

**IMPLEMENTASI ALGORITMA *DAMERAU-LEVENSHTTEIN*
DISTANCE DAN PENDEKATAN *DEEP NEURAL NETWORK*
DALAM ANALISIS DAN PERBAIKAN KESALAHAN
EJA ATAU PENULISAN *REAL-WORD ERROR*
BAHASA INDONESIA**

SKRIPSI

RIFQI ALNAHWANDI PUTRA

201402127



**PROGRAM STUDI S-1 TEKNOLOGI INFORMASI
FAKULTAS KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2025**

**IMPLEMENTASI ALGORITMA *DAMERAU-LEVENSHT*EIN
DISTANCE DAN PENDEKATAN *DEEP NEURAL NETWORK*
DALAM ANALISIS DAN PERBAIKAN KESALAHAN
EJA ATAU PENULISAN *REAL-WORD ERROR*
BAHASA INDONESIA**

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah
Sarjana Teknologi Informasi

RIFQI ALNAHWANDI PUTRA

201402127



**PROGRAM STUDI S-1 TEKNOLOGI INFORMASI
FAKULTAS KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA**

MEDAN

2025

PERSETUJUAN

Judul : Implementasi Algoritma Damerau-Levenshtein
Distance Dan Pendekatan Deep Neural Network
Dalam Analisis Dan Perbaikan Kesalahan Eja Atau
Penulisan Real-Word Error Bahasa Indonesia

Kategori : Skripsi

Nama : Rifqi Alnahwandi Putra

Nomor Induk Mahasiswa : 201402127

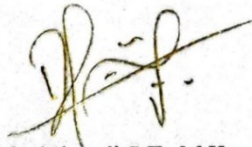
Program Studi : Sarjana (S1) Teknologi Informasi

Fakultas : Ilmu Komputer Dan Teknologi Informasi
Universitas Sumatera Utara

Medan, 10 Januari 2025

Komisi Pembimbing :

Pembimbing 2



Dedy Arisandi S.T., M.Kom.
NIP. 197908312009121002

Pembimbing 1



Ivan Jaya S.Si., M.Kom.
NIP. 198407072015041001

Diketahui/disetujui oleh

Program Studi S1 Teknologi Informasi
Ketua,



Dedy Arisandi S.T., M.Kom.
NIP. 197908312009121002

PERNYATAAN

IMPLEMENTASI ALGORITMA DAMERAU-LEVENSHTTEIN DISTANCE
DAN PENDEKATAN DEEP NEURAL NETWORK DALAM ANALISIS
DAN PERBAIKAN KESALAHAN EJA ATAU PENULISAN
REAL-WORD ERROR BAHASA INDONESIA

SKRIPSI

Saya mengakui bahwa skripsi ini merupakan hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 10 Januari 2025



Rifqi Almahwandi Putra

201402127



UCAPAN TERIMA KASIH

Segala puji dan syukur penulis panjatkan kepada Allah *Subhanahu Wa Ta'ala* atas rahmat, hidayah, dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi ini yang berjudul “Implementasi Algoritma *Damerau-Levenshtein Distance* Dan Pendekatan *Deep Neural Network* Dalam Analisis Dan Perbaikan Kesalahan Eja Atau Penulisan *Real-Word Error* Bahasa Indonesia” Shalawat dan salam semoga selalu tercurah kepada Nabi Muhammad Salallahu Alaihi Wasallam, beserta keluarga, sahabat, dan para pengikutnya.

Skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer pada Program Studi S1 Teknologi Informasi, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara. Selama proses penyusunan, penulis menghadapi berbagai tantangan, namun berkat bimbingan, dukungan, dan bantuan dari berbagai pihak, semua kesulitan tersebut dapat teratasi.

Dengan penuh rasa hormat dan syukur, penulis ingin menyampaikan apresiasi yang mendalam kepada semua pihak yang telah berkontribusi dalam penyelesaian skripsi ini.

1. Kepada kedua orang tua penulis, Bapak Sahril dan Emak Heni (Sri Handa yani) yang senantiasa mendidik, mendukung, mendoakan, dan memberi makan penulis tiada henti siang dan malam, serta semangat yang selalu diberikan oleh kedua adik saya Dhea Muthia Sari dan M. Zamroni Putra, sehingga penulis tetap semangat dan kuat hingga pada tahap penyelesaian skripsi ini.
2. Bapak Ivan Jaya S.Si., M.Kom. selaku Sekretaris Program Studi Teknologi Informasi Universitas Sumatera Utara dan Dosen Pembimbing I saya yang telah memberi bimbingan, masukan, motivasi, kritik hingga saran yang sangat berguna untuk penulis sehingga penulis dapat menyelesaikan skripsi ini.
3. Bapak Dedy Arisandi S.T., M.Kom. selaku Ketua Program Studi Teknologi Informasi Universitas Sumatera Utara dan Dosen Pembimbing II saya yang telah memberi bimbingan, dukungan, motivasi, serta kritik dan saran yang

mendukung, yang telah diberikan kepada penulis sehingga penulis dapat menyelesaikan skripsi ini.

4. Ibu Dr. Maya Silvi Lydia, B.Sc., M.Sc. selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi.
5. Bapak dan Ibu Dosen Fasilkom-TI USU yang telah memberikan ilmu baik di kelas perkuliahan maupun kegiatan akademik lainnya.
6. Seluruh staff Fasilkom-TI USU yang telah membantu dalam segala urusan administrasi selama masa perkuliahan.
7. Teman-teman sukses ‘bermain dan belajar’, Ikhsan, Raissa, Amira, Anggi, Raja, dan Rocky yang selalu mendukung dan menyemangati satu sama lain.
8. Kepada teman-teman seperjuangan saya, ‘Jemut Squad’ , Danny, Riski, Ikhwani, Felix, Logi, Ridho, Rio, dan Hafazh.
9. Kepada Eric Martin selaku teman yang berjasa dan telah rela memberi dukungan, semangat, dan saran yang sangat berguna kepada penulis dalam pengerjaan skripsi ini.
10. Kepada sahabat ‘RPE’, Wilbert, Habibi, Fauzan Mono, Fauzan Botak, Tarish, Rehan Rere, Davin, dan Jodod yang senantiasa memberikan canda tawa dan semangat kepada penulis.
11. Kepada teman-teman ‘spy’, Bertrand Kaca Film, Roy Oli Samping, dan Yola, dan tidak lupa juga Bagas Koping, Yoka Rem Depan yang banyak menemani dan selalu memberi penulis semangat selama pengerjaan skripsi ini. Dan tidak lupa Bang Ridwan (DBIR) sebagai penyedia tempat yang nyaman untuk pengerjaan tiap proses skripsi ini.
12. Kepada teman-teman, ‘Bismillah Cumlaude’ sekaligus teman lomba saya, Afdoni dan M.Ridho yang senantiasa selalu mendukung penulis.
13. Kepada teman-teman seperjuangan yang membersamai penulis selama proses seminar hasil dan sidang meja hijau Pretty, Stephani, Kevin, Rere, dan Monica.
14. Teman-teman seperjuangan Teknologi Informasi Angkatan 2020 terutama Kom A.
15. Seluruh keluarga dan teman-teman lain yang tidak dapat disebutkan satu persatu, yang juga turut serta memberi doa, dukungan, dan semangat kepada

penulis hingga berhasil menyelesaikan skripsi ini. Penulis mengucapkan terimakasih.

16. *And the Last but not least*, dengan penuh rasa syukur, terima kasih kepada Syavira Nur Khairani atas kebaikan hatinya untuk menemani penulis melewati semua petualangan hingga suka duka penulis sejak Maret 2023. Dengan dukungan, doa dan semangat yang diberikannya selalu kepada penulis sehingga penulis dapat menyelesaikan skripsi dengan sebaik-baiknya.

Semoga Allah *Subhanahu Wa Ta'ala* memberikan balasan yang berlipat ganda kepada setiap orang yang telah membantu dalam penulisan dan penyelesaian tugas akhir ini.

Medan, 10 Januari 2025

Rifqi Alnahwandi Putra



ABSTRAK

Komunikasi yang efektif, baik secara verbal maupun tertulis, merupakan aspek penting dalam kehidupan manusia untuk menyampaikan informasi dan membangun hubungan yang harmonis. Dalam komunikasi tertulis, kesalahan dalam penulisan, seperti *non-words* (kata yang tidak ditemukan dalam kamus) dan *real-words* (kata yang terdapat dalam kamus tetapi penggunaannya tidak sesuai konteks), dapat menghambat pemahaman pesan. Meskipun teknologi telah mempermudah proses menulis, tantangan dalam mendeteksi dan memperbaiki kesalahan penulisan, terutama yang berkaitan dengan konteks, masih sering terjadi. Penelitian ini bertujuan untuk merancang sebuah sistem yang mampu mengidentifikasi dan memperbaiki *real-word errors* pada teks berbahasa Indonesia dengan menggunakan algoritma *Damerau-Levenshtein Distance* serta pendekatan *Deep Neural Network* (DNN). Algoritma *Damerau-Levenshtein Distance* berfungsi menghitung jarak edit dengan mempertimbangkan operasi *delete*, *insert*, *substitution*, dan *transposition*, sehingga sangat cocok untuk mengatasi kesalahan ketik. Sementara itu, pendekatan *Deep Neural Network* digunakan untuk menilai kesesuaian kandidat kata berdasarkan konteks kalimat secara keseluruhan, di mana kandidat dengan kesesuaian tertinggi akan dipilih sebagai hasil koreksi. Sistem yang dibangun berhasil mencapai tingkat akurasi sebesar 98.56% dan tingkat keberhasilan koreksi kata sebesar 94.17%. Hasil penelitian ini membuktikan bahwa kombinasi algoritma *Damerau-Levenshtein Distance* dan pendekatan *Deep Neural Network* memiliki kemampuan yang baik dalam mendeteksi dan memperbaiki kesalahan kata, khususnya *real-word errors*.

Kata kunci: *Real-word errors*, Damerau-Levenshtein Distance, Deep Neural Network, Natural Language Processing, koreksi ejaan.

*IMPLEMENTATION OF DAMERAU-LEVENSHTEIN DISTANCE ALGORITHM
AND DEEP NEURAL NETWORK APPROACH IN ERROR ANALYSIS AND
CORRECTION SPELLING OR WRITING REAL-WORD ERRORS
INDONESIAN LANGUAGE*

ABSTRACT

Effective communication, both verbal and written, is an important aspect of human life to convey information and build harmonious relationships. In written communication, errors in writing, such as non-words (words not found in the dictionary) and real-words (words found in the dictionary but used out of context), can hinder message comprehension. Although technology has made the writing process easier, the challenge of detecting and correcting writing errors, especially those related to context, is still common. This research aims to design a system that is able to identify and correct real-word errors in Indonesian text using the Damerau-Levenshtein Distance algorithm and the Deep Neural Network (DNN) approach. The Damerau-Levenshtein Distance algorithm calculates the edit distance by considering deletion, insertion, substitution, and transposition operations, making it very suitable for overcoming typos. Meanwhile, the Deep Neural Network approach is used to assess the suitability of candidate words based on the overall sentence context, where the candidate with the highest suitability will be selected as the correction result. The system successfully achieved an accuracy rate of 98.56 % and a word correction success rate of 94.17 %. The results of this study prove that the combination of the Damerau-Levenshtein Distance algorithm and the Deep Neural Network approach has a good ability to detect and correct word errors, especially real-word errors.

