e- ISSN: 2828–299X

Sistem Deteksi Dan Koreksi Otomatis Penulisan Bahasa Indonesia Menggunakan *Dictionary Lookup* dan *Fuzzy Logic*

Willy Rahma Wijaya
Teknik Informatika, Fakultas Teknik dan Ilmu Komputer, Universitas Nusantara PGRI Kediri
Email: *sayawillyrw@gmail.com

Abstrak - *Ketidakbakuan dalam penulisan bahasa Indonesia di media digital mencakup spektrum yang luas, mulai dari penggunaan bahasa gaul (slang), singkatan, kesalahan penulisan karakter (typo), hingga kesalahan tata bahasa pada kata depan (preposisi). Alat bantu pengecekan ejaan (spell checker) yang ada saat ini seringkali memiliki keterbatasan kosakata, sehingga gagal mengenali kata non-baku yang berkembang di masyarakat maupun entitas spesifik seperti nama daerah. Penelitian ini bertujuan membangun sistem normalisasi teks yang mampu menangani keseluruhan aspek kesalahan ejaan bahasa Indonesia secara otomatis. Metode yang digunakan adalah pendekatan Hybrid yang menggabungkan kecepatan Dictionary Lookup dan fleksibilitas Fuzzy Logic berbasis Levenshtein Distance. Sistem ini didukung oleh korpus data masif berjumlah 633.350 entri, yang terdiri dari dataset kosakata umum dan slang (347.416 kata), dataset entitas wilayah administratif (285.851 kata), serta aturan preposisi. Hasil pengujian menunjukkan sistem mampu memperbaiki ejaan pada berbagai konteks kalimat, baik itu bahasa santai, formal, maupun kalimat yang memuat nama lokasi spesifik, dengan tingkat akurasi yang tinggi dan waktu proses yang efisien*.
Kata Kunci - Fuzzy Logic, Normalisasi Teks, Dictionary Lookup, Koreksi Otomatis.

1. PENDAHULUAN

Perkembangan media sosial dan komunikasi digital telah memicu penggunaan ragam bahasa Indonesia yang tidak baku secara masif. Fenomena ketidakteraturan teks ini meliputi penggunaan kata gaul (*slang*), singkatan (seperti "yg", "kuy"), hingga kesalahan penulisan karakter (*typo*) yang tidak disengaja. Selain itu, kesalahan gramatikal pada penulisan preposisi (kata depan) "di" dan "ke" yang disambung dengan nama tempat (contoh: "disurabaya") menjadi masalah spesifik yang sering diabaikan [1]. Ketidakteraturan ini menjadi tantangan serius dalam *Natural Language Processing* (NLP), karena teks yang kotor menurunkan akurasi analisis sentimen, penerjemahan mesin, dan klasifikasi teks [2].

Untuk mengatasi masalah tersebut, diperlukan teknik normalisasi teks. Pendekatan yang paling umum digunakan adalah metode berbasis kamus atau *Direct Lookup* [3]. Metode ini bekerja dengan mencocokkan kata input secara eksak (*exact match*) terhadap daftar kata baku. Keunggulan utama metode ini adalah kecepatan komputasi yang sangat tinggi ($O(1)$) dan akurasi 100% untuk kata-kata yang sudah terdaftar, seperti singkatan atau bahasa gaul popular [4]. Namun, kelemahan fatal dari *Direct Lookup* adalah ketidakmampuannya menangani kata yang tidak ada di dalam kamus (*Out-of-Vocabulary*), termasuk kesalahan ketik acak (misal: "mkaanan" untuk "makanan").

Di sisi lain, pendekatan berbasis ukuran kemiripan string (*string similarity*) atau *Fuzzy Logic* menawarkan solusi yang lebih fleksibel. Algoritma seperti *Levenshtein Distance* mampu mengukur jarak pengeditan antara dua kata, sehingga dapat mengenali kata yang *typo* berdasarkan kemiripan karakter [5]. Efektivitas pendekatan berbasis kemiripan teks (*text similarity*) ini juga telah dibuktikan dalam penelitian lain, seperti pada sistem penilaian esai otomatis yang menggunakan metode *Cosine Similarity* untuk mencocokkan jawaban siswa dengan kunci jawaban [6]. Meskipun handal dalam menangani *typo*, penerapan *Fuzzy Logic* pada korpus data yang sangat besar memiliki kelemahan dari sisi waktu komputasi yang lambat jika harus membandingkan setiap kata satu per satu.

