BAB II
LANDASAN TEORI

2.1 Penerjemah Otomatis

Penerjemah otomatis adalah teknologi berbasis kecerdasan buatan yang memungkinkan konversi teks dari satu bahasa ke bahasa lain secara instan dan gratis. Sistem ini dikembangkan melalui berbagai metode, seperti *Rule-Based Machine Translation* (RBMT), *Statistical Machine Translation* (SMT), dan *Neural Machine Translation* (*NMT*) (Bun San, Sujaini, and Hadari Nawawi 2023). Teknologi *NMT*, yang paling umum digunakan saat ini, mengandalkan jaringan saraf tiruan untuk memahami konteks sehingga menghasilkan terjemahan yang lebih akurat dibandingkan metode sebelumnya. Namun, meskipun memberikan kemudahan, keakuratan hasil terjemahan masih menjadi perdebatan, terutama dalam menangkap makna kontekstual dan gramatikal dari bahasa sumber ke bahasa sasaran (Citra, n.d.).

Penelitian ini akan menggunakan **@vitalets/google-translate-api**, sebuah *library* berbasis *NPM* yang memungkinkan akses ke layanan *Google Translate* melalui permintaan *HTTP*. *API* ini tidak memerlukan kredensial resmi dan dapat diintegrasikan dengan mudah ke dalam aplikasi chat untuk mendukung penerjemahan otomatis secara real-time. Keunggulan utama dari pendekatan ini adalah kemudahan implementasi dan latensi rendah dalam menghasilkan terjemahan. Namun, terdapat keterbatasan seperti ketergantungan pada layanan pihak ketiga dan potensi perubahan dalam struktur *API* yang dapat memengaruhi keberlanjutan penggunaannya.

2.2 Text Processing

Pra-pemrosesan teks (text preprocessing) adalah langkah untuk mengubah dokumen dan query menjadi sekumpulan kata-kata yang dapat diindeks. Tahapan ini merupakan bagian awal dari text mining yang bertujuan untuk mempersiapkan teks agar bisa diproses lebih lanjut dalam tahap berikutnya. Proses ini penting untuk

mengonversi teks menjadi data yang terstruktur. Dalam pra-pemrosesan teks, terdapat beberapa tahapan seperti *case folding, filtering*, dan *tokenisasi* (Hidayat et al., 2020).

a) *Case folding* adalah proses untuk menyeragamkan bentuk huruf menjadi huruf kecil (lowercase), dengan tujuan menjadikan lowercase sebagai bentuk standar pada teks
b) *Filtering* adalah proses untuk menghapus kata-kata yang tidak relevan atau kata-kata yang termasuk dalam daftar stopword
c) *Tokenisasi* adalah proses memisahkan setiap kata dalam dokumen sehingga setiap kata tersebut bisa berdiri sebagai unit yang terpisah

2.3 N-Gram

Metode *N-Gram* telah diterapkan untuk menyelesaikan berbagai masalah, seperti prediksi kata, koreksi ejaan, pengenalan suara, koreksi terjemahan, dan pencarian string. Salah satu keuntungan utama dari metode *N-Gram* adalah kemampuannya untuk bekerja secara independen terhadap bahasa tertentu (Agung et al., n.d.). Dalam konteks koreksi ejaan, *N-Gram* merujuk pada urutan *N* huruf yang membentuk sebuah kata atau string. *N-Gram* juga dapat digunakan untuk mengukur kesamaan antara dua string dengan cara menghitung jumlah *N-Gram* yang tumpang tindih di antara keduanya. Semakin banyak *N-Gram* yang sama antara dua kalimat, semakin mirip kedua kalimat tersebut.(Situmorang et al., 2021)

N-Gram dapat dikategorikan berdasarkan jumlah huruf yang digunakan dalam pemisahan setiap kata, yang diwakili dengan nilai n. Nilai $\boldsymbol{N} = 1$ disebut *Unigram*, $\boldsymbol{N} = 2$ disebut *Bigram*, dan $\boldsymbol{N} = 3$ disebut *Trigram,Trigram* yang mempertimbangkan tiga kata atau karakter secara bersamaan dalam satu unit analisis. Setiap variasi memiliki keunggulan dan keterbatasan masing-masing, di mana *Unigram* cenderung kurang memahami konteks, sementara *Trigram* lebih efektif dalam menangkap makna dalam teks yang lebih panjang. (Faried Ikhwan & Yunita, 2024)

2.4 Levenshtein Distance

Algoritma *Levenshtein Distance* adalah algoritma yang digunakan untuk mengukur perbedaan antara dua kata dalam sebuah string. Dengan menggunakan algoritma ini, sistem dapat menghitung perbedaan antara dua string, yang mana string-string tersebut diambil satu per satu dari dalam database dan dibandingkan. Kata dengan perbedaan jarak yang paling kecil akan dianggap sebagai kata yang paling akurat dibandingkan dengan kata yang dicari oleh pengguna (Adawiyah & Saragih, 2022). Setelah menghitung perbedaan jarak antara kata yang dimasukkan, sistem akan mengembalikan beberapa kata dengan perbedaan jarak terkecil sebagai rekomendasi kata.