Keywords: *Real-word errors, Damerau-Levenshtein Distance, Deep Neural Network, Natural Language Processing, spelling correction.*

DAFTAR ISI

PERSETUJUAN	Error!
Bookmark not defined.	
PERNYATAAN	iii
UCAPAN TERIMA KASIH	v
ABSTRAK	viii
ABSTRACT	ix
DAFTAR ISI	x
DAFTAR TABEL	xii
DAFTAR GAMBAR	xiii
DAFTAR PSEUDOCODE	xiv
BAB 1 PENDAHULUAN	15
1.1 Latar Belakang	15
1.2 Rumusan Masalah	16
1.3 Tujuan Penelitian	17
1.4 Batasan Masalah Penelitian	17
1.5 Manfaat Penelitian	17
1.6 Metode Penelitian	18
1.7 Sistematika Penulisan	18
BAB 2 LANDASAN TEORI	20
2.1 <i>Spell Correction</i>	20
2.1.1 Jenis Kesalahan Eja	20
2.2 <i>Parallel Corpus</i>	21
2.3 Algoritma Damerau Levenshtein Distance	22
2.4 <i>Deep Neural Network</i>	25
2.4.1 <i>Long Short-Term Memory</i>	26
2.4.2 <i>Softmax</i>	27
2.5 <i>Web Scraping</i>	28
2.6 Penelitian terdahulu	28
BAB 3 ANALISIS DAN PERANCANGAN SISTEM	34
3.1 Pengumpulan Data	34
3.1.1 <i>Web Scraping</i>	35
3.1.2 <i>Parallel Corpus</i>	37
3.2 <i>Pre-processing</i>	39
3.2.1 <i>Record Punctuation</i>	40

3.2.2 <i>Data Cleaning</i>	41
3.2.3 <i>Tokenization</i>	42
3.2.4 <i>Word to Integer</i>	43
3.2.5 <i>Labelize</i>	44
3.2.6 <i>Untokenize</i>	45
3.2.7 <i>Dataset</i>	46
3.3 Pembentukan sistem	46
3.3.1 Koreksi kata <i>non-word errors</i> oleh <i>Damerau Levenshtein Distance</i>	46
3.3.2 <i>Deep Neural Network</i>	48
3.3.3 <i>Train dan modelling data</i>	50
3.3.4 Koreksi jarak bobot oleh <i>Damerau-Levenshtein Distance</i>	52
3.3.5 <i>Restored Punctuation</i>	53
3.4 Perancangan Alur Sistem	53
3.4.1 Desain Halaman Antarmuka Aplikasi	56
3.4.2 Desain Halaman <i>View Details</i>	57
3.5 Metode Evaluasi	58
BAB 4 IMPLEMENTASI DAN PENGUJIAN	60
4.1 Implementasi Sistem	60
4.1.1 Perangkat Keras dan Perangkat Lunak	60
4.1.2 Implementasi Perancangan Antarmuka	61
4.2 Pengujian Sistem	62
4.3 Implementasi Model	64
4.3.1 <i>Parallel Corpus</i>	64
4.3.2 <i>Pre-processing</i>	65
4.3.3 <i>Deep Neural Network</i>	69
4.3.4 <i>Train dan Modelling</i>	69
4.3.5 Evaluasi	71
BAB V KESIMPULAN DAN SARAN	75
5.1 Kesimpulan	75
5.2 Saran	75
DAFTAR PUSTAKA	77

DAFTAR TABEL

Tabel 2. 1 Contoh <i>non-word errors</i>	21
Tabel 2. 2 Contoh <i>real-word errors</i>	21
Tabel 2. 3 Pasangan kalimat paralel benar-salah	22
Tabel 2. 4 Ilustrasi operasi <i>Damerau-Levenshtein Distance</i>	22
Tabel 2. 5 Contoh <i>Damerau Levenhstein-Distance</i>	24
Tabel 2. 6 Penelitian Terdahulu	29
Tabel 3. 1 Kalimat Hasil <i>Web Scraping</i>	36
Tabel 3. 2 Contoh <i>record punctuation</i>	41
Tabel 3. 3 <i>Data Cleaning</i>	42
Tabel 3. 4 Tahap <i>Tokenize</i>	43
Tabel 3. 5 Contoh <i>Word to Integer</i>	44
Tabel 3. 6 Contoh <i>Labelize</i>	45
Tabel 3. 7 Contoh Tahap <i>Untokenize</i>	45
Tabel 3. 8 Contoh koreksi kata <i>non-word errors</i>	48
Tabel 3. 9 Contoh kandidat kata oleh <i>Damerau-Lavenshtein Distance</i>	48
Tabel 3. 10 Contoh koreksi <i>real-word errors</i> oleh <i>learned model DNN</i>	52
Tabel 3. 11 Contoh koreksi jarak bobot oleh <i>Damerau-Lavenshtein Distance</i>	52
Tabel 3. 12 <i>Restored Punctuation</i>	53
Tabel 3. 13 Ilustrasi <i>Confusion Matrix</i>	59
Tabel 4. 1 Daftar Kalimat Pengujian Langsung	62
Tabel 4. 2 Kinerja model per <i>epoch</i>	70

DAFTAR GAMBAR

Gambar 2. 1 <i>Perceptron</i>	26
Gambar 3. 1 Arsitektur Umum	39
Gambar 3. 2 Diagram Alir Sistem	54
Gambar 3. 3 Halaman antarmuka aplikasi	56
Gambar 3. 4 Halaman <i>view details</i> aplikasi	58
Gambar 4. 1 Halaman <i>Spell Correction</i>	61
Gambar 4. 2 Tampilan setelah proses koreksi	62
Gambar 4. 3 Pasangan Kalimat Benar dan Salah	65
Gambar 4. 4 Grafik <i>loss</i> untuk <i>train</i> dan <i>valid</i> setiap <i>epoch</i>	71
Gambar 4. 5 Hasil <i>Confusion Matrix</i>	72



DAFTAR PSEUDOCODE

Pseudocode 3. 1 <i>Web Scraping</i>	36
Pseudocode 3. 2 Mencatat tanda baca	40
Pseudocode 3. 3 Tahap <i>Data Cleaning</i>	42
Pseudocode 3. 4 Tahap <i>Tokenize</i>	43
Pseudocode 3. 5 Tahap <i>Word to Integer</i>	43
Pseudocode 3. 6 Tahap <i>Labelize</i>	44
Pseudocode 3. 7 Tahap <i>untokenize</i>	45
Pseudocode 3. 8 Tahap Koreksi Kata <i>Non-word Errors</i>	47
Pseudocode 3. 9 Tahap <i>Deep Neural Network</i>	49
Pseudocode 3. 10 Tahap <i>train dan modelling data</i>	51



BAB 1

PENDAHULUAN

1.1 Latar Belakang

Kehidupan manusia bergantung pada komunikasi, baik dalam interaksi personal maupun profesional. Menulis merupakan salah satu cara berkomunikasi yang efektif untuk mengekspresikan pikiran, dan menyebarkan informasi. Dalam komunikasi, penting untuk terhindar dari kesalahan ejaan dan tata bahasa agar pesan tersampaikan dengan jelas dan mudah dipahami (Aziz et al., 2023). Untuk mencapai tujuan dan membangun hubungan yang positif, kemampuan berkomunikasi secara efektif diperlukan, baik secara lisan maupun tertulis. Kualitas tulisan, yang mencakup ketepatan ejaan dan tata bahasa, sangat penting untuk komunikasi tertulis. Kesalahan ejaan, baik *non-words* (kata yang tidak ada dalam kamus) maupun *real-words* (kata yang ada dalam kamus tetapi digunakan secara tidak tepat), dapat mempersulit pemahaman. Hal ini dapat terjadi dalam berbagai situasi, seperti komunikasi bisnis, dokumen resmi, dan media sosial.

Dengan perkembangan teknologi yang telah memfasilitasi pekerjaan sehari-hari manusia, termasuk dalam bidang bahasa dan komunikasi. Aktivitas menulis, sebagai contoh, mengalami perubahan signifikan dengan bantuan teknologi, meskipun masih dihadapkan pada kesalahan pengetikan. Untuk mengatasi masalah ini, muncul kebutuhan akan penggunaan algoritma dalam bentuk aplikasi untuk memeriksa kesalahan penulisan kata (Fu'adi et al., 2015).

Dengan pemanfaatan dari NLP (*Natural Language Processing*), mengidentifikasi kata yang salah dapat dicapai dengan menggunakan proses tersebut dengan mempertimbangkan arti kata dan keberadaan kata dalam kamus, penelitian sebelumnya dapat menemukan kata yang tidak tepat. Namun, ini masih

menyebabkan masalah, beberapa kata memiliki arti dan ada dalam kamus, tetapi tidak sesuai dengan konteks kalimat. Penelitian terdahulu terkait *non-word errors* lebih banyak dibandingkan penelitian terkait *real-word errors* terutama untuk bahasa Indonesia, seperti penelitian dengan judul “*Deep Learning-Based Context-Sensitive Spelling Typing Error Correction*” yang dilakukan oleh Lee et al. (2020) yang telah meneliti terkait perbaikan *real-word errors*, namun masih untuk bahasa asing. Juga seperti penelitian yang dilakukan oleh Hamidah (2019) dengan judul “*Spelling Checker menggunakan Algoritma Damerau Levenshtein Distance dan Cosine Similarity*” yang telah meneliti terkait *real-word errors* tapi hanya berfokus pada pengecekan kata yang salah ejaan dan memberikan saran kandidat kata, tidak sampai perbaikan. Penelitian *non-word errors* bahasa Indonesia lainnya yaitu penelitian berjudul “*Damerau levenshtein distance for indonesian spelling correction*” yang dilakukan oleh Santoso et al. (2019).

Algoritma *Damerau Levenshtein Distance* mempertimbangkan jumlah minimum operasi edit yang diperlukan untuk mengubah satu string menjadi string lainnya dengan cara mengukur jarak edit dengan mempertimbangkan penghapusan, penyisipan, substitusi, dan transposisi, sehingga cocok untuk menangani kesalahan ketik. *Deep Neural Network* merupakan pendekatan berbasis jaringan saraf tiruan yang memiliki dua atau lebih lapisan tersembunyi dengan tujuan memodelkan data yang kompleks secara akurat (Wu et al., 2016). Tujuan dari pendekatan *Deep Neural Network* adalah untuk mencari perbandingan kecocokan dari masing-masing kandidat kata terhadap kesatuan kalimat. Kata yang memiliki tingkat kecocokan tertinggi akan dianggap sebagai kata yang tepat.

Dari latar belakang yang sudah dijelaskan, dilakukan penelitian dengan judul “Implementasi Algoritma *Damerau-Levenshtein Distance* dan Pendekatan *Deep Neural Network* Dalam Analisis dan Perbaikan Kesalahan Eja atau Penulisan *Real-word Error* Bahasa Indonesia”.

1.2 Rumusan Masalah

Para peneliti sebelumnya telah melakukan banyak penelitian tentang *spell correction* hingga saat ini. Meskipun demikian, sebagian besar penelitian hanya fokus pada kesalahan eja *non-word errors*. Identifikasi *real-word errors* lebih sulit

karena membutuhkan pendekatan yang lebih komprehensif daripada hanya melihat daftar kata dalam kamus bahasa yang relevan. Selain itu, tidak banyak aplikasi yang dapat memperbaiki kosa kata untuk teks bahasa Indonesia. Sehingga diperlukan suatu sistem yang dapat mengidentifikasi serta memperbaiki *real-word errors* dalam bahasa Indonesia.