Berdasarkan analisis terhadap kelebihan dan kekurangan kedua metode tersebut, penelitian ini mengusulkan pendekatan Hybrid yang menggabungkan efisiensi *Direct Lookup* dan fleksibilitas *Fuzzy Logic*. Sistem ini dirancang untuk memprioritaskan pencarian kamus untuk menangani kata *slang* dan entitas wilayah secara instan, dan baru menggunakan *Fuzzy Logic* ketika kata tidak ditemukan. Kebaruan (*novelty*) utama penelitian ini terletak pada integrasi Big Data Leksikal berjumlah 633,353 entri yang mencakup dataset wilayah administratif Indonesia (BPS) [7], sehingga sistem mampu melakukan koreksi ejaan yang komprehensif mulai dari kata sehari-hari hingga nama lokasi spesifik.

2. METODE PENELITIAN

2.1 Arsitektur Data

Dataset ini berasal dari github https://github.com/aryakdaniswara/kbbi-dataset-kbbi-v dan BPS yang dikembangkan lebih lanjut oleh Penulis. Sistem ini dibangun di atas pondasi data leksikal yang sangat besar untuk mencakup segala kemungkinan kesalahan penulisan. Total korpus data yang dikumpulkan berjumlah 633.350 entri, yang dibagi menjadi tiga modul utama:

e- ISSN: 2828–299X

1. Modul Kosakata Umum & Slang (347.416 data): Modul ini adalah inti dari sistem yang menangani percakapan sehari-hari. Data ini mencakup kata baku KBBI, ribuan variasi kata gaul (*slang*), singkatan SMS, dan bentuk *typo* umum yang sering muncul di media sosial [3].
2. Modul Entitas Wilayah (285.851 data): Untuk melengkapi kekurangan *spell checker* biasa, modul ini memuat data hirarki wilayah Indonesia dari BPS (Provinsi hingga Desa) [4]. Tujuannya adalah menangani kesalahan penulisan yang melibatkan nama tempat dan preposisi (contoh: "disurabaya", "kejakarta").
3. Modul Aturan Preposisi (83 aturan): Menangani kesalahan tata bahasa dasar pada kata depan yang sering terbalik penggunaannya.

Tabel 1 Contoh Dataset

Kata Salah	Kata Baku
Dimana	Di mana
Disana	Di sana
Almunium	Aluminium
Ijasah	Ijazah

2.2 *Natural Language Processing*

Natural Language Processing (NLP) merupakan salah satu cabang kecerdasan buatan (*Artificial Intelligence*/AI) yang berfokus pada interaksi antara komputer dan bahasa manusia. Dengan NLP, komputer mampu memahami, menginterpretasikan, serta menghasilkan bahasa yang menyerupai cara manusia berkomunikasi. Proses ini melibatkan berbagai teknik dalam analisis teks maupun ucapan, sehingga dapat diterapkan pada berbagai aplikasi seperti penerjemahan otomatis, asisten virtual, dan analisis sentiment [8].

2.3 Fuzzy *Logic*

Logika Fuzzy sering disebut sebagai *logika baru yang lama* karena meskipun teori dan metode modernnya baru dikembangkan beberapa dekade terakhir, konsep dasarnya sebenarnya telah lama melekat dalam cara manusia berpikir. Logika Fuzzy merupakan pengembangan dari logika Boolean yang tidak hanya mengenal nilai benar atau salah, tetapi juga memungkinkan adanya derajat kebenaran di antara keduanya. Teori ini pertama kali diperkenalkan oleh Dr. Lotfi Zadeh dari University of California, Berkeley, pada era 1960-an sebagai pendekatan untuk memodelkan ketidakpastian. Dalam memahami sistem Fuzzy, terdapat beberapa konsep penting yang perlu diketahui [9].