Levenshtein Distance pertama kali diperkenalkan oleh Vladimir Levenshtein pada tahun 1965. Algoritma ini menghitung perbedaan antara dua string dengan melakukan tiga jenis operasi berdasarkan keadaan kata yang dimasukkan oleh pengguna. Operasi-operasi tersebut adalah:

1. *Insertion* (Penyisipan Karakter**) :** Operasi ini dilakukan dengan cara menambahkan satu karakter ke dalam string.
2. *Deletion* (Penghapusan Karakter) **:** Operasi ini menghapus satu karakter dari string
3. *Subtitution* (Penukaran Karakter) **:** Operasi ini mengganti satu karakter dengan karakter lain dalam string

Ketiga operasi ini digunakan untuk menghitung jarak edit yang dibutuhkan agar sebuah string dapat berubah menjadi string lainnya.

Persamaan 2.1 *Levenshtein Distance*

$$D(i,j) = \begin{cases} \mathbf{Max(i,j)} \\ \mathbf{Min} \begin{cases} \mathbf{D(i-1,j)+1} \\ \mathbf{D(i,j-1)+1} \\ \mathbf{D(i=1,j=1)+cost} \begin{cases} +\mathbf{1}\ \boldsymbol{if}\ \boldsymbol{x(i)!=y(j)} \\ +\mathbf{0}\ \boldsymbol{if}\ \boldsymbol{x(i)=y(j)} \end{cases} \end{cases} \end{cases} \quad (2.1)$$

Keterangan:

- D (i,j) adalah jarak edit minimum antara substring pertama dengan panjang i dan substring kedua dengan panjang j
- $\boldsymbol{D}(i-1,j)+1$ Menghapus karakter dari string pertama (deletion)
- $D(i,j-1)+1$ Menyisipkan karakter ke string pertama (insertion)
- $D(i=1,j=1)+cost$ Mengganti karakter (substitution)

2.5 Probabilitas *N-Gram*

Penelitian ini menggunakan *Maximum Likelihood Estimation (MLE)* untuk menghitung probabilitas *N-Gram*, di mana probabilitas dihitung berdasarkan frekuensi relatif suatu *N-Gram* dalam data latih (Dewi & Qoiriah, 2021).

1. Maximum Likelihood Estimation (MLE)

   Maximum Likelihood Estimation (MLE) adalah metode statistik yang digunakan untuk memperkirakan probabilitas suatu kejadian berdasarkan data historis. menghitung probabilitas dalam model *N-Gram*, metode yang umum digunakan adalah *Maximum Likelihood Estimation (MLE)***.**(Dasgupta et al., 2024) MLE menghitung probabilitas kemunculan kata berdasarkan frekuensi dalam kumpulan teks yang diberikan Dalam Penelitian ini menggunakan *Unigram* dan *Bigram*. Persamaan 2.2 *N-Gram Unigram dan* Persamaan 2.3 *N-Gram Bigram*

- Rumus Perhitungan Probabilitas *MLE Unigram*

$$\mathrm{P}(W_i|) = \frac{\mathrm{C}(W_i)}{\mathrm{N}} \quad (2.2)$$

- Rumus Perhitungan Probabilitas *MLE Bigram*

$$\mathrm{P}(W_i|W_{i-1}) = \frac{\mathrm{C}(W_{i-1},W_{i,})}{\mathrm{C}(W_{i-1})} \quad (2.3)$$

Keterangan :

- P(W_i) adalah probabilitas kata W_i muncul
- C(W_i) adalah jumlah kemunculan kata W_i dalam korpus
- N adalah jumlah total kata dalam korpus
- P($W_i|W_{i-1}$) adalah probabilitas kata W_i muncul setelah kontek W_{i-1}
- C(W_{i-1}, W_i) adalah jumlah kemunculan *Bigram*
- C(W_{i-1}) adalah jumlah jumlah kemunculan kata sebelumnya

Implementasi pada sistem *autocorrect*, metode ini digunakan untuk menentukan probabilitas kandidat perbaikan terbaik. Jika suatu urutan kata tidak pernah muncul dalam data pelatihan, probabilitasnya akan nol.