1.3 Tujuan Penelitian

Penelitian ini dilakukan dengan tujuan mendeteksi dan memperbaiki *real-word errors* dalam teks bahasa Indonesia, mengatasi kesalahan ejaan dan penulisan dalam teks yang mengurangi makna asli atau kelengkapan suatu informasi dengan menggunakan algoritma *Damerau-Levenshtein Distance* dan pendekatan *Deep Neural Network*.

1.4 Batasan Masalah Penelitian

Agar penelitian ini tidak menyimpang dari tujuan yang telah ditetapkan, maka terdapat batasan-batasan dalam penelitian ini, yaitu sebagai berikut:

1. *Input file* fokus pada artikel atau tulisan terkait dengan ilmu komputer dan teknologi informasi oleh *user*
2. Artikel atau tulisan yang diobservasi hanya artikel dan tulisan yang berbahasa Indonesia
3. Sumber data teks untuk pelatihan dan evaluasi model berasal dari repositori Universitas Sumatera Utara dengan fokus konteks pada teknologi.
4. *Input file* sistem berformat *text* (.txt), dan *Portable Document Format* (.pdf).

1.5 Manfaat Penelitian

Manfaat yang diperoleh dari penelitian ini adalah:

1. Memberikan informasi mengenai penerapan algoritma *Damerau-Levenshtein Distance* dan pendekatan *Deep Neural Network* untuk memprediksi kata yang tepat dari kata yang salah eja (*real-word errors*) atau tulis berdasarkan kecocokan dalam sebuah kalimat;
2. Bisa dimanfaatkan sebagai acuan untuk penelitian lebih lanjut mengenai koreksi ejaan maupun *Natural Language Processing* (NLP).

1.6 Metode Penelitian

Dalam menjalankan penelitian, terdapat tahapan-tahapan yang dilakukan sebagai berikut :

1. Studi Literatur

Di dalam tahap ini, penulis mencari dan mengumpulkan informasi melalui publikasi penelitian hingga pakar ahli serta melakukan peninjauan pustaka melalui buku-buku, jurnal, e-book, artikel ilmiah, makalah ataupun situs internet terpercaya yang berhubungan dengan algoritma *Damerau-Levenshtein Distance* dan metode *Deep Neural Network*.

2. Analisis Permasalahan

Pada tahap ini dilakukan analisis terhadap informasi yang telah dikumpulkan dan berdasarkan ruang lingkup penelitian, agar mendapat pemahaman terkait kontep penggunaan algoritma *Damerau-Levenshtein Distance* dan metode *Deep Neural Network* yang akan diterapkan dalam penelitian untuk melakukan analisis dan perbaikan kesalahan eja atau penulisan *real-word error* bahasa Indonesia.

3. Perancangan Sistem

Tahap berikutnya adalah melakukan perancangan sistem untuk penelitiann yang akan dilakukan berupa perancangan arsitektur umum, penentuan training dan testing data.

4. Penyusunan Laporan

Pada tahap ini peneliti akan melakukan penyusunan laporan serta dokumentasi dari hasil penelitian yang dilakukan.

1.7 Sistematika Penulisan

Sistem penulisan skripsi ini terdiri dari beberapa bagian yaitu:

BAB 1: PENDAHULUAN

Pada bab ini berisikan latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, serta metodologi dan sistematika penulisan.

BAB 2: LANDASAN TEORI

Pada bab ini berisikan teori yang berhubungan dengan jenis kesalahan eja, *spell correction*, algoritma *damerau levenshtein distance*, *deep neutral network* dan korpus.

BAB 3: ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini memuat penjelasan mengenai arsitektur umum yang digunakan pada penelitian ini, dan analisis terhadap masalah yang diteliti.

BAB 4: IMPLEMENTASI DAN PENGUJIAN

Pada bab ini meliputi tahapan proses implementasi dan pengujian dari hasil rancangan sistem yang telah dijelaskan pada bab sebelumnya agar dapat memperlihatkan hasil dari penelitian ini.

BAB 5: KESIMPULAN DAN SARAN

Pada bab ini berisikan kesimpulan dari hasil penelitian yang telah dilakukan berdasarkan paparan dalam setiap bab, dan juga memberikan saran saran yang diberikan oleh peneliti sebagai masukan untuk penelitian selanjutnya.



BAB 2

LANDASAN TEORI

2.1 *Spell Correction*

Spell correction merupakan proses koreksi ejaan yang digunakan untuk mendeteksi dan mengoreksi kesalahan ortografis dalam teks (Ahmadzade et al., 2021), hal ini penting untuk meningkatkan kualitas dan keterbacaan teks, agar memastikan pesan yang disampaikan jelas dan akurat. Bentuk umum dari kesalahan penulisan ini meliputi penyisipan, penghapusan, substitusi, dan transposisi dari kata aslinya. Kesalahan ejaan dalam teks bisa terjadi karena beberapa faktor, termasuk kelalaian, letak huruf yang berdekatan pada *keyboard*, hingga ketidaktahuan penulis mengenai ejaan yang benar.

2.1.1 *Jenis Kesalahan Eja*

Kesalahan eja secara umum terbagi menjadi dua bentuk, yaitu *non-word errors* dan *real-word errors*.

2.1.1.1 *Non-word Errors*

Non-word errors merupakan kesalahan pada suatu kata yang di mana kata tersebut sebenarnya tidak ada dalam kamus (Ratnasari et al., 2017). *Non-word errors* dianggap kurang tepat karena kata tersebut tidak memiliki makna.

Tabel 2. 1 Contoh *non-word errors*

Sumber	Kalimat mengandung <i>non-word errors</i>
Jurnal Pendidikan dan Kebudayaan, Volume 25, Nomor 3, 2020	Analisis Efektip dalam Proses Pembelajaran.

Kesalahan penulisan eja pada kata “efektip” yang seharusnya “efektif”.

2.1.1.2 Real-word Errors

Real-Word Errors merupakan kesalahan pada suatu kata yang di mana kata tersebut benar dan bermakna tetapi kata tersebut bukan kata yang dimaksud dalam suatu konteks kalimat (Samanta et al., 2013).

Tabel 2. 2 Contoh *real-word errors*

Sumber	Kalimat mengandung <i>real-word errors</i>
Detik, Artikel tanggal 15 Maret 2021	Pelang Lalu Lintas.

Kesalahan penulisan eja pada kata “pelang” yang seharusnya “pelanggaran”.

2.2 Parallel Corpus

Parallel corpus merupakan kumpulan teks berbahasa yang disusun paralel berpasangan-pasangan secara paralel dalam dua bahasa atau lebih, yang dimana masing masing teks memiliki terjemahan persisnya dalam bahasa lain. Pasangan paralel dari berbagai istilah-istilah ini seperti teks, kata-kata, frasa, maupun entitas linguistik lainnya yang disatukan secara sistematis untuk diproses lebih lanjut.

Parallel corpus pada koreksi ejaan merupakan kumpulan teks yang berisi kalimat-kalimat yang mirip, di mana satu kalimat memiliki ejaan yang benar dan kalimat lainnya memiliki ejaan yang salah. Penggunaan *parallel corpus* dalam penelitian koreksi ejaan bertujuan untuk melatih sistem agar mampu mengenali dan memperbaiki kesalahan ejaan dalam teks. Dengan melatih model pada berbagai jenis teks dan topik yang terdapat dalam *parallel corpus*, sistem koreksi ejaan dapat menghasilkan koreksi yang akurat dan relevan dalam berbagai konteks, termasuk perubahan huruf, penghilangan, penambahan, substitusi huruf yang mirip. Bentuk

pasangan kalimat paralel benar-salah diilustrasikan pada tabel 2.3.

Tabel 2. 3 Pasangan kalimat paralel benar-salah

Kalimat benar	Kalimat salah
Dia sering menghadiri seminar ilmiah di kampusnya.	Dia sering menghadiri seminar ilmiah di kampusya.
Guru itu memberikan tugas yang menantang kepada siswa-siswinya.	Guru itu memberikan tugas yang mendatang kepada siswa-siswinya.
Pada liburan akhir tahun, kami berencana untuk berkunjung ke desa.	Pada liburan akhir tahun, kami bencana untuk berkunjung ke desa.

2.3 Algoritma *Damerau Levenshtein Distance*

Algoritma *Damerau-Levenshtein Distance* merupakan pengembangan dari Algoritma *Levenshtein* yang digunakan untuk mencari jumlah operasi *string* yang paling sedikit untuk mentransformasikan suatu *string* menjadi *string* yang lain dan menawarkan akurasi yang lebih baik. Algoritma ini digunakan dalam pencarian *string* dengan pendekatan perkiraan (*Approximate String Matching*) (Santoso et al., 2019).

Operasi yang dihitung pada algoritma ini antara lain penyisipan (*insert*), penghapusan (*delete*), substitusi atau penggantian karakter (*substitute*), dan menukar atau transposisi (*transposition*). Sebagai contoh, ilustrasi dari masing-masing operasi dalam menghitung jarak perubahan kata dengan algoritma *Damerau-Levenshtein Distance* dapat dilihat pada Tabel 2.4.

Tabel 2. 4 Ilustrasi operasi *Damerau-Levenshtein Distance*

	Sebelum	Sesudah
Penyisipan	akan	makan
Penghapusan	makan i	makan
Substitusi	makin	makan
Transposisi	mak n a	makan

Perhitungan jarak perubahan kata antara satu untaian kata dengan untaian yang lain dihitung dengan fungsi rekursif seperti berikut:

$$a = a_1 \dots a_i \dots a_n$$

$$b = b_1 \dots b_j \dots b_m$$

$$d_{a,b}(i, j) = \min \begin{cases} 0 & \text{if } i = j = 0 \\ d_{a,b}(i-1, j) + 1 & \text{if } i > 0 \\ d_{a,b}(i, j-1) + 1 & \text{if } j > 0 \\ d_{a,b}(i-1, j-1) & \text{if } i, j > 0 \text{ and } a_i = b_j \\ d_{a,b}(i-1, j-1) + 1 & \text{if } i, j > 0 \text{ and } a_i \neq b_j \\ d_{a,b}(i-2, j-2) + 1 & \text{if } i, j > 0 \text{ and } a_i = b_{j-1} \text{ and } a_{i-1} = b_j \end{cases}$$

Lalu setiap panggilan rekursif mencakup empat operasi utama seperti :

$$1. \text{ Penghapusan (Deletion)} : d_{a,b}(i-1, j) + 1 \quad (2.1)$$

Menghapus satu karakter dari string a untuk mencocokkan dengan b.

$$2. \text{ Penyisipan (Insertion)} : d_{a,b}(i, j-1) + 1 \quad (2.2)$$

Menyisipkan satu karakter ke dalam string a agar cocok dengan b.

$$3. \text{ Penggantian/substitusi (substitution)} : d_{a,b}(i-1, j-1) + 1_{(a_i \neq b_j)} \quad (2.3)$$

Substitusi terjadi jika karakter a dan b berbeda.

$$4. \text{ Transposisi (transposition)} : d_{a,b}(i-2, j-2) + 1_{(a_i \neq b_j)} \quad (2.4)$$

Jika dua karakter berturut-turut bertukar tempat.