1. Variabel FuzzyVariabel Fuzzy merupakan variabel yang hendak dibahas dalam suatu sistem Fuzzy contoh umur, temperatur, permintaan dan sebagainya.
2. Fuzzifikasi adalah proses untuk mengubah variabel-variabel non Fuzzy (variabel numerik) menjadi variabel Fuzzy (variabel linguistik) nilai masukan-masukan yang masih dalam bentuk variabel numerik. Sebelum diolah oleh pengendali logika Fuzzy harus diubah terlebih dahulu kedalam variabel Fuzzy. Melalui fungsi keanggotaan yang telah disusun maka nilai-nilai masukan tersebut menjadi reformasi Fuzzy yang berguna nantinya untuk proses pengolahan.

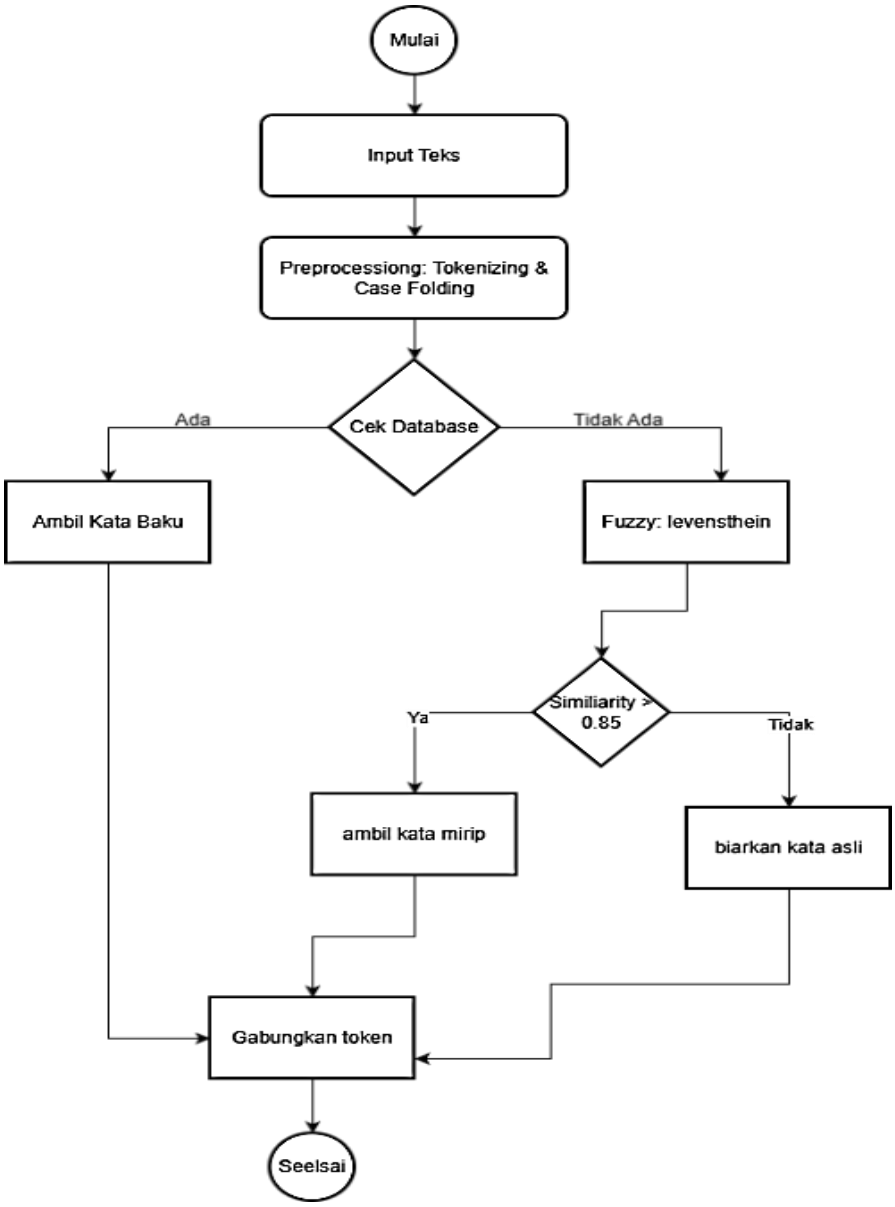
2.4 Dictionary Lookoup

Dictionary lookup adalah metode yang bekerja dengan cara mencocokkan kata input dengan daftar kata atau kamus yang tersedia. Apabila kata yang dimasukkan tidak ditemukan dalam kamus tersebut, maka kata tersebut dikategorikan sebagai kata tidak valid. Metode ini umum digunakan untuk mendeteksi kesalahan penulisan jenis *non-word error* [3].

2.5 Alur program

Sistem dirancang menggunakan alur kerja bertahap untuk memastikan efisiensi waktu komputasi. Sebagaimana diilustrasikan pada Gambar 1, proses dimulai dari input teks, preprocessing, hingga proses normalisasi inti

e- ISSN: 2828–299X



Gambar 1 Flowchart Alur Program

Tahapan alur program adalah sebagai berikut:

1. Input Teks: Memasukkan Teks
2. Prepocessing Tokenisasi dan Case Folding**:** Tahap preprocessing meliputi tokenisasi untuk memecah kalimat menjadi unit kata serta case folding untuk mengubah seluruh karakter menjadi huruf kecil. Tanda baca diperlakukan sebagai pemisah token dan tidak dilibatkan dalam proses normalisasi.
3. Pencarian Kamus (*Dictionary Lookup*): Setiap token dicocokkan dengan basis data 633 ribu kata. Proses ini menggunakan struktur data *Hash Map* dengan kompleksitas waktu O(1) untuk kecepatan tinggi.
