. Pengujian dilakukan dengan menggunakan kode perhitungan sederhana sehingga diperoleh struktur dasar confusion matrix sebagai berikut:

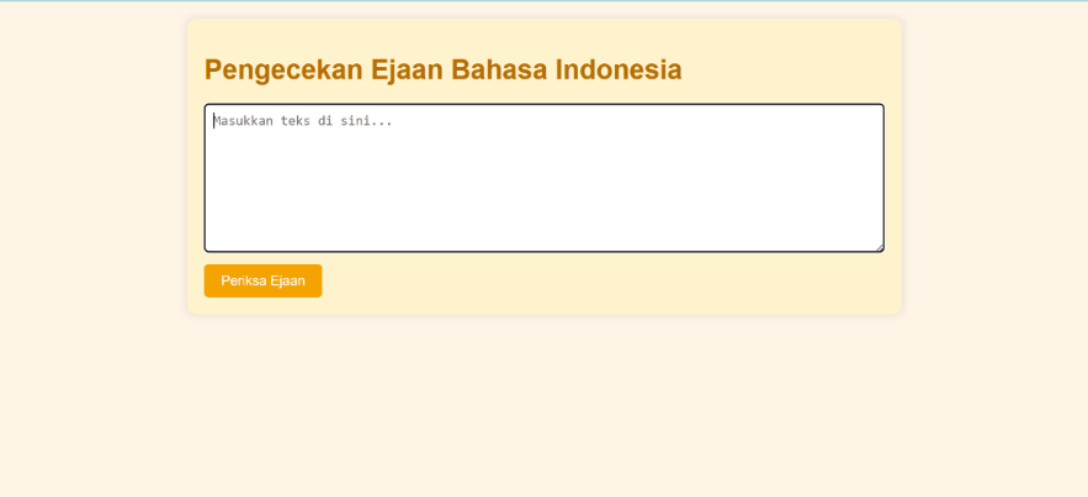
		Nilai Aktual	
		Positif	Negatif
Nilai Presdiksi	Positif	TP = 140	FP = 13
	Negatif	FN = 10	TN = 137

Gambar 8 hasil pengujian dataset

Dari gambar diatas, dilakukan perhitungan matrik evaluasi. Berdasarkan perhitungan tersebut didapatkan hasil perhitungan matrix evaluasi dengan nilai akurasi sebesar 92%, nilai presisi sebesar 92%, nilai *recall* sebesar 93% dan nilai *f1-score* sebesar 92%.

3.2. Implementasi

Pengimplementasian sistem pendeteksi kesalahan penulisan telah berhasil dilakukan. Antarmuka pengguna dirancang sederhana dan intuitif untuk memudahkan interaksi pengguna. Gambar dibawah memperlihatkan tampilan awal aplikasi yang menyajikan kolom input teks dan tombol “Periksa Ejaan”. Pengguna dapat langsung memasukkan teks karya ilmiah ke dalam kolom tersebut untuk dilakukan pengecekan kesalahan.



Gambar 9 Tampilan awal sistem

Setelah pengguna menekan tombol "Periksa Ejaan", sistem akan memproses input menggunakan metode *Natural Language Processing* (NLP). Sistem akan melakukan tokenisasi,

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15219

mencocokkan setiap kata dengan basis data KBBI, dan mengidentifikasi kata-kata yang tidak sesuai. Kata yang tidak sesuai akan disimpan dalam database yang berbeda guna penelitian selanjutnya. Gambar dibawah menunjukkan tampilan hasil deteksi ejaan, yang memuat revisi kalimat dan daftar kata tidak baku beserta saran dari database KBBI.

Gambar 10 Tampilan hasil deteksi

Dari hasil implementasi, sistem mampu memberikan umpan balik berupa koreksi teks serta daftar kata salah. Fitur ini mendukung tujuan utama aplikasi, yaitu membantu pengguna dalam menyesuaikan ejaan teks karya ilmiah agar sesuai dengan standar Bahasa Indonesia yang berlaku.