Algoritma *Damerau-Levenshtein Distance* merupakan pengembangan lebih lanjut dari algoritma *Levenshtein Distance* dan *Weighted Levenshtein Distance*. Algoritma ini tidak hanya menghitung operasi penyuntingan seperti penghapusan, penyisipan, substitusi, dan transposisi, tetapi juga mempertimbangkan bobot dari setiap operasi tersebut. Bobot operasi dapat berbeda tergantung pada konteksnya. Faktor-faktor yang memengaruhi bobot ini meliputi jarak antar huruf pada papan ketik, kemiripan fonetis atau visual antar karakter, serta pola kesalahan umum dalam pengetikan.

Secara ilmiah, pendekatan berbasis bobot ini dirancang untuk meningkatkan akurasi dalam mengidentifikasi kesalahan ejaan yang lebih relevan secara kontekstual, terutama dalam aplikasi pengolahan bahasa alami (*Natural Language Processing*) seperti koreksi ejaan otomatis dan pemrosesan teks multibahasa.

Algoritma *Damerau Levenshtein Distance* terdiri dari beberapa langkah proses perhitungan sebagai berikut:

1. Menentukan sebuah matriks jarak, di mana jumlah baris dan kolomnya sama dengan panjang string yang diberikan.
2. Menambah satu baris dan satu kolom ekstra yang akan memuat indeks karakter untuk perhitungan.
3. Menghitung nilai fungsi $d_{a,b}(i,j)$, yang bergantung pada baris i dan kolom j , menggunakan formula yang telah ditentukan.
4. Pilih nilai minimum dari semua kemungkinan nilai $d_{a,b}(i,j)$.
5. Isi sel pada perpotongan kolom dan baris dengan nilai minimum yang ditemukan.
6. Setelah seluruh matriks terisi, skor pada sel di sudut kanan bawah akan menjadi hasilnya.

Contoh hasil akhir matriks algoritma *Damerau Levenshtein Distance* yang mengukur jarak kata "malam" dengan kata "makan" dengan menetapkan bobot setiap operasi sama dengan satu (1) dapat dilihat pada tabel 2.5.

Tabel 2. 5 Contoh Damerau Levenhstein-Distance

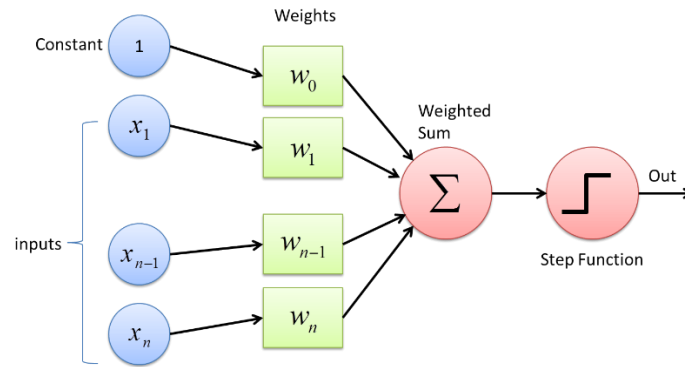
		m	a	k	a	n
	0	1	2	3	4	5
m	1	0	1	2	3	4
a	2	1	0	1	2	3
l	3	2	1	1	2	3
a	4	3	2	2	1	2
m	5	4	3	3	2	2

Jarak kata yang diukur menggunakan algoritma *Damerau Levenshtein Distance* dapat dilihat pada nilai pada sel akhir matriks. Tabel 2.5 menunjukkan bahwa jarak yang ada di antara kata "malam" dengan kata "makan" adalah sebanyak 2 bobot operasi.

2.4 Deep Neural Network

Deteksi dan koreksi kesalahan kata yang sebenarnya (*real-word errors*) lebih sulit dibandingkan dengan kesalahan kata yang bukan kata (*non-word errors*). Hal ini disebabkan oleh pengaruh hubungan kata yang salah dengan konteks kata terdekat dan kalimatnya (Lee et al., 2020). Oleh karena itu, diperlukan metode yang dapat memahami konteks dari suatu kalimat. Salah satu metode tersebut adalah *Deep Neural Network*. *Deep Neural Network* adalah salah satu aplikasi dari bidang ilmu yang berhubungan dengan interaksi antara komputer dan bahasa manusia. Bidang ilmu ini mempelajari dan fokus pada pengembangan teknologi yang dapat memahami bahasa alami manusia baik secara lisan, tulisan, maupun isyarat (Fu'adi, 2015; Rosyadi et al., 2020). Berdasarkan hasil pemrosesan, akan dihasilkan respons yang sesuai dengan konteksnya. Selain *Deep Neural Network*, bidang ilmu ini juga dapat digunakan dalam analisis sentimen, terjemahan mesin, klasifikasi teks, dan penjawab pertanyaan. Bidang ilmu ini dikenal dengan istilah *Natural Language Processing* atau pemrosesan bahasa alami.

Deep Neural Network juga merupakan jenis khusus dari *Artificial Neural Network*. *Artificial Neural Network* atau jaringan saraf tiruan adalah salah satu model dari *machine learning* yang didasarkan pada sistem saraf biologis manusia. Dengan meniru cara kerja otak manusia, jaringan saraf tiruan memiliki kemampuan untuk mempelajari pola-pola kompleks sehingga dapat melakukan tugas seperti pengenalan pola, klasifikasi, prediksi, dan pengambilan keputusan. Salah satu komponen dasar dalam membangun *Deep Neural Network* adalah *perceptron*. *Perceptron* dan *neuron* mengacu pada konsep yang serupa, di mana *perceptron* dirancang untuk meniru secara matematis sifat dasar neuron biologis. *Perceptron* menerima masukan dalam bentuk numerik, memproses informasi berdasarkan masukan tersebut, dan menghasilkan keluaran.



Gambar 2. 1 *Perceptron (medium.com)*

Sebuah *perceptron* terdiri dari 5 komponen yaitu:

1. Masukan (x_i);
2. Bobot (w_i) dan bias (w_0);
3. Penjumlahan (Σ);
4. Fungsi aktivasi (f);
5. Keluaran (y).

Perceptron beroperasi dengan menerima *input* berupa data numerik. Setiap *input* memiliki bobot, yang menggambarkan sejauh mana kontribusi *input* terhadap *output*. Setiap input dikalikan dengan bobot yang sesuai, lalu hasilnya dijumlahkan dengan bias. Fungsi aktivasi, yang bersifat *non-linear*, akan memproses hasil penjumlahan tersebut. Fungsi aktivasi berfungsi untuk mengubah nilai yang diperoleh menjadi nilai yang relevan dan sesuai. *Output* yang dihasilkan merupakan hasil perhitungan *perceptron* dalam bentuk data numerik.

2.4.1 Long Short-Term Memory

Long Short-Term Memory (LSTM) adalah bagian dari *Recurrent Neural Network (RNN)* yang sering digunakan dalam tugas pemrosesan bahasa alami (Qori et al., 2022). LSTM dirancang khusus untuk menangani masalah yang melibatkan data berurutan atau data temporal. *Long Short-Term Memory (LSTM)* dikembangkan untuk mengatasi tantangan yang dihadapi oleh *Recurrent Neural Network (RNN)* dalam mengingat informasi dalam jangka panjang dan untuk mengatasi masalah hilangnya gradien saat melatih model.

Dalam konteks koreksi ejaan, *LSTM* dapat digunakan untuk mendeteksi dan memperbaiki kesalahan ejaan dalam teks. Proses koreksi ejaan menggunakan *LSTM* melibatkan pelatihan model dengan data yang berisi kata-kata yang dieja dengan benar dan yang salah eja (Jayanthi et al., 2020). Model ini belajar untuk memprediksi ejaan yang benar berdasarkan bentuk kata yang salah eja. Beberapa model juga memanfaatkan informasi kontekstual untuk meningkatkan akurasi koreksi.

2.4.2 *Softmax*

Fungsi aktivasi *Softmax* merupakan salah satu fungsi yang banyak diterapkan dalam jaringan saraf, terutama pada tugas klasifikasi yang melibatkan lebih dari dua kelas (klasifikasi multi-kelas). Fungsi ini mengubah output model menjadi probabilitas yang terdistribusi secara proporsional, dengan memastikan jumlah total outputnya selalu 1.0. Secara matematis, *Softmax* mengubah setiap nilai input menjadi nilai dalam rentang $[0, 1]$, yang dapat dipahami sebagai probabilitas untuk masing-masing kelas. Kelas dengan probabilitas tertinggi umumnya dipilih sebagai hasil prediksi model.

Softmax sangat berguna dalam aplikasi klasifikasi karena menghasilkan distribusi probabilitas yang jelas untuk setiap kelas. Dengan demikian, model dapat menentukan kelas yang paling mungkin menjadi jawaban yang benar. Fungsi ini juga mempermudah penggunaan fungsi *loss* seperti *cross-entropy* untuk menghitung kesalahan model. *Cross-entropy* bekerja secara efektif jika digabungkan dengan *Softmax*, karena kedua fungsi ini saling mendukung untuk mengoptimalkan prediksi kelas yang benar dengan meminimalkan kesalahan klasifikasi selama pelatihan.

Penelitian oleh Goodfellow et al. (2016) dan Bengio et al. (2006) menunjukkan bahwa penerapan *Softmax* dalam jaringan saraf multi-kelas membantu menghubungkan output model dengan probabilitas kelas yang lebih mudah dipahami. Penelitian ini juga mengungkapkan bahwa *Softmax* mendukung proses pembelajaran yang lebih stabil dan konvergensi yang lebih cepat jika dilatih bersama dengan fungsi *loss* yang sesuai. Oleh karena itu, *Softmax* menjadi elemen

penting dalam berbagai aplikasi pembelajaran mesin, seperti klasifikasi gambar dan teks.

Secara keseluruhan, *Softmax* memainkan peran penting dalam proses pengambilan keputusan klasifikasi berbasis probabilitas, yang memungkinkan model memilih kelas dengan tingkat kepercayaan tertinggi. Penggunaannya yang luas dalam klasifikasi multi-kelas, bersama dengan optimisasi menggunakan *cross-entropy*, menjadikannya alat yang sangat efektif untuk meningkatkan akurasi dan efisiensi pelatihan model jaringan saraf.

2.5 Web Scraping

Web scraping adalah teknik otomatisasi untuk mengekstraksi data dari halaman web yang digunakan secara luas dalam konteks pengumpulan *Big Data*. Teknik ini melibatkan navigasi situs web, pengambilan informasi yang relevan dari elemen-elemen tertentu dalam dokumen *Hypertext Markup Language* (HTML), dan transformasi data yang diperoleh menjadi format yang terstruktur seperti *JavaScript Object Notation* (JSON) atau *Extensible Markup Language* (XML). Proses ini memungkinkan data yang tidak terstruktur atau semi-terstruktur pada situs web dikonversi menjadi format yang dapat dianalisis lebih lanjut (Naing *et al.*, 2024).

2.6 Penelitian terdahulu

Hingga saat ini, telah banyak dilakukan penelitian mengenai *Text processing* seperti *Text Classification*, *Sentiment Analysis*, hingga *Spell Correction*. Berbagai metode juga telah banyak digunakan untuk mendeteksi kesalahan masing masing penelitian. Tabel 2.6 memuat beberapa penelitian terdahulu yang membahas metode deteksi kesalahan eja atau biasa dikenal dengan *Spell Correction*.