4. Logika Samar (*Fuzzy Logic*): Jika token tidak ditemukan di kamus, sistem menghitung jarak kemiripan menggunakan algoritma *Levenshtein Distance*.
5. Penentuan Koreksi**:** Jika skor kemiripan tertinggi melampaui ambang batas (*threshold*) 0.85, kata tersebut dikoreksi. Jika tidak, kata asli dipertahankan.

2.6 Rumus jarak *Levenshtein*

Untuk menangani kesalahan penulisan (*typo*) yang bersifat acak (tidak ada di kamus), sistem menggunakan algoritma *Levenshtein Distance*. Algoritma ini menghitung jumlah operasi minimum (penyisipan, penghapusan,

e- ISSN: 2828–299X

atau substitusi) yang diperlukan untuk mengubah string a menjadi string b. Persamaan matematisnya adalah sebagai berikut:

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) & \text{Jika } min(i,j) = 0, \\ min\begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_i)} \end{cases} & \text{Jika tidak.} \end{cases} \quad \text{.....(1)}$$

Dimana $lev_{a,b}(i,j)$ adalah jarak antara i karakter pertama dari string a dan j karakter pertama dari string b.

2.7 Perhitungan Simialirity

Jarak edit dinormalisasi menjadi skor kemiripan (S) dengan ambang batas $\theta = 0.85$

$$S(a,b) = 1 - \frac{lev(a,b)}{\max(|a|,|b|)} \quad \text{.....(2)}$$

Penelitian ini menetapkan nilai ambang batas (*threshold*) $\theta = 0.85$. Artinya, sebuah kata dianggap sebagai *typo* yang valid untuk diperbaiki jika memiliki tingkat kemiripan minimal 85% dengan kata baku target.

2.8 Algoritma Hybrid

Sistem menggunakan mekanisme *Hybrid* untuk menyeimbangkan antara kecepatan dan kecerdasan koreksi

1. Pencarian Eksak (*Dictionary Lookup*): Langkah pertama adalah mencocokkan kata input dengan *database*. Jika kata ditemukan (misal: "aq" → "aku"), koreksi dilakukan instan (Kompleksitas $O(1)$)
2. Pencarian Samar (*Fuzzy Logic):* Jika kata tidak ada di database dan bukan kata baku, sistem menggunakan algoritma *Levenshtein Distance* [5] untuk mencari kata yang paling mirip secara morfologi.