3.3. Pengujian

a. White box testing

White box testing merupakan pengujian perangkat lunak yang berfokus pada desain dan kode program. Pengujian ini bertujuan untuk memastikan semua kode berjalan dengan benar tanpa adanya kesalahan [29]. Berikut ini adalah hasil dari pengujian *white box* yang telah dilakukan pada sistem tersebut:

Tabel 5 Hasil Pengujian *White Box*

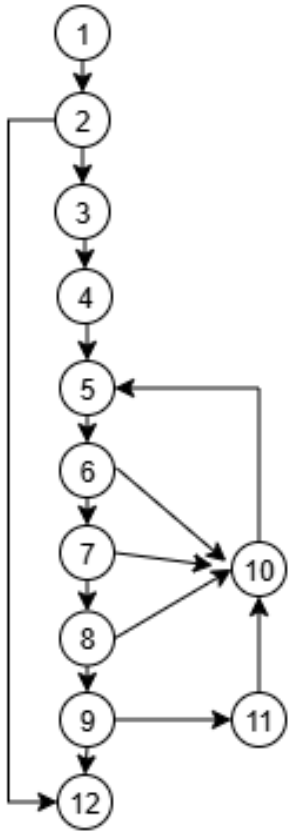
No	Kode Program	Tujuan / Hasil yang Diharapkan	Hasil
1.	text = data.get('text')	Input teks dari user melalui request JSON	Sesuai
2.	kbbi_words = load_kbbi()	Memuat database kbbi	Sesuai
3.	text.lower()	mengubah semua huruf menjadi kecil	Sesuai
4.	word_tokenize(text.lower())	Melakukan Tokenisasi dengan memecah teks menjadi kata	Sesuai

5.	for word in words:
	if not word.isalpha():
	corrected_text.append(word)
	continue
		Melakukan iterasi pada setiap kata	Sesuai

6.	stemmer.stem(word)	Melakukan Stemming dengan mengembalikan kata ke bentuk dasar	Sesuai

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15219

7.	doc = nlp(text) ent.label_ in ['GPE','LOC']	Melakukan pengenalan lokasi/entitas	Sesuai
8.	if word in kbbi_words:	Pengecekan kata terhadap database KBBI	Sesuai
9.	spell.correction(word)	Spell correction untuk memperbaiki ejaan salah	Sesuai
10.	corrected_text = [ ]	Menampung hasil teks yang sudah dikoreksi	Sesuai
11.	save_misspelled_word(word, suggestion)	Menyimpan kata salah ke DB jika belum ada	Sesuai
12.	return jsonify(result)	Generate output JSON berisi teks terkoreksi dan daftar error	Sesuai



Gambar 11 *Flowgraph whitebox*

Berdasarkan gambar 11, akan dilakukan perhitungan kompleksitas siklomatis (*Cyclomatic Complexity*) untuk menentukan jumlah jalur independen. Jumlah node yang teridentifikasi sebanyak 12 sedangkan jumlah edge ada sebanyak 16. Dengan menggunakan rumus berikut:

$$V(G) = E - N + 2 \quad (9)$$

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15219

Dari hasil perhitungan diperoleh enam jalur independen yang harus diuji. Jalur pertama merupakan alur ketika KBBI gagal memuat data, sehingga proses tidak dapat dilanjutkan. Jalur kedua adalah alur ketika kata melewati proses stemming, namun tidak memenuhi kondisi tertentu, lalu masuk ke *corrected_text* dan kembali ke loop. Jalur ketiga merupakan kelanjutan dari jalur kedua, yaitu kata melewati proses NER (*Named Entity Recognition*) tetapi tidak memenuhi kondisi, kemudian dimasukkan ke *corrected_text* dan kembali ke loop. Jalur keempat adalah kelanjutan dari jalur ketiga, ketika kata melewati pengecekan KBBI, namun tidak memenuhi kondisi, sehingga dimasukkan ke *corrected_text* dan kembali ke loop. Jalur kelima merupakan lanjutan dari jalur keempat, yaitu kata dilakukan *spell correction*. Jika ditemukan kesalahan ejaan, kata tersebut akan disimpan, lalu dimasukkan ke *corrected_text* dan kembali ke loop. Jalur keenam adalah kelanjutan dari jalur kelima, ketika sistem memberikan saran kata dan perbaikan ejaan, yang kemudian akan ditampilkan melalui output.