2. Laplace Smoothing

Perhitungan probabilitas *N-Gram* terkadang menghasilkan nilai probabilitas nol, yang dapat menyebabkan masalah dalam analisis lebih lanjut. Untuk mengatasi hal ini, digunakan teknik *Laplace Smoothing*, di mana setiap frekuensi kemunculan kata ditambah dengan suatu konstanta agar tidak ada probabilitas nol. Teknik ini memastikan bahwa setiap kata dalam dataset tetap memiliki probabilitas yang masuk akal tanpa mendistorsi distribusi secara signifikan.(Dasgupta et al., 2024) Persamaan 2.4 *MLE Unigram dan* Persamaan 2.5 *MLE Bigram*

- Rumus Perhitungan Probabilitas *MLE Unigram dan Laplace Smoothing*

$$\mathrm{P}(W_i) = \frac{\mathrm{C}(W_i) + 1}{\mathrm{N} + \mathrm{V}} \quad (2.4)$$

- Rumus Perhitungan Probabilitas *MLE Bigram dan Laplace Smoothing*

$$\mathrm{P}(W_i|W_{i-1}) = \frac{\mathrm{C}(W_{i-1},W_{i,}) + 1}{\mathrm{C}(W_{i-1}) + \mathrm{V}} \quad (2.5)$$

Keterangan :

- P(W_i) adalah probabilitas kata W_i muncul
- C(W_i) adalah jumlah kemunculan kata W_i dalam korpus

- N adalah jumlah total kata dalam korpus
- P($W_i|W_{i-1}$) adalah probabilitas kata W_i muncul setelah kontek W_{i-1}
- C(W_{i-1}, W_i) adalah jumlah kemunculan *Bigram*
- C(W_{i-1}) adalah jumlah jumlah kemunculan kata sebelumnya
- V adalah jumlah kata unik

2.6 Pengukuran Akurasi

Pengukuran akurasi adalah teknik evaluasi yang bertujuan untuk menilai sejauh mana suatu sistem mampu menghasilkan output yang sesuai dengan target yang diharapkan. Dalam penerapannya pada sistem koreksi kata berbasis metode *N-Gram*, akurasi dimanfaatkan untuk mengukur efektivitas sistem dalam memperbaiki kesalahan pengetikan sekaligus mengenali bahasa secara akurat.(Shalihah et al., 2023)

akurasi dihitung dengan membandingkan jumlah koreksi yang benar dengan keseluruhan jumlah koreksi yang dihasilkan oleh sistem, yang dirumuskan dalam persamaan tertentu untuk memperoleh nilai persentasenya. Persamaan untuk menghitung akurasi dituliskan sebagai berikut:

Persamaan 2. 6 Akurasi

$$\text{Akurasi} = \frac{Kb}{\text{Kb+Ks}} \; x \; 100 \; \% \quad (2.6)$$

Keterangan:

- Kb = Jumlah prediksi atau koreksi benar yang dihasilkan oleh sistem
- Ks = Jumlah prediksi atau koreksi salah yang dihasilkan oleh sistem

2.7 Penelitian Terdahulu

Untuk mendukung penelitian ini diperlukan penelitian – penelitian terdahulu yang terkait deng, yang di tampilkan pada tabel 2.2

Tabel 2.2 Penelitian Terkait

Masalah	Hasil Penelitian	Metode yang digunakan	Objek Penelitian	Landasan Literatur
Pengolahan teks masih memiliki tingkat kesalahan tinggi dalam koreksi ejaan otomatis.	Metode berbasis *N-Gram* dan Edit Distance berhasil meningkatkan akurasi koreksi ejaan.	*N-Gram* Probabilistic Model, Edit Distance		(Dewi & Qoiriah, 2021) Implementasi Algoritma Jaro-Winkler Distance dan *N-gram* untuk Deteksi dan Prediksi Perbaikan Kesalahan Penulisan Kata Bahasa Indonesia pada Karya Tulis Ilmiah Mahasiswa
Kesalahan ejaan dalam teks bahasa Indonesia menyebabkan ketidaksesuaian hasil analisis data	Penggunaan algoritma Damerau-Levenshtein dengan *N-gram* menghasilkan akurasi koreksi hingga 96%	Damerau-Levenshtein dan *N-gram*	Data pelatihan diambil dari bahasa Indonesia kamus dan TED-Monolingual-Parallel-Corpus bagian diekstraksi dari www.ted.com.	(Anggreni et al,2024) SPELLING ERROR CORRECTION IN INDONESIAN USING DAMERAU-LEVENSHTEIN DISTANCE DAN N-GRAM