Sistem mengintegrasikan kedua metode di atas dengan prioritas eksekusi. Fungsi normalisasi $N(w)$ untuk setiap kata w didefinisikan sebagai

$$N(w) = \begin{cases} D(w) & \text{Jika } w \in Kamus \\ ArgMax(S(w,b)) & \text{Jika } max(S) \geq 0.85 \\ w & \text{Jika tidak ditemukan} \end{cases} \quad \text{.....(3)}$$

Metode ini memastikan bahwa kata-kata spesifik (seperti nama desa "Cilodong" atau bahasa gaul "kuy") ditangani secara instan dan akurat oleh Kamus, sementara kesalahan ketik yang tidak terduga (seperti "mkaanan") ditangani oleh *Fuzzy Logic.*

2.9 Streamlit

Streamlit merupakan *framework* berbasis Python yang digunakan untuk membangun aplikasi web dengan antarmuka pengguna yang interaktif, khususnya untuk mendukung pengembangan proyek *data science* dan *machine learning* [10].

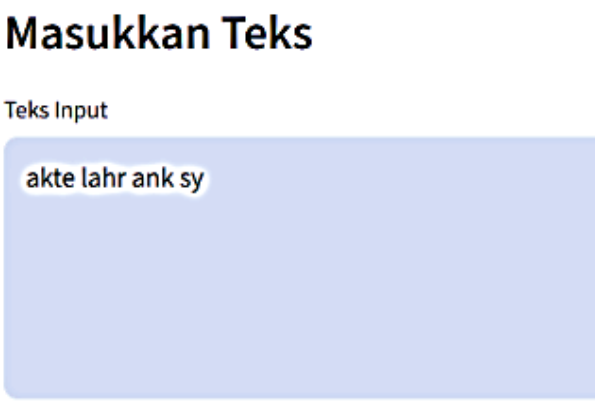
3. HASIL DAN PEMBAHASAN

3.1 Implementasi Sistem

Sistem normalisasi teks ini dikembangkan sebagai aplikasi berbasis web menggunakan bahasa pemrograman Python dengan antarmuka yang dibangun menggunakan *framework* Streamlit. Fokus utama implementasi adalah pada integrasi korpus data berskala besar yang terdiri dari **633.382 entri**, mencakup kosakata umum, bahasa gaul (*slang*), dan data wilayah administratif

Gambar 2 Tampilan Sistem

Antarmuka sistem dirancang sederhana untuk memudahkan pengguna memasukkan kalimat uji. Fitur utama meliputi kotak input teks, tombol proses normalisasi, dan visualisasi hasil yang membandingkan kalimat asli dengan kalimat hasil perbaikan. Sistem juga menampilkan metrik waktu pemrosesan untuk transparansi kinerja.

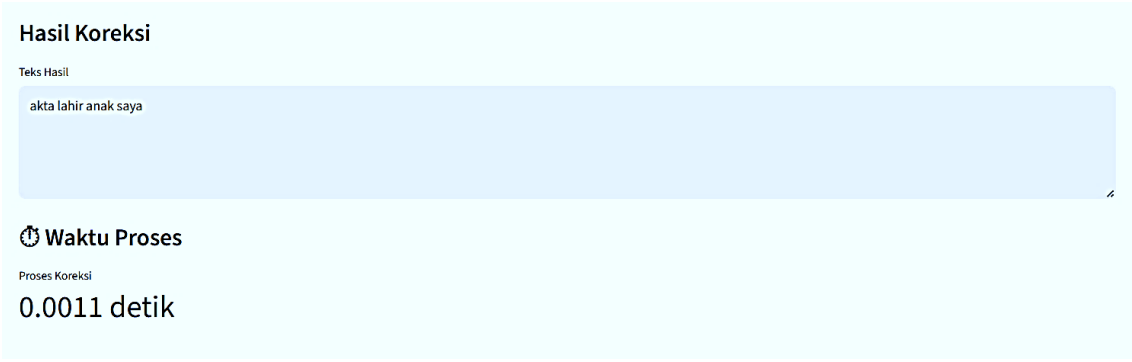


Gambar 3 Masukkan Teks

Masukkan teks yang ingin dianalisis letak kesalahan pada kalimat, pada gambar ada tulisan ”akte lahr ank s” pada *text area*.