b. Black box testing

Pengujian *Black Box* adalah pengujian yang memverifikasi hasil eksekusi aplikasi berdasarkan masukan yang diberikan untuk memastikan fungsional dari aplikasi sudah sesuai dengan kebutuhan. Pengujian ini berfokus pada tampilan dan pengujian fungsional yang terdapat pada aplikasi, serta kesesuaian pada alur fungsi yang dibutuhkan oleh user [30]. Berikut ini adalah hasil dari pengujian *Black Box* yang telah dilakukan pada sistem tersebut:

Tabel 6 Hasil Pengujian *Black Box*

Komponen yang Diuji	Skenario pengujian	Output yang diharapkan	Hasil	Status
Input teks	Pengguna memasukkan teks karya ilmiah	Sistem menerima teks input	Teks diterima sistem	Berhasil
Tombol periksa	Pengguna menkan “tombol periksa”	Sistem mulai memproses teks	Teks mulai diproses oleh sistem	Berhasil
Hasil kata	Memberikan teks yang berisi kata benar dan salah	Memberikan tanda pada kata yang salah	Kata salah ditandai	Berhasil
Saran kata	Memberikan teks dengan kata salah	Menyarankan kata sesuai dengan kata terdekat yang terdapat di database	Saran kata ditampikan	Berhasil
Simpan kata salah	Memberikan kata salah	Kata yang salah otomatis tersimpan di dalam database khusus	Kata yang salah tersimpan	Berhasil

3.4. Pembahasan

Implementasi sistem pendeteksi kesalaan penulisan karya ilmiah dengan metode nlp pada ejaan kbbi telah berhasil dilakukan. Pada evaluasi model yang dilakukan menggunakan empat metrik utama yaitu akurasi, presisi, *recall*, dan *F1-score*. Berdasarkan evaluasi tersebut didapatkan hasil akurasi sebesar 92%, presisi 92%, *recall* 93% dan *f1-score* sebesar 92%. Akurasi sebesar 92% menunjukkan bahwa dari total 300 data uji sistem berhasil mengklasifikasikan data secara benar. Presisi 92% berarti bahwa dari semua kata yang terdeteksi sebagai kesalahan ejaan oleh sistem, sebanyak 92% memang benar-benar salah. *Recall* 93% mengindikasikan bahwa dari semua kata yang memang merupakan kesalahan ejaan dalam dataset, sebanyak 93% berhasil dikenali oleh sistem. *F1-score* 92% menunjukkan bahwa sistem memiliki keseimbangan yang baik antara presisi dan recall, mengindikasikan konsistensi dalam sistem.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15219

Tabel 7 Perbandingan Hasil Sistem

No	Judul Penelitian	Metode	Metriks yang Dilaporkan	Hasil Performa
1.	Sistem deteksi kesalahan penulisan karya ilmiah dengan metode natural language processing (NLP) pada ejaan KBBI	Nlp, levenshtein dan KBBI	akurasi, presisi, recall, dan f1-score	accuracy 92%, precision 92%, recall 93%, f1-score 92%
2.	Sistem pengecekan ejaan menggunakan pendekatan NLP umum dan database KBBI	NLP dan KBBI (dictionary-based)	akurasi	81,64%
3.	Deteksi kata tak baku dan kesalahan penulisan kata pada tugas akhir mahasiswa menggunakan metode dictionary lookup	dictionary lookup	presisi dan error rate	presisi 28,71%, error 2,76%
4.	Koreksi kesalahan ejaan pada hasil konversi file dalam ekstraksi informasi	soundex dan damerau-levenshtein	recall, presisi dan koreksi	recall 100%, presisi 59,5%, akurasi 71,4%
5.	LSTM-based NLP approach for spelling error detection and correction in scientific writing indonesian language	lstm (deep learning)	akurasi	93%
6.	Automatic spelling error detection and correction for tigrigna information retrieval: a hybrid approach	Jaccard bigram, edit distance dan probabilitas kata	f-measure dan akurasi	f1 deteksi 98,85%, akurasi koreksi 95,36%