Masalah	Hasil Penelitian	Metode yang digunakan	Objek Penelitian	Landasan Literatur
Kesalahan ejaan dalam pencarian hadis menyebabkan informasi yang dicari tidak ditemukan.	Implementasi metode *N-Gram* dan Jaccard Similarity untuk koreksi ejaan meningkatkan akurasi pencarian dengan precision 50%-100% dan recall 100%.	*N-Gram*, Jaccard Similarity	Data berupa dokumen bahasa Indonesia dan Inggris sejumlah 110 dokumen	(Shalihah et al. 2023) PENERAPAN METODE N-GRAM UNTUK MEMPERBAIKI KESALAHAN PENULISAN EJAAN KATA KUNCI PADA APLIKASI PENCARIAN HADIS
Sulitnya mengoreksi kesalahan ketik berbasis konteks	Hasilnya menunjukkan bahwa koreksi kesalahan ejaan peka konteks menyumbang deteksi dan koreksi lebih dari 96%(F1) kesalahan.	Deep Learning-Based Context-Sensitive Model	Data teks dengan kesalahan ketik berbasis konteks	(Lee, Kim, and Kwon 2020)Deep Learning-Based Context Sensitive Spelling Typing Error Correction
Kesalahan umum dalam penulisan huruf kapital dan tanda baca sesuai PUEBI	sistem berhasil mendeteksi kesalahan penggunaan tanda baca dan penulisan huruf kapital pada karya tulis ilmiah menggunakan algoritma Boyer-Moore. Sistem menghasilkan nilai rata-rata presisi dan recall sebesar 0,969 dan 0,976.	Algoritma Boyer-Moore	Dataset berupa Beberapa artikel.	(Ilmiyah and Qoiriah 2021) Sistem Deteksi Kesalahan Tanda Baca dan Huruf Kapital Pada Karya Tulis Ilmiah Berbahasa Indonesia Menggunakan Algoritma Boyer-Moore

Masalah	Hasil Penelitian	Metode yang digunakan	Objek Penelitian	Landasan Literatur
Kesulitan dalam mendeteksi dan mengoreksi kesalahan ejaan dalam bahasa yang kurang terwakili	Hasil yang dapat diperbaiki meningkat secara signifikan dan kompleksitas pemrosesan informasi menurun. Kemungkinan koreksi kesalahan PAK meningkat dari 0,5557 menjadi 0,7797 atau lebih, dan indikator padat karya menurun dari 2,22 menjadi 0,96.	Sentence-Level *N-Gram* Context Feature	Data teks dalam bahasa Amharic, Afaan Oromo, dan Tigrigna	(Tsegay Mullu Kassa 2021) Sentence Level N-Gram Context Feature in Real-Word Spelling Error Detection and Correction: Unsupervised Corpus Based Approach
Kesalahan ejaan dalam teks bahasa Vietnam menyebabkan kesalahan pemahaman dan interpretasi informasi	Model menggunakan BERT pra-terlatih dan model bahasa *N-gram* meningkatkan skor F1 hingga 14% dibanding metode lain	*N-Gram*, BERT	Dataset Berupa Kumpulan artikel	(Tien et al. 2022) Vietnamese Spelling Error Detection and Correction Using BERT and N-gram Language Model
Kesalahan tipografi pada chatbot akibat kombinasi bahasa “Manglish” dan typo menyebabkan respons chatbot yang tidak akurat	Metode overlapping *N-gram* dengan algoritma berbasis aturan berhasil mendeteksi 86,2% kesalahan dan memperbaiki 84,3%	Kombinasi metode embedding *N-Gram* dan algoritma berbasis aturan	Kesalahan 150.000 Kata tipografi dalam bahasa “Manglish”	(Anbananthen et al., 2022) Typographic Error Identification and Correction in Chatbot Using N-Gram Overlapping Approach

Masalah	Hasil Penelitian	Metode yang digunakan	Objek Penelitian	Landasan Literatur
Optimasi deteksi dan koreksi kesalahan dalam dokumen elektronik	Kombinasi analisis morfologi multilevel dengan model *N-Gram* meningkatkan keandalan informasi dalam sistem manajemen dokumen elektronik	Multilevel Morphological Analysis & *N-Gram*	Dokumen elektronik dengan kesalahan ejaan dari Web (corpora)	(Jumanov and Karshiev 2020) Mechanisms for optimization of detection and correction of text errors based on combining multilevel morphological analysis with N-Gram models
Kesalahan ejaan real-word dalam bahasa Urdu menyebabkan makna kalimat menjadi ambigu dan sulit dipahami	Model berbasis *N-Gram* dan pembelajaran mesin mencapai akurasi hingga 83,67% dalam koreksi kesalahan real-word	Levenshtein dan model bahasa *N-Gram*	kumpulan data dari dua korpus yang ada yaitu korpus paralel bahasa Inggris-bahasa Urdu1 dan korpus bahasa Urdu monolingual 2	(Aziz et al. 2023) Real Word Spelling Error Detection and Correction for Urdu Language