Gambar 4 Kata Yang Terdeteksi

Pada gambar tersebut bahwa ada 4 kata yang terdeteksi, 3 diantaranya bersumber dari kbbi/ data dan 1 kata dari kinerja fuzzy logic. Bersumber dari kbbi ialah akte, ank, sy menjadi akta, anak, saya sedangkan lahr menjadi lahir bersumber dari fuzzy.



Gambar 5 Hasil Koreksi Dan Waktu Proses

Pada gambar tersebut ada hasil koreksi di mana kalimat awal ”akte lahr ank sy” menjadi ”akta lahir anak saya” dengan waktu proses 0.00111 detik. Ini menunjukkan proses bekerja dengan baik dan efisien dalam segi watku

e- ISSN: 2828–299X

3.2 Pengujian Kinerja Komputasi

Mengingat besarnya ukuran dataset yang digunakan, efisiensi memori dan waktu eksekusi menjadi parameter kritis. Pengujian kinerja dilakukan pada perangkat keras standar (RAM 16GB, Processor AMD Ryzen 5).

1. Waktu Inisialisasi (*Cold Start*): Saat sistem pertama kali dijalankan, sistem menunjukkan efisiensi tinggi dalam memuat korpus "Big Data". Kecepatan ini dicapai berkat implementasi struktur data yang teroptimasi serta penerapan mekanisme *Caching* yang mencegah pemuatan ulang data yang redundan
2. Kecepatan Normalisasi (*Runtime*): Waktu eksekusi sistem bergantung sepenuhnya pada karakteristik kalimat yang diinputkan oleh pengguna. Kecepatan proses dipengaruhi secara linear oleh panjang kalimat (jumlah token kata) dan tingkat kompleksitas kesalahan ejaan. Kalimat yang mengandung banyak kata *typo* acak akan membutuhkan waktu komputasi sedikit lebih lama karena melibatkan perhitungan matematis *Fuzzy Logic*, dibandingkan kalimat yang hanya berisi kata *slang* yang dapat diproses instan melalui kamus. Meskipun demikian, berkat optimasi algoritma, sistem tetap mampu memberikan respons yang efisien dan cepat untuk berbagai variasi input.

3.3 Pengujian Akurasi Normalisasi

Pengujian fungsional dilakukan untuk memvalidasi kemampuan metode *Hybrid* dalam menangani berbagai variasi kesalahan ejaan. Skenario pengujian mencakup kesalahan *slang*, *typo* karakter, kesalahan preposisi dasar, hingga kesalahan penulisan nama wilayah yang spesifik. Hasil pengujian dirangkum dalam Tabel 1.

Tabel 2 Hasil Pengujian Akurasi Normalisasi “akte lahr ank sy”

Token Asli	Normalisasi	Metode	Status
akte	akta	KBBI	benar
lahr	lahir	Fuzzy	typo acak
ank	anak	KBBI	slang
sy	saya	KBBI	slang

Berdasarkan pengujian pada tabel, sistem berhasil melakukan normalisasi pada seluruh token tidak baku secara akurat. Namun, sistem masih bekerja pada tingkat token, sehingga tidak melakukan normalisasi berbasis frasa, seperti perubahan ‘akta lahr’ menjadi ‘akta lahir. Hal ini menunjukkan bahwa sistem telah optimal pada level leksikal, namun belum mencakup pemrosesan semantik antar token.