Berdasarkan Tabel 7, sistem yang dikembangkan dalam penelitian ini menunjukkan performa yang kompetitif dibandingkan beberapa penelitian terdahulu, khususnya pada nilai akurasi, presisi, dan F1-score. Meskipun beberapa penelitian berbasis *deep learning* dan hybrid model pada bahasa lain menunjukkan performa yang lebih tinggi, pendekatan NLP berbasis *rule-based* dan *lexicon based* yang digunakan dalam penelitian ini tetap mampu menghasilkan kinerja yang baik dengan akurasi mencapai 92%.

Pencapaian performa tersebut dipengaruhi oleh beberapa faktor. Pertama, penggunaan database KBBI yang terbarukan memungkinkan sistem memiliki cakupan kosakata yang lebih lengkap. Kedua, pemanfaatan spaCy untuk *Named Entity Recognition* (NER) membantu sistem dalam mengenali kata-kata yang merupakan nama tempat, nama orang, atau entitas tertentu sehingga tidak salah terdeteksi sebagai kesalahan ejaan. Ketiga, sistem dilengkapi dengan logika tambahan untuk menangani kasus penggandaan huruf di akhir kata, yang umum terjadi dalam kesalahan penulisan. Keempat, penerapan algoritma *Levenshtein Distance* membantu sistem dalam memberikan saran perbaikan ejaan berdasarkan tingkat kemiripan karakter antar kata.

Meski demikian, penelitian ini memiliki beberapa keterbatasan. Salah satunya adalah model belum mampu mengenali konteks kalimat secara menyeluruh, sehingga terkadang masih terjadi *false positive*, yaitu ketika kata yang sebenarnya benar dianggap salah. Selain itu, jumlah kata yang diproses oleh sistem mempengaruhi waktu kinerja sistem, semakin banyak kata maka proses sistem akan semakin lama juga. Sistem ini juga sangat bergantung pada KBBI yang statis, sehingga bila ada perubahan atau penambahan kata perlu dilakukan pembaharuan secara manual agar tetap relevan.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15219

4. KESIMPULAN

Sistem pendeteksi kesalahan penulisan karya ilmiah berbasis NLP dengan acuan ejaan KBBI telah berhasil diimplementasikan dengan baik dan menunjukkan performa yang memadai. Untuk pengembangan lebih lanjut terdapat beberapa langkah yang dapat dilakukan seperti mengintegrasi model berbasis *deep learning* seperti *transformer* agar sistem mampu memahami konteks kalimat secara lebih mendalam. Optimasi kinerja sistem, khususnya dalam efisiensi pemrosesan data skala besar sehingga waktu respons tetap cepat dan stabil. Pembaruan database KBBI secara dinamis, sehingga sistem dapat otomatis menyesuaikan dengan perkembangan kosakata Bahasa Indonesia. Pemanfaatan data kata salah yang tersimpan untuk dijadikan referensi pembelajaran maupun pengembangan sistem di masa mendatang. Penambahan fitur interaktif, misalnya kemampuan menambahkan kata baku melalui antarmuka web atau unggahan file, agar sistem lebih fleksibel dan mudah digunakan.

DAFTAR PUSTAKA

[1] *T. Handayani et al., "Indikator Iptek, Riset, dan Inovasi Indonesia 2024". Jakarta: Penerbit BRIN, 2024.* Jakarta, 2024.

[2] D. Khoyriyah, A. Rosari Siregar, M. Wati Siregar, and F. Bahasa Dan Seni, "Analisis Kesalahan Berbahasa dalam Penulisan Karya Ilmiah: Skripsi Mahasiswa-i Universitas Negeri Medan."