Tabel 2. 6 Penelitian Terdahulu

No.	Penulis	Metode	Keterangan
1.	Herry Sujaini, <i>et. Al.</i> (2022)	<i>Jaro-Winkler Distance</i> dan <i>N-Gram</i>	Penelitian ini mendeteksi kesalahan ketik dalam sebuah dokumen, lalu memberikan saran untuk koreksi kata <i>real-word errors</i> dan <i>non-word errors</i> . Algoritma <i>N-Gram</i> digunakan sebagai seperti unigram, bigram, dan trigram untuk menganalisis data teks, lalu <i>Jaro-Winkler Distance</i> melakukan perhitungan kesamaan antara dua string. Hasil akurasi menemukan saran perbaikan kata sebesar 71,48% dan 98,449% sebagai tingkat keberhasilan memberikan saran perbaikan kata.
2.	Anggasta TA Kusuma, <i>et. Al.</i> (2023)	<i>Peter Norvig</i> , LSTM, dan <i>N-gram</i>	Penelitian ini mendeteksi membandingkan metode pemeriksaan ejaan dalam Bahasa Indonesia, seperti metode <i>Peter Norvig</i> , <i>Long Short-Term Memory</i> (LSTM), dan <i>N-gram</i> untuk mengukur akurasi dalam mengoreksi kesalahan ejaan, memanfaatkan data SPECIL (Spell Error Corpus for Indonesian Language) untuk pengujian, dengan akurasi 89% oleh <i>Peter</i>

Tabel 2. 6 Penelitian Terdahulu (Lanjutan)

No	Penulis	Metode	Keterangan
			<i>Norvig</i> , lalu 75% oleh metode <i>N-Gram</i> , dan LSTM dengan akurasi 74%.
3.	Puji Santoso, <i>et. Al.</i> (2019)	<i>Damerau Levenshtein Distance</i>	Penelitian ini mengukur perbandingan efektivitas algoritma jarak Levenshtein dan algoritma jarak Damerau-Levenshtein dalam mengoreksi kesalahan ejaan Indonesia, khususnya dalam dongeng anak-anak, dengan akurasi 75% untuk Algoritma <i>Damerau Levenshtein Distance</i> dibandingkan dengan Algoritma <i>Levenshtein Distance</i> dengan akurasi 73%.
4.	Jung-Hun Lee, <i>et. Al.</i> Fujiwara (2020)	<i>Large Language Models</i>	Penelitian ini bertujuan untuk memecahkan kesalahan ejaan yang peka konteks dalam dokumen bahasa Inggris, dengan fokus pada kesalahan ketik menggunakan <i>Deep Learning</i> , yang melibatkan informasi penyematan kata, informasi penyematan kontekstual, <i>auto-regressive (AR) language model</i> , dan <i>auto-encoding (AE) language model</i> . Akurasi dari penelitian ini mencapai lebih dari 96%

Tabel 2. 6 Penelitian Terdahulu (Lanjutan)

No	Penulis	Metode	Keterangan
			skor F1 dalam deteksi dan koreksi kesalahan.
5.	Ahmad Ahmadzade, et. Al. (2021)	<i>Levenshtein Distance dan Deep Neural Network</i>	Penelitian ini telah menggunakan algoritma Levenshtein dalam mendeteksi kesalahan dan pendekatan Deep Neural Network dalam menentukan probabilitas kata yang benar. Namun, penelitian ini masih ditujukan untuk real-word errors pada teks Bahasa Azerbaij. Model yang diusulkan untuk koreksi ejaan Azerbaijan menunjukkan peningkatan akurasi yang signifikan dibandingkan dengan model sebelumnya, mencapai 31 ingkat akurasi yang lebih tinggi di jarak edit yang berbeda, seperti 75% untuk jarak 0, 90% untuk jarak 1, dan 96% untuk jarak 2.
6.	Guimin Huang, et. Al. (2020)	<i>Levenshtein Distance, Double Metaphone Algorithm, dan Glove</i>	Penelitian ini menerapkan algoritma Levenshtein sebagai algoritma menghitung jarak kata untuk mendeteksi dan mengoreksi non-word errors pada teks Bahasa Inggris. Hasil akurasi dari metode yang diusulkan, dengan skor F1 84%.

Tabel 2. 6 Penelitian Terdahulu (Lanjutan)

No	Penulis	Metode	Keterangan
7.	Eureka Jeremy Aritomatika, <i>et.</i> <i>Al.</i> (2021)	<i>Jaro Winkler</i> <i>Distance</i>	Hasil dari penelitian ini pengujian aplikasi English Conversation dengan Algoritma Jaro Winkler Distance memperoleh hasil nilai perbandingan kemiripan kata dengan hasil nilai maksimal sebesar 100% serta nilai minimal sebesar 68.1%.

Penelitian ini memiliki sejumlah keunggulan dibandingkan dengan penelitian sebelumnya. Dengan mengintegrasikan algoritma *Damerau-Levenshtein Distance*, yang mampu menangani kesalahan transposisi, dan pendekatan *Deep Neural Network*, yang memahami konteks kalimat, penelitian ini menawarkan solusi yang lebih unggul dalam mendeteksi dan memperbaiki kesalahan ejaan, khususnya untuk *real-word errors* dalam Bahasa Indonesia. Berbeda dengan penelitian sebelumnya yang menggunakan metode seperti *Levenshtein Distance*, *Jaro-Winkler*, atau *Peter Norvig*, yang umumnya hanya fokus pada kesalahan *string* atau akurasi di tingkat kata, penelitian ini memastikan bahwa perbaikan yang dilakukan tetap menjaga makna asli kalimat.

Penelitian ini juga memberikan kontribusi dalam pengolahan Bahasa Indonesia, yang memiliki kompleksitas tersendiri dan sering kali tidak sepenuhnya teratasi oleh metode umum atau dataset seperti SPECIL (*Spell Error Corpus for Indonesian Language*). Pendekatan ini melampaui keterbatasan penelitian sebelumnya, seperti penelitian Ahmad Ahmadzade (2021) yang hanya berfokus pada Bahasa Azerbaijan, atau penelitian Puji Santoso (2019) yang sebatas mengukur efektivitas algoritma *Damerau-Levenshtein Distance* tanpa mempertimbangkan konteks. Oleh karena itu, penelitian ini mampu menghasilkan akurasi yang lebih baik dan solusi yang lebih sesuai untuk memperbaiki kesalahan ejaan dalam Bahasa Indonesia,

sekaligus menjadi terobosan signifikan dibandingkan penelitian-penelitian sebelumnya.



BAB 3

ANALISIS DAN PERANCANGAN SISTEM

Analisis penting dilakukan terhadap sistem yang akan dirancang untuk mendeteksi dan memperbaiki kesalahan ejaan dalam Bahasa Indonesia, terutama pada *real-word errors*. Tantangan utama yang dihadapi adalah kompleksitas struktur Bahasa Indonesia, seperti penggunaan imbuhan, pengulangan kata, dan konteks semantik, yang sering kali sulit diatasi oleh algoritma konvensional. Selain itu, metode sebelumnya, seperti algoritma *Levenshtein Distance* dan *Jaro-Winkler Distance*, memiliki kelemahan karena hanya berfokus pada tingkat kata tanpa memperhatikan konteks kalimat, sehingga sering menghasilkan koreksi yang kurang tepat dengan maksud kalimat.

Sebagai solusi, penelitian ini mengombinasikan algoritma *Damerau-Levenshtein Distance*, yang mampu menangani empat jenis kesalahan utama seperti penyisipan, penghapusan, penggantian, dan transposisi, dengan pendekatan *Deep Neural Network* yang memungkinkan pemahaman konteks kalimat secara lebih mendalam. Berbeda dari penelitian sebelumnya yang terbatas pada teks berbahasa Inggris atau mengandalkan dataset seperti SPECIL (*Spell Error Corpus for Indonesian Language*), penelitian ini berfokus pada pengolahan Bahasa Indonesia dengan menggunakan *parallel corpus* yang dirancang khusus. Dengan pendekatan ini, sistem mampu memberikan hasil koreksi yang lebih akurat dan relevan, tanpa mengubah arti pesan yang ingin disampaikan.

3.1 Pengumpulan Data

Pada tahap awal, dilakukan pengambilan dan pengumpulan data karya ilmiah yang berasal dari repositori Fakultas Ilmu Komputer dan Teknologi Informasi

Universitas Sumatera Utara yang mengandung unsur teknologi pada bidang ilmu komputer dengan teknik *crawling*. Setelah pengumpulan data dilakukan, maka data tersebut dimasukkan ke dalam *Parallel corpus* untuk melatih sistem *spell correction* dalam mengidentifikasi dan memperbaiki kesalahan ejaan dalam teks.

3.1.1 Web Scraping

Untuk mengumpulkan data berupa kalimat berbahasa Indonesia dengan variasi kata dalam konteks teknologi, dilakukan proses web scraping pada laman repositori Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara. *Web scraping* ini memanfaatkan *library BeautifulSoup* untuk mengekstrak teks abstrak dari karya ilmiah. Sebanyak 4000 kalimat bahasa Indonesia berhasil dikumpulkan melalui pengiriman permintaan ke server melalui link repositori USU untuk mendapatkan daftar karya ilmiah dari fakultas terkait. Selanjutnya, permintaan tambahan dilakukan ke server untuk setiap URL individu dari karya ilmiah tersebut. Proses *web scraping* ditunjukkan pada Pseudocode 3.1.

```

DEFINE baseUrl AS 'http://repositori.usu.ac.id'

FUNCTION ScrapAbstract(url):
    INITIALIZE abstract AS a dictionary WITH keys 'en' and 'id',
    both set to empty strings
    SEND an HTTP GET request TO (baseUrl + url)
    PARSE the response HTML USING BeautifulSoup
    FIND all <meta> tags WITH the attribute name='DCTERMS.abstract'

    FOR EACH meta tag found:
        EXTRACT the content AND replace line breaks WITH a space
        IF the content IS not in English:
            SET abstract['id'] TO the content
        ELSE:
            SET abstract['en'] TO the content

    RETURN the abstract dictionary

FUNCTION ScrapAbstracts(facultyCode, currentPage = 1):
    INITIALIZE abstracts AS a dictionary WITH keys 'en' and 'id',
    both set to empty lists
    GENERATE the URL TO scrape USING facultyCode
    SET itemsPerPage TO 20
    CALCULATE the offset BASED ON currentPage
    SEND an HTTP GET request TO the generated URL WITH query
    parameters

    IF the response status code IS NOT 200:

```

RETURN a dictionary WITH the status code AND an empty data field

PARSE the response HTML USING BeautifulSoup
 FIND the container FOR recent submissions USING its ID
 EXTRACT all <h4> elements REPRESENTING articles

INITIALIZE articleUrls AS an empty list
 FOR EACH article:
 FIND the <a> tag INSIDE the article AND extract its href attribute
 APPEND the URL TO articleUrls

USE asyncio TO concurrently gather abstract data FOR all URLs in articleUrls

STORE English abstracts IN abstracts['en']
 STORE Indonesian abstracts IN abstracts['id']

INITIALIZE meta AS an empty dictionary
 CHECK IF a "next page" link EXISTS:
 IF it exists:
 SET meta['hasNextPage'] TO True
 SET meta['nextUrl'] TO the href OF the "next page" link

RETURN a dictionary CONTAINING:
 - the response status code
 - the abstracts dictionary
 - the meta information

Pseudocode 3.1 Web Scraping

Data kalimat yang diperoleh kemudian disimpan dalam format .csv untuk memudahkan pengolahan dan analisis lebih lanjut.