3.4Analisis dan Pembahasan

Berdasarkan hasil pengujian pada kalimat uji *“akte lahr ank sy”*, sistem menunjukkan kemampuan normalisasi yang adaptif melalui pendekatan pemrosesan berbasis token (*token-level normalization*). Kalimat uji tersebut mengandung beberapa jenis kesalahan sekaligus, yaitu kesalahan ejaan, kesalahan ketik acak, serta penggunaan singkatan tidak baku. Sistem mampu menangani seluruh kesalahan tersebut dalam satu alur pemrosesan tanpa menimbulkan koreksi yang keliru Berikut adalah analisis mendalam mengenai perilaku sistem:

1. Granularitas Deteksi Per Token
   Analisis menunjukkan bahwa pemecahan kalimat menjadi token tunggal (*akte*, *lahr*, *ank*, *sy*) memungkinkan sistem untuk memberikan perlakuan yang berbeda pada setiap kata sesuai dengan jenis kesalahannya. Token *“akte”* berhasil dikenali sebagai bentuk tidak baku dari kata *“akta”* melalui pencocokan kamus KBBI, sedangkan token *“lahr”* yang merupakan kesalahan ketik acak berhasil dinormalisasi menjadi *“lahir”* menggunakan mekanisme fuzzy matching. Token *“ank”* dan *“sy”* terdeteksi sebagai bentuk singkatan atau slang dan langsung dipetakan ke bentuk bakunya, yaitu *“anak”* dan *“saya”*, menggunakan pendekatan dictionary lookup. Pendekatan ini menunjukkan bahwa sistem tidak melakukan koreksi secara seragam terhadap seluruh kalimat, melainkan memvalidasi dan menormalisasi setiap token secara independen. Dengan demikian, kesalahan dengan karakteristik yang berbeda dapat ditangani secara simultan dalam satu kalimat.
2. Efektivitas Pemisahan Metode Koreksi
   Kasus uji ini juga memperlihatkan efektivitas pemisahan metode koreksi berdasarkan prioritas yang telah dirancang. Token yang ditemukan dalam kamus langsung dinormalisasi tanpa melibatkan perhitungan jarak edit, sehingga proses berjalan cepat dan deterministik. Sementara itu, mekanisme fuzzy matching hanya diaktifkan pada token yang tidak ditemukan dalam kamus, seperti *“lahr”*. Dengan pendekatan ini, sistem berhasil menghindari koreksi berlebihan (*over-correction*) terhadap token yang sebenarnya telah valid.

3. Keterbatasan Pada Tingkat Frasa
   Meskipun sistem berhasil menormalisasi seluruh token menjadi bentuk baku sehingga menghasilkan keluaran *"akta lahir anak saya"*, sistem belum melakukan normalisasi pada tingkat frasa. Dalam konteks bahasa formal, frasa *"akta lahir"* sering kali digunakan sebagai padanan dari *"akta kelahiran"*. Namun, karena sistem dirancang untuk bekerja pada tingkat token, perubahan berbasis frasa atau semantik lintas kata belum diterapkan. Hal ini menunjukkan bahwa sistem telah optimal pada level leksikal, tetapi masih memiliki ruang pengembangan pada level frasa atau semantik.

4. SIMPULAN

Berdasarkan hasil perancangan dan pengujian sistem normalisasi teks berbasis *Hybrid*, penelitian ini menyimpulkan bahwa integrasi metode *Dictionary Lookup* dan *Fuzzy Logic* terbukti efektif dalam menangani ragam kesalahan ejaan yang kompleks. Mekanisme prioritas yang diterapkan mampu menyeimbangkan akurasi dan kecepatan, di mana kata slang ditangani secara instan melalui kamus, sedangkan algoritma Levenshtein Distance dengan ambang batas 0,85 berfungsi optimal sebagai jaring pengaman untuk memperbaiki kesalahan ketik (*typo*) karakter yang bersifat acak.

Selain itu, kontribusi dataset wilayah administratif (BPS) yang berjumlah 285.851 entri menjadi faktor pembeda utama dalam peningkatan akurasi sistem dibandingkan spell checker konvensional. Data ini memungkinkan sistem untuk mendeteksi kesalahan penulisan preposisi tempat yang sering terjadi, seperti memisahkan kata "disurabaya" atau "kecilodong" menjadi dua token terpisah yang valid. Dari segi performa, penerapan struktur data Hash Map dan mekanisme Caching terbukti sangat efisien dalam memproses dataset berskala besar, dengan latensi rata-rata yang sangat rendah sebesar 0,0011 detik per kata, membuktikan bahwa penggunaan *Big Data* dapat dilakukan tanpa mengorbankan responsivitas aplikasi.