[3] G. H. Tarigan *et al.*, "Analisis Kesalahan Ejaan Bahasa Indonesia Pada Artikel Ilmiah Mahasiswa," Medan, May 2024.

[4] J. Eisenstein, *"Natural Language Processing". Cambridge, MA: The MIT Press,*. 2019.

[5] A. Michael Bala Ledjap, F. Putri Rochmawati, D. Ayu Eka Marsanda, A. Puspita Sari, and U. Pembangunan Nasional Veteran Jawa Timur, "Pemanfaatan Natural Language Processing Untuk Pengecekan Ejaan Sesuai KBBI."

[6] A. Misbullah *et al.*, "DETEKSI KATA TAK BAKU DAN KESALAHAN PENULISAN KATA PADA TUGAS AKHIR MAHASISWA MENGGUNAKAN METODE DICTIONARY LOOKUP," *Jurnal Pendidikan Teknologi informasi*, vol. 6, no. 2, pp. 130–141, 2022.

[7] A. Rohim and K. K. Purnamasari, "KOREKSI KESALAHAN EJAAN PADA HASIL KONVERSI FILE DALAM EKSTRAKSI INFORMASI."

[8] A. Wildan, D. Adam, and T. Wahyuni, "Arus Jurnal Sains dan Teknologi (AJST) Pengaplikasian Spell Checker pada Aplikasi Kamus Bahasa Bugis dengan Metode Levenshtein INFO PENULIS," vol. 2, no. 2, 2024, [Online]. Available: http://jurnal.ardenjaya.com/index.php/ajsthttp://jurnal.ardenjaya.com/index.php/ajst

[9] F. Risang Agung Samudero, J. Sasongko Wibowo, and J. Tri Lomba Juang Semarang, "Aplikasi Spelling Correcting Pada Penulisan Bahasa Indonesia Dengan Metode Jaro Winkler," vol. 17, no. 1, pp. 65–73, 2024, doi: 10.51903/elkom.v17i1.1445.

[10] T. Hartina, "Pendeteksi Kesalahan Pengetikan Kata Non Baku pada Karya Tulis Menggunakan Metode N-Gram," *JURNAL INFORMATIKA*, vol. 7, no. 1, 2020, [Online]. Available: http://ejournal.bsi.ac.id/ejurnal/index.php/ji

[11] N. C. Dewi and A. Qoiriah, "Implementasi Algoritma Jaro-Winkler Distance dan N-gram untuk Deteksi dan Prediksi Perbaikan Kesalahan Penulisan Kata Bahasa Indonesia pada Karya Tulis Ilmiah Mahasiswa," *Journal of Informatics and Computer Science*, vol. 02, 2021.

[12] Y. D. P. Halim and I. Nurhaida, "LSTM-Based NLP Approach for Spelling Error Detection and Correction in Scientific Writing Indonesian Language," *Electronic Journal of Education, Social Economics and Technology*, vol. 5, no. 1, pp. 30–39, Apr. 2024, doi: 10.33122/ejeset.v5i1.309.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15219

[13] A. Saputra, D. Sakethi, Y. Tri Utami, J. Ilmu Komputer FMIPA Universitas Lampung Jalan Sumantri Brojonegoro No, and B. Lampung, "SISTEM PENDETEKSI PENULISAN KATA PADA DOKUMEN BERBASIS WEB," 2021.

[14] "fahrielfazza,+9.+Rancang+Bangun+Aplikasi+Deteksi+Kesalahan+Penulisan+(53+-+58)".

[15] "FULL-TEXT INDEXING BERBASIS ANDROID."

[16] S. G. Desta and G. S. Lehal, "Automatic spelling error detection and correction for Tigrigna information retrieval: a hybrid approach," *Bulletin of Electrical Engineering and Informatics*, vol. 12, no. 1, pp. 387–394, Feb. 2023, doi: 10.11591/eei.v12i1.4209.