Tabel 3.1 Kalimat Hasil *Web Scraping*

Nomor	Kalimat
1	Kerahasiaan data adalah hal yang penting dalam keamanan data.
2	Guru itu memberikan tugas yang mendatang kepada siswa-siswinya.
3	Pada liburan akhir tahun, kami bencana untuk berkunjung ke desa.
4	Namun bagi para pengguna mereka tidak memikirkan seberapa sulit algoritmanya, yang penting data mereka aman.
5	Dalam penelitian ini penulis menerapkan proses text mining dan proses n-gram karakter untuk seleksi fitur serta menggunakan algoritma Naive Bayes Classifier untuk mengklasifikasi sentimen secara otomatis.
.	.

.	.
3998	Pada penelitian ini dilakukan pemilihan piksel-piksel pada citra yang sesuai dengan nilai data penyisip sebagai tempat penyisipan dengan algoritma Genetika (GA).
3999	Penyisipan data pada piksel yang terpilih dengan algoritma GA dilakukan dengan menggunakan algoritma Modified Least Significant Bit (MLSB) pada posisi bit least significant bits (LSB).
4000	Perubahan bit-bit pada posisi LSB tidak memiliki pengaruh yang besar terhadap citra yang disisipkan sehingga dapat diperoleh nilai MSE yang tidak terlalu besar.

Kemudian data disimpan pada format *Comma Separated Values* (.csv) untuk diolah pada tahap berikutnya.

3.1.2 Parallel Corpus

Model *Deep Neural Network* dapat dilatih secara efektif untuk mengenali dan memperbaiki kesalahan ejaan dalam teks melalui pembelajaran model melalui pembentukan *parallel corpus*. Berdasarkan data kalimat yang diperoleh dari abstrak karya ilmiah di repositori Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara, *parallel corpus* ini dibuat dalam format pasangan kalimat dengan ejaan benar dan salah.

Algoritma *Damerau Levenshtein Distance* juga menggunakan *parallel corpus* untuk menghasilkan kandidat koreksi kata yang mungkin, dengan membandingkan jarak edit antara kata masukan dengan kata dalam kosakata dari *parallel corpus*.

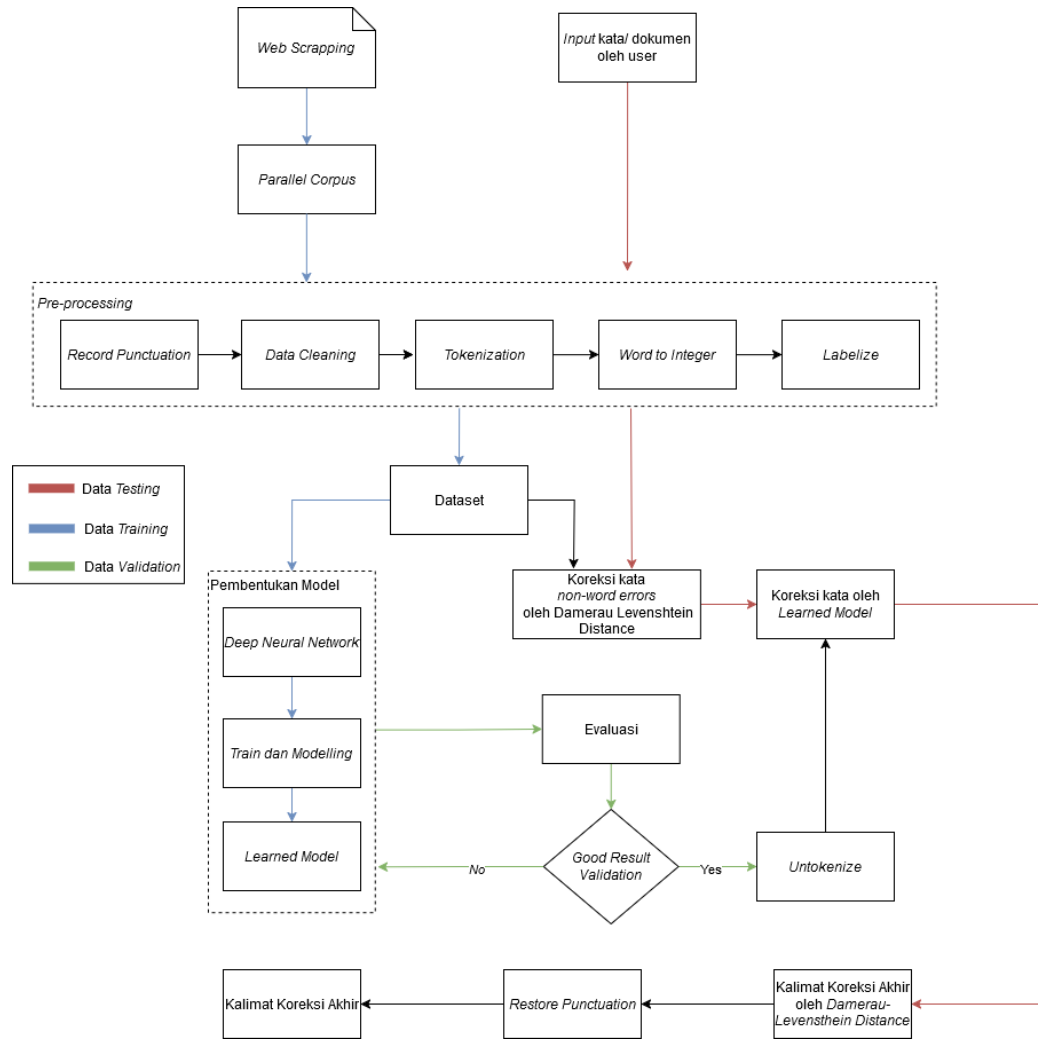
3.2 Arsitektur Umum

Penelitian ini diawali dengan proses pengumpulan data menggunakan teknik *web scraping*, yang menghasilkan *parallel corpus* sebagai sumber data utama. Data yang berhasil dihimpun kemudian diproses melalui tahap preprocessing, yang mencakup beberapa langkah penting, yaitu *data cleaning* untuk menghilangkan

karakter yang tidak relevan, dan untuk mengubah semua huruf menjadi huruf kecil demi keseragaman, *tokenization* untuk memisahkan teks menjadi kata-kata individual, serta *word to integer* yang mengonversi kata-kata menjadi representasi numerik. Setelah tahap *pre-processing* selesai, data dibagi ke dalam tiga kelompok utama: *data training*, *data validation*, dan *data testing*.

Data *training* digunakan untuk melatih model *Deep Neural Network* agar mampu mengenali pola dalam data. Sementara itu, data *validation* dimanfaatkan untuk mengevaluasi kinerja model selama proses pelatihan dan memastikan model tidak mengalami *overfitting*. Setelah pelatihan selesai, data *testing* digunakan untuk menguji performa model yang telah terbentuk. Proses pembentukan model mencakup tahap *training* dan *modelling*, yang pada akhirnya menghasilkan *learned model*.

Setelah model selesai dibangun, sistem digunakan untuk mendeteksi dan memperbaiki kesalahan ejaan dalam teks yang diberikan oleh pengguna. Untuk menjamin akurasi hasil koreksi, model yang telah dilatih dievaluasi menggunakan data *validation*. Jika hasil evaluasi menunjukkan performa yang memadai, proses koreksi dilakukan menggunakan *learned model*. Namun, jika hasil validasi kurang memuaskan, algoritma *Damerau-Levenshtein Distance* digunakan sebagai metode alternatif. Hasil akhir dari keseluruhan proses ini adalah kalimat yang telah dikoreksi dengan tingkat akurasi yang tinggi, sehingga makna pesan tetap terjaga tanpa perubahan. Arsitektur umum dari proses penelitian ditunjukkan pada Gambar 3.1.



Gambar 3. 1 Arsitektur Umum

3.2 Pre-processing

Pada proses ini, data diolah untuk menghasilkan data yang lebih berkualitas dan mudah dipahami oleh model. Hasil dari proses *tokenization* digunakan sebagai *vocabulary* untuk algoritma *Damerau Levenshtein Distance*. Sementara itu, hasil *tokenization* yang dilanjutkan dengan *word to integer* digunakan sebagai *vocabulary* untuk model *deep neural network*. Pada 4000 pasangan kalimat benar-salah, dilakukan pemisahan data dengan rasio umum, yaitu 70% atau 2800 kalimat untuk data pelatihan, 15% atau 600 kalimat untuk data validasi, dan 15% atau 600 kalimat sisanya untuk data pengujian. Rasio ini dipilih dengan mempertimbangkan kebutuhan algoritma *Damerau-Levenshtein Distance*, yang dirancang untuk

menghitung jarak edit dengan memperhitungkan operasi seperti penyisipan, penghapusan, substitusi, dan transposisi. Menurut penelitian yang telah dilakukan oleh Adinugroho (2022), pembagian data harus disesuaikan dengan sifat algoritma yang digunakan untuk memastikan performa optimal dalam pelatihan, validasi, dan pengujian. Algoritma *Damerau-Levenshtein Distance* memerlukan data pelatihan yang cukup besar untuk mengidentifikasi pola perbandingan karakter dengan akurat, sementara data validasi berfungsi untuk mencegah *overfitting* dan menjamin generalisasi model pada data baru. Selain itu, data pengujian yang memadai sangat penting untuk mengevaluasi kinerja algoritma secara objektif. Dengan demikian, rasio 70:15:15 dianggap paling sesuai karena mampu menciptakan keseimbangan yang ideal sesuai dengan kebutuhan spesifik algoritma *Damerau-Levenshtein Distance*.

3.2.1 Record Punctuation

Proses ini mencatat posisi tanda baca dalam sebuah string yang dimulai dengan memecah string menjadi daftar kata, kemudian mencari tanda baca tertentu (seperti titik, koma, atau tanda tanya) di setiap kata. Jika tanda baca ditemukan, tanda tersebut bersama posisinya disimpan ke dalam daftar. Output akhirnya adalah daftar posisi tanda baca dalam *string*.

```

FUNCTION RecordPunctuationByWord(string):
    Split the string INTO a list of words
    DEFINE a pattern to search for punctuation marks (e.g., '.',
    ',', '?')
    INITIALIZE an empty list called punctuation_positions

    FOR EACH word AND its index IN the list of words:
        SEARCH for a punctuation mark IN the word
        IF a punctuation mark IS found:
            EXTRACT the matched punctuation mark
            APPEND the punctuation mark AND its position (index +
1) TO punctuation_positions

    RETURN punctuation_positions

```

Pseudocode 3. 2 Mencatat tanda baca

Berikut contoh *record punctuation* pada tabel 3.2.

Tabel 3. 2 Contoh *record punctuation*

Contoh Kalimat	Record punctuation
Halo, apa kabar? Semoga baik.	[(',', 1), ('?', 3), ('.', 5)]

Fungsi ini mencatat posisi tanda baca seperti koma, tanda tanya, atau titik dalam setiap kata, kemudian menyimpannya bersama dengan posisi kata tersebut. Hasil dari proses ini adalah daftar posisi tanda baca, seperti [(', 1), ('?', 3), ('.', 5)], yang dimana artinya posisi ', ' dicatat pada setelah kata ke-1, lalu ' ? ' berada pada kata ke-3, dan ' . ' pada setelah kata ke-5, sementara teks diubah menjadi "Halo apa kabar Semoga baik" dengan tanda baca dihapus untuk proses lebih lanjut.

3.2.2 Data Cleaning

Data Cleaning merupakan proses membersihkan data yang bertujuan untuk memastikan bahwa data yang digunakan dalam analisis atau pengambilan keputusan adalah akurat, konsisten, dan mengurangi risiko dari kesalahan. Pada tahap ini, dilakukan peninjauan terhadap data yang ada, dan jika ditemukan duplikasi atau nilai yang hilang, maka data tersebut akan dihapus. Pada penelitian ini, fungsi *cleaning string* merupakan bagian dari proses *preprocessing* yang berfungsi untuk membersihkan data teks dari komponen-komponen yang tidak mendukung analisis, seperti tanda baca yang tidak diperlukan, spasi berlebih, dan pengubahan semua huruf dalam data menjadi huruf kecil (*case folding*). Langkah ini memastikan teks memiliki format yang konsisten dan siap digunakan sebagai input untuk model. Berikut pseudocode dari proses *data cleaning* seperti pada pseudocode 3.3.

```
FUNCTION CleaningString(string):
```

```
    Remove all characters from the string EXCEPT letters, numbers,
    hyphens ('-'), and spaces
```

```
    Replace multiple consecutive spaces in the string WITH a
    single space
```

```
    Remove leading and trailing spaces from the string
```

```
Convert the string to lowercase
RETURN the cleaned string
```

Pseudocode 3.3 Tahap *Data Cleaning*

Proses ini melibatkan penghapusan karakter yang tidak diperlukan, kecuali huruf, angka, spasi, dan tanda hubung, lalu penggantian spasi ganda dengan spasi tunggal, pemotongan spasi di awal dan akhir string, serta mengubah seluruh karakter menjadi huruf kecil (*Case Folding*). Hasil akhirnya adalah string yang bersih dan terstandar. Contoh ditampilkan pada tabel 3.3.

Tabel 3.3 *Data Cleaning*

Contoh kalimat	Kalimat hasil <i>cleaning</i>
Selamat ulang tahun	selamat ulang tahun

Pada contoh kalimat pada tabel 3.3, didapatkan hasil berupa penghapusan spasi berlebih dan merubah karakter menjadi huruf kecil.

3.2.3 *Tokenization*

Tokenization adalah proses memecah rangkaian karakter dalam data menjadi unit-unit kecil yang disebut token dan dibagi menjadi beberapa bagian, yang bertujuan untuk memudahkan analisis lebih lanjut. Berikut pseudocode dari proses *tokenization* seperti pada pseudocode 3.4.

```
DEFINE FUNCTION sclstm_tokenize(sentences, vocab):
    RETRIEVE character token indices and unknown token index from
    vocab

    DEFINE FUNCTION vectorize(word):
        INITIALIZE vectors for start, middle, and end characters
        ASSIGN value to start vector based on the first character in
        the word
        UPDATE middle vector for each character in the middle of the
        word
        ASSIGN value to end vector based on the last character in the
        word
        RETURN concatenated start, middle, and end vectors
```

```

APPLY vectorize TO each word in the sentences
RETURN tokenized sentences

```

Pseudocode 3. 4 Tahap *Tokenize*

Tahap ini mentransformasi teks input berupa kalimat menjadi representasi numerik berbasis karakter. Setiap kata dipecah menjadi tiga bagian utama: karakter awal, karakter tengah, dan karakter akhir, yang kemudian dipetakan ke indeks tertentu berdasarkan kosakata. Representasi ini membantu model memahami pola dalam struktur kata dengan lebih mendalam. Contoh penerapan *Tokenize* pada tabel 3.4.

Tabel 3. 4 Tahap *Tokenize*

Contoh kalimat	Kalimat hasil <i>tokenize</i>
selamat ulang tahun	[“selamat”, “ulang”, “tahun”]

3.2.4 *Word to Integer*

Pada proses ini, penulis juga menggunakan fungsi *word to integer* untuk mengonversi kata-kata dalam teks menjadi representasi numerik berupa angka-angka, dengan mengubah kata-kata menjadi angka, data dapat dipahami dan mudah untuk diolah. Berikut pseudocode dari proses *word to integer* seperti pada pseudocode 3.5.

```

FUNCTION word_to_integer(word, token_to_idx, unk_token):
    IF word EXISTS in token_to_idx:
        RETURN token_to_idx[word]
    ELSE:
        RETURN token_to_idx[unk_token]

```

Pseudocode 3. 5 Tahap *Word to Integer*

Hasil dari proses ini berupa perubahan kata representasi numerik untuk pemrosesan oleh model. Contoh penerapan *word to integer* pada tabel 3.5.

Tabel 3. 5 Contoh *Word to Integer*

Kalimat <i>tokenize</i>	<i>Word to Integer</i>
[“selamat”, “ulang”, “tahun”]	{ 'selamat' : 1, 'ulang' : 2, 'tahun' : 3 }

3.2.5 Labelize

Proses *labelize* digunakan untuk mengonversi mengubah setiap kalimat atau *string* menjadi menjadi *tensor* indeks numerik, yang juga dimana *labelize* menggunakan *word to integer* sebagai komponen utama untuk memproses setiap kata dalam kalimat.

```

DEFINE FUNCTION labelize(labels, vocab):
    RETRIEVE token indices, padding token, and unknown token from
    vocab

    INITIALIZE an empty list for processed labels
    FOR each label sequence in labels:
        INITIALIZE an empty list for tokenized labels
        FOR each token in the sequence:
            MAP token to index if it exists; otherwise, use the
            unknown token index
            ADD mapped index to the tokenized list
        END FOR
        ADD tokenized list to processed labels
    END FOR

    PAD processed labels to uniform length
    COMPUTE lengths of each sequence
    RETURN padded labels and their lengths

```

Pseudocode 3. 6 Tahap *Labelize*

Pada tahap ini mengonversi label teks menjadi representasi numerik yang sesuai dengan kosakata. Proses ini mencakup pemetaan setiap token label ke indeks kosakata, penanganan token yang tidak dikenal, serta *padding* untuk memastikan semua label memiliki panjang yang seragam. Fungsi ini juga menghitung panjang

asli label sebelum padding untuk referensi lebih lanjut. Contoh penerapan *Labelize* pada tabel 3.6.

Tabel 3. 6 Contoh *Labelize*

Word to Integer	<i>Labelize</i>
{ 'selamat' : 1, 'ulang' : 2, 'tahun' : 3 }	Tensor: [1, 2, 3, 99, 99] Panjang Asli: 3

Jika panjang batch misalnya berjumlah 5, tambahkan padding hingga mencapai panjang maksimal batch, pada contoh angka “99” digunakan sebagai *Unknown Token* atau token khusus yang digunakan untuk menggantikan kata-kata yang tidak ditemukan dalam kamus atau sebagai *padding* untuk proses menambahkan token khusus untuk membuat semua data input memiliki panjang yang seragam.

3.2.6 *Untokenize*

Tahap ini menginterpretasikan hasil prediksi model. Fungsi ini mengubah indeks numerik hasil prediksi kembali menjadi teks dengan menggunakan kosakata. Proses ini mempertimbangkan panjang asli label sebelum padding untuk memastikan hasilnya sesuai dengan format data awal.

```

DEFINE FUNCTION untokenize(predictions, lengths, vocab):
RETRIEVE index-to-token mapping and unknown token from vocab
    INITIALIZE an empty list for reconstructed sentences
    FOR each prediction and its length:
MAP indices to tokens for the valid range in the prediction
    JOIN tokens into a single sentence
    ADD reconstructed sentence to the output list
END FOR
RETURN reconstructed sentences

```

Pseudocode 3. 7 Tahap *untokenize*

Contoh dari *untokenize* pada tabel 3.7.

Tabel 3. 7 Contoh Tahap *Untokenize*

Tensor	Untokenize
Tensor: [1, 2, 3, 99, 99]	selamat ulang tahun

3.2.7 Dataset

Untuk mengembangkan sistem, diperlukan persiapan kumpulan data yang relevan sebagai sumber informasi dan bahan pelatihan bagi sistem yang akan dibangun. Dalam kasus koreksi ejaan *real-word error*, dataset yang dibutuhkan mencakup daftar kata yang valid beserta penggunaannya dalam kalimat secara kontekstual. Dataset ini dikembangkan secara mandiri, artinya tidak menggunakan dataset yang telah ada.

3.3 Pembentukan sistem

Tahap ini bertujuan untuk mengolah dataset yang dibentuk sebelumnya menjadi sebuah sistem.

3.3.1 Koreksi kata *non-word errors* oleh Damerau Levenshtein Distance

Pada tahap ini sistem akan melakukan koreksi kalimat menggunakan algoritma *Damerau-Levenshtein Distance* untuk mengurangi kemungkinan terjadinya *non-word errors* dan *out of vocabulary* (OOV), serta agar lebih memusatkan perhatian pada *real-word errors*. Setiap kata yang akan dikoreksi menghasilkan kandidat kata dengan batas jarak yang akan diatur dengan bobot operasi minimal dan maksimal perubahan koreksi. Berikut adalah pseudocode untuk tahap koreksi kata *non-word errors* menggunakan algoritma *Damerau-Levenshtein Distance* pada pseudocode 3.8.

```

DEFINE FUNCTION generate_candidates(word, vocabs,
distance_threshold, limit, weights):
    INITIALIZE candidate_list as an empty list

    FOR each vocab in vocabs:
        CALCULATE distance using Damerau-Levenshtein algorithm
        with word, vocab, and weights
        IF distance is less than or equal to distance_threshold:
            ADD {candidate: vocab, distance: distance} to
            candidate_list
        END IF
    END FOR
  
```

```

    SORT candidate_list in ascending order by distance

    IF size of candidate_list is greater than or equal to limit:
        SELECT the first `limit` elements from candidate_list as
top_candidates
    ELSE:
        ASSIGN all elements of candidate_list to top_candidates
    END IF

    RETURN list of candidates from top_candidates
END FUNCTION

```

Pseudocode 3. 8 Tahap Koreksi Kata *Non-word Errors*

Pada tahap ini, sistem melakukan perbaikan terhadap *non-word errors* menjadi kata yang benar atau *real-word errors* dengan menerapkan algoritma Damerau-Levenshtein Distance, yang digunakan untuk menghitung jarak edit antara kata masukan dan kata-kata dalam kosakata. Proses ini mencakup empat jenis operasi utama, yaitu penyisipan, penghapusan, penggantian, dan pertukaran posisi karakter. Setiap operasi diberikan bobot sesuai tingkat kesulitannya, di mana penyisipan (*insert*) dan penghapusan (*delete*) masing-masing memiliki bobot 0.5, sementara penggantian (*substitution*) dan pertukaran (*transposition*) diberi bobot 1. Penggunaan bobot operasi ini dirancang untuk memprioritaskan perubahan kecil pada kata dibandingkan perubahan besar dan menunjukkan bahwa bobot tersebut secara efektif mencerminkan tingkat kompleksitas setiap operasi dan meningkatkan akurasi koreksi ejaan pada kalimat (Santoso et al., 2019).

Untuk membatasi hasil koreksi agar lebih relevan, sistem menetapkan jarak maksimum operasi sebesar 1.5 bobot. Hal ini memastikan kandidat koreksi yang dihasilkan memiliki kemiripan tinggi dengan kata asli, sehingga mengurangi kemungkinan menghasilkan kata yang tidak sesuai. Setelah perhitungan jarak selesai, kandidat yang memenuhi kriteria diurutkan berdasarkan jarak edit terkecil, dan hingga lima kandidat terbaik dipilih untuk setiap kata. Proses ini memastikan bahwa kandidat yang dipilih relevan dan sesuai dengan struktur kalimat. Berikut contoh koreksi kata *non-word errors* pada tabel 3.8.

Tabel 3. 8 Contoh koreksi kata *non-word errors*

Kalimat mengandung <i>non-word errors</i>	Koreksi kata <i>non-word errors</i> oleh <i>Damerau-Levenshtein Distance</i>
Saya sedang makan buah rambuta.	Saya sedang makan buah rambut

Berdasarkan contoh pada tabel 3.8, kalimat yang mengandung *non-word errors* yaitu kata “rambuta” berubah menjadi “rambut”. Contoh kandidat kata hasil proses sistem sebagai pengganti *non-word errors* pada tabel 3.9.

Tabel 3. 9 Contoh kandidat kata oleh Damerau-Lavenshtein Distance

Kandidat pengganti kata “rambuta”	Bobot operasi <i>Damerau-Levenshtein Distance</i>	Operasi
Rambut	0.5	<i>Delete</i>
Rambutan	0.5	<i>Insert</i>
Rambuti	1	<i>Substitution</i>
Rambu	1	<i>Delete, Delete</i>

Selanjutnya, kandidat terbaik yang telah diseleksi dapat diproses lebih lanjut pada tahap koreksi berikutnya menggunakan metode berbasis konteks, seperti Deep Neural Network. Langkah ini memastikan bahwa koreksi yang dihasilkan tidak hanya sesuai secara leksikal, tetapi juga selaras dengan konteks kalimat secara keseluruhan.

3.3.2 Deep Neural Network

Deep Neural Network memanfaatkan data yang dibentuk dari tahap sebelumnya untuk menganalisis konteks kalimat secara mendalam melalui lapisan-lapisan tersembunyi, seperti *Long Short-Term Memory* (LSTM) dan *dropout*, guna mengidentifikasi dan memperbaiki *real-word errors*. Proses ini melibatkan pemetaan kata dalam teks ke representasi numerik, yang kemudian digunakan oleh model untuk memilih kata yang paling sesuai berdasarkan konteks. Model ini memanfaatkan fungsi aktivasi *softmax* dan arsitektur yang dapat disesuaikan sesuai dengan tingkat kompleksitas dan kebutuhan spesifik tugas *spell correction*. Berikut

adalah pseudocode pembuatan model untuk mengidentifikasi dan memperbaiki *real-word errors* menggunakan algoritma *Deep Neural Network* pada pseudocode 3.9.

```

DEFINE CLASS SCLSTM:
    FUNCTION initialize(screp_dim, padding_idx, output_dim):
        SET bidirectional to True
        SET hidden_size to 512
        SET nlayers to 2

        INITIALIZE LSTM module with:
            input dimension = screp_dim
            hidden size = hidden_size
            number of layers = nlayers
            batch processing enabled
            dropout probability = 0.4
            bidirectionality = True

        DETERMINE output dimension of LSTM module:
        IF bidirectional is True:
            SET lstmmodule_output_dim to hidden_size * 2
        ELSE:
            SET lstmmodule_output_dim to hidden_size

        VERIFY that output_dim is greater than 0

        INITIALIZE dropout layer with probability 0.5
        INITIALIZE dense layer to map LSTM output to the specified
output dimension

        INITIALIZE loss function as Cross-Entropy Loss with:
            mean reduction
            ignoring indices specified by padding_idx

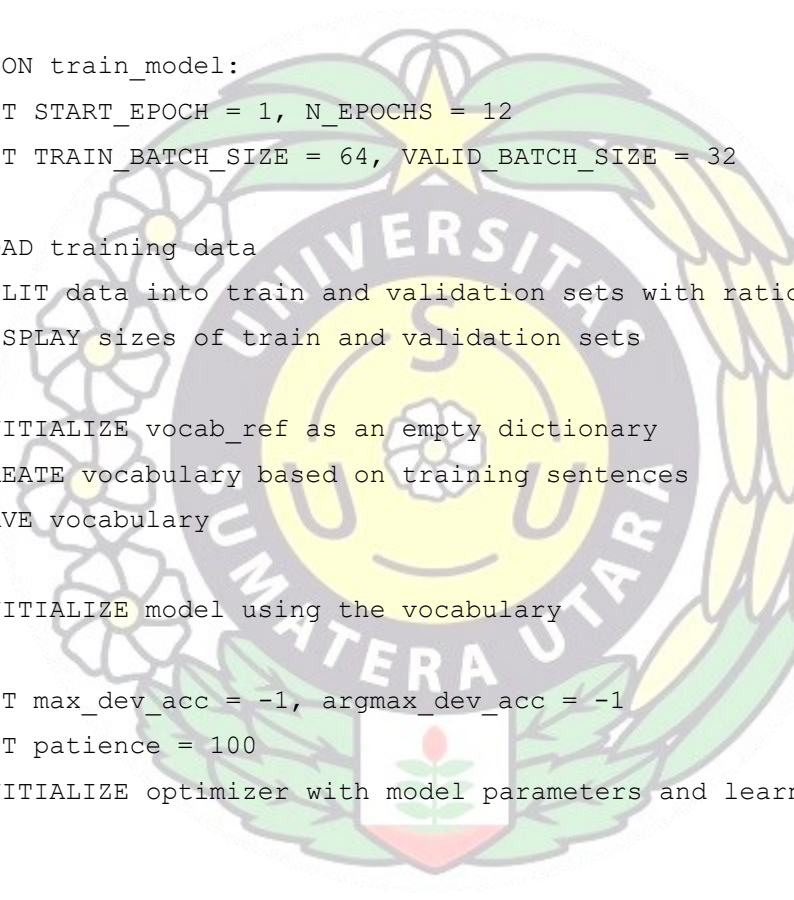
END CLASS

```

Pseudocode 3. 9 Tahap *Deep Neural Network*

3.3.3 Train dan modelling data

Pembelajaran model agar dapat mengenali jenis kesalahan eja yang terdapat pada kalimat. Sebelum melatih model, penting untuk menentukan hyperparameter terlebih dahulu. Beberapa hyperparameter yang dilatih dalam model *Deep Neural Network* pada penelitian ini meliputi jumlah *epoch*, ukuran *batch*, nilai *learning rate*, dan *dropout*, kemudian *learned model* akan digunakan ke dalam sistem berbasis web. Berikut adalah pseudocode untuk tahap *train* dan *modelling data* pada pseudocode 3.10.



```

FUNCTION train_model:
    SET START_EPOCH = 1, N_EPOCHS = 12
    SET TRAIN_BATCH_SIZE = 64, VALID_BATCH_SIZE = 32

    LOAD training data
    SPLIT data into train and validation sets with ratio 0.8
    DISPLAY sizes of train and validation sets

    INITIALIZE vocab_ref as an empty dictionary
    CREATE vocabulary based on training sentences
    SAVE vocabulary

    INITIALIZE model using the vocabulary

    SET max_dev_acc = -1, argmax_dev_acc = -1
    SET patience = 100
    INITIALIZE optimizer with model parameters and learning rate =
0.005

    FOR each epoch from START_EPOCH to N_EPOCHS:
        IF patience threshold exceeded:
            DISPLAY early stopping message
            BREAK the loop

        DISPLAY current epoch

        INITIALIZE train_loss, train_acc, train_acc_count to 0
        CREATE training data iterator
  
```

```

FOR each batch in training data:
    RESET optimizer gradients
    TOKENIZE labels and sentences
    TRAIN model with batch data
    UPDATE train_loss
    PERFORM periodic accuracy calculation

DISPLAY average training loss for the epoch

INITIALIZE valid_loss, valid_acc to 0
CREATE validation data iterator

FOR each batch in validation data:
    TOKENIZE labels and sentences
    EVALUATE model with batch data
    UPDATE valid_loss and valid_acc
DISPLAY average validation loss for the epoch

IF validation accuracy improves:
    SAVE model and optimizer state
    DISPLAY model saved message
    UPDATE max_dev_acc and argmax_dev_acc

RETURN vocabulary and trained model

```

Pseudocode 3. 10 Tahap *train* dan *modelling data*

Setelah mendapatkan *learned model* terbaik, kemudian model akan digunakan untuk mengidentifikasi dan memperbaiki *real-word errors*. Berikut contoh koreksi *real-word errors* sebelumnya oleh *learned model Deep Neural Network* pada tabel 3.10.

Tabel 3. 10 Contoh koreksi *real-word errors* oleh *learned model Deep Neural Network*

<i>Real-word errors</i>	Koreksi <i>real-word errors</i> oleh <i>learned model</i>
Saya sedang makan buah rambut	Saya sedang makan buah rambutan

Selanjutnya, koreksi dihasilkan oleh proses ini akan diproses lebih lanjut pada tahap koreksi berikutnya menggunakan algoritma *Damerau-Levenshtein Distance*. Langkah ini memastikan bahwa koreksi kata yang dihasilkan tidak terlalu jauh dengan kata aslinya, sekaligus agar tidak melewati bobot operasi yang sudah ditentukan, dengan begitu diharapkan hasil akhir terbaik secara leksikal, dan selaras dengan konteks kalimat secara keseluruhan.

3.3.4 Koreksi jarak bobot oleh *Damerau-Levenshtein Distance*

Proses ini berguna untuk membandingkan perubahan jarak bobot operasi dari hasil koreksi *real-word errors* oleh *learned model Deep Neural Network* dengan hasil koreksi *non-word errors* oleh *Damerau-Levenshtein Distance*.

Tabel 3. 11 Contoh koreksi jarak bobot oleh *Damerau-Lavenshtein Distance*

Tahap Koreksi	Hasil Koreksi	Bobot Operasi
Koreksi <i>non-word errors</i> oleh <i>Damerau-Levenshtein Distance</i> .	Saya sedang makan buah rambut	0.5
Koreksi <i>real-word errors</i> oleh <i>learned model Deep Neural Network</i>	Saya sedang makan buah rambutan	0.5

Pada contoh kasus ini, masing masing tahap koreksi memiliki jarak bobot operasi yang sama yaitu berjumlah 0.5, maka dari itu sistem akan memilih *output* dari koreksi kedua yaitu pada hasil koreksi *real-word errors* oleh *learned model*

Deep Neural Network untuk dijadikan kalimat koreksi akhir yaitu “Saya sedang makan buah rambutan”.

3.3.5 Restored Punctuation

Proses ini mengembalikan tanda baca ke posisi semula dalam sebuah *string* berdasarkan daftar posisi tanda baca. Proses ini dilakukan dengan memecah *string*, memeriksa apakah posisi tanda baca berada dalam rentang indeks kata, kemudian menambahkan tanda baca ke kata yang sesuai, lalu menyusun ulang *string* sesuai dengan kalimat akhir yang sudah diperiksa nantinya.

Tabel 3. 12 *Restored Punctuation*

Kalimat koreksi akhir	<i>Restored Punctuation</i>
Saya sedang makan buah rambutan	Saya sedang makan buah rambutan.