5. SARAN

Berdasarkan evaluasi terhadap kinerja sistem, terdapat beberapa aspek yang dapat dikembangkan pada penelitian selanjutnya. Pengembangan utama yang disarankan adalah penerapan analisis konteks (context awareness) menggunakan pendekatan N-Gram atau model bahasa berbasis Deep Learning seperti BERT, mengingat sistem saat ini bekerja pada tingkat token (kata per kata). Pendekatan tersebut diperlukan agar sistem mampu menangani ambiguitas makna kata (homonim) serta melakukan normalisasi berbasis konteks antar kata dalam satu kalimat. Selain itu, penelitian selanjutnya dapat memperluas cakupan sistem dengan menambahkan penanganan tanda baca dan struktur kalimat, sehingga proses normalisasi tidak hanya terbatas pada ejaan kata, tetapi juga mencakup aspek sintaksis teks secara lebih komprehensif. Pengembangan ini diharapkan dapat meningkatkan kualitas normalisasi pada teks formal maupun semi-formal. Selanjutnya, sistem ini memiliki potensi besar untuk dikembangkan menjadi Application Programming Interface (API) publik agar dapat diintegrasikan secara real-time dengan berbagai platform eksternal, seperti layanan chatbot, sistem analisis sentimen, maupun fitur moderasi komentar di media sosial.

DAFTAR PUSTAKA

[1] Badan Pengembangan dan Pembinaan Bahasa, *Pedoman Umum Ejaan Bahasa Indonesia (PUEBI)*, 4th ed. Jakarta: Kementerian Pendidikan dan Kebudayaan, 2016.

[2] B. Liu, *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. New York: Cambridge University Press, 2015.

[3] M. D. Etsa, H. Sujaini, and N. Safriadi, "Pengaruh Metode Dictionary Lookup pada Cleaning Korpus Terhadap Akurasi Mesin Penerjemah Statistik Indonesia–Melayu Pontianak," *J. Edukasi dan Penelit. Inform.*, vol. 4, no. 1, p. 49, Jun. 2018, doi: 10.26418/JP.V4I1.24595.

[4] A. R. Wibowo and H. S. Sholihin, "Normalisasi Teks Bahasa Indonesia Berbasis Kamus untuk Penanganan Singkatan dan Kata Gaul di Media Sosial," *J. Teknol. Inf.*, vol. 15, no. 2, pp. 120–129, 2021.

[5] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Sov. Phys. Dokl.*, vol. 10, no. 8, pp. 707–710, 1966.

[6] M. R. Arifuddin, I. A. Rafiq, and R. Mubarok, "Sistem Cerdas Penilaian Ujian Essay Menggunakan Metode Cosine Similarity," *Gener. J.*, vol. 7, no. 1, pp. 31–38, 2023, doi: 10.29407/gj.v7i1.18318.

[7] Badan Pusat Statistik, "Master File Wilayah Provinsi/Kabupaten/Kecamatan/Desa Tahun 2024," 2024, *Jakarta*.

[8] R. Danar, D. M. Kom, M. M. Kom, A. Bahtiar, M. Kom, and I. Ali, *Dasar Dasar Natural Language Processing (NLP)*. Tangerang: Minhaj Pustaka, 2024. Accessed: Dec. 13, 2025. [Online]. Available: https://repository.minhajpustaka.id/publications/593976/

[9] V. S. Aryadi, R. Wulanningrum, and P. Kasih, "Implementasi Fuzzy Logic Identifikasi Gesture Tangan

e- ISSN: 2828–299X

Untuk Deteksi Kerusuhan," *Semin. Nas. Teknol. Sains*, vol. 1, no. 1, pp. 44–50, Feb. 2022, doi: 10.29407/STAINS.V1I1.1487.
[10] T. Richards, *Getting Started with Streamlit for Data Science*. Birmingham, UK: Packt Publishing, 2021.