[17] Roger S. Pressman and Bruce R Maxim, *Software Engineering A Practitioner's Approach*, 9th ed. New York, 2020.

[18] C. Fournet, J. Mirault, M. Perea, and J. Grainger, "Effects of Letter Case on Processing Sequences of Written Words," *J Exp Psychol Learn Mem Cogn*, vol. 48, no. 12, pp. 1995–2003, Aug. 2022, doi: 10.1037/xlm0001179.

[19] S. J. Mielke *et al.*, "Between words and characters: A Brief History of Open-Vocabulary Modeling and Tokenization in NLP," Dec. 2021, [Online]. Available: http://arxiv.org/abs/2112.10508

[20] D. Mustikasari, I. Widaningrum, R. Arifin, W. Henggal, and E. Putri, "Comparison of Effectiveness of Stemming Algorithms in Indonesian Documents," 2021. [Online]. Available: http://tiny.cc/rootwords.

[21] R. M. Yanti, I. Santoso, and L. H. Suadaa, "Application of Named Entity Recognition via Twitter on SpaCy in Indonesian (Case Study: Power Failure in the Special Region of Yogyakarta)," 2021.

[22] "Comparing Apples to Apples. Testing Named Entity Recognition… | by Anna Kiefer | Medium." Accessed: Jul. 26, 2025. [Online]. Available: https://medium.com/%40askiefer/apples-to-apples-d0345d05b87b

[23] S. Vychegzhanin and E. Kotelnikov, "Comparison of named entity recognition tools applied to news articles," in *Proceedings - 2019 Ivannikov Ispras Open Conference, ISPRAS 2019*, Institute of Electrical and Electronics Engineers Inc., Dec. 2019, pp. 72–77. doi: 10.1109/ISPRAS47671.2019.00017.

[24] "GitHub - dyazincahya/KBBI-SQL-database: Kamus Besar Bahasa Indonesia (KBBI) SQL Database | Total data 115.978 Kata | Tersedia untuk MySQL, SQLite dan PostgreSQL. Juga tersedia untuk format data CSV, JSON, Markdown, PHP Array, XML, DbUnit, HTML." Accessed: Aug. 20, 2025. [Online]. Available: https://github.com/dyazincahya/KBBI-SQL-database

[25] "Pencarian - KBBI VI Daring." Accessed: Aug. 20, 2025. [Online]. Available: https://kbbi.kemdikbud.go.id/

[26] Y. Yanfi, R. Setiawan, H. Soeparno, and W. Budiharto, "Comparison of Spelling Error Correction Algorithms for the Indonesian Language," in *2023 11th International Conference on Information and Education Technology, ICIET 2023*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 443–447. doi: 10.1109/ICIET56899.2023.10111191.

[27] N. Elmobark, "A Comparative Analysis of Python Text Matching Libraries: A Multilingual Evaluation of Capabilities, Performance and Resource Utilization," *International Journal of Environment, Engineering and Education*, vol. 7, no. 1, pp. 48–60, Apr. 2025, doi: 10.55151/ijeedu.v7i1.188.

[28] S. Sathyanarayanan, "Confusion Matrix-Based Performance Evaluation Metrics," *African Journal of Biomedical Research*, pp. 4023–4031, Nov. 2024, doi: 10.53555/ajbr.v27i4s.4345.

[29] M. Ghibran and A. L. Khamaeni, "IMPLEMENTASI WHITE BOX TESTING BERBASIS PATH PADA APLIKASI BERBASIS WEB," *Jurnal Siliwangi*, vol. 9, no. 1, p. 2023.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15219

[30] M. Mintarsih, "Pengujian Black Box Dengan Teknik Transition Pada Sistem Informasi Perpustakaan Berbasis Web Dengan Metode Waterfall Pada SMC Foundation," *Jurnal Teknologi Dan Sistem Informasi Bisnis*, vol. 5, no. 1, pp. 33–35, Feb. 2023, doi: 10.47233/jteksis.v5i1.727.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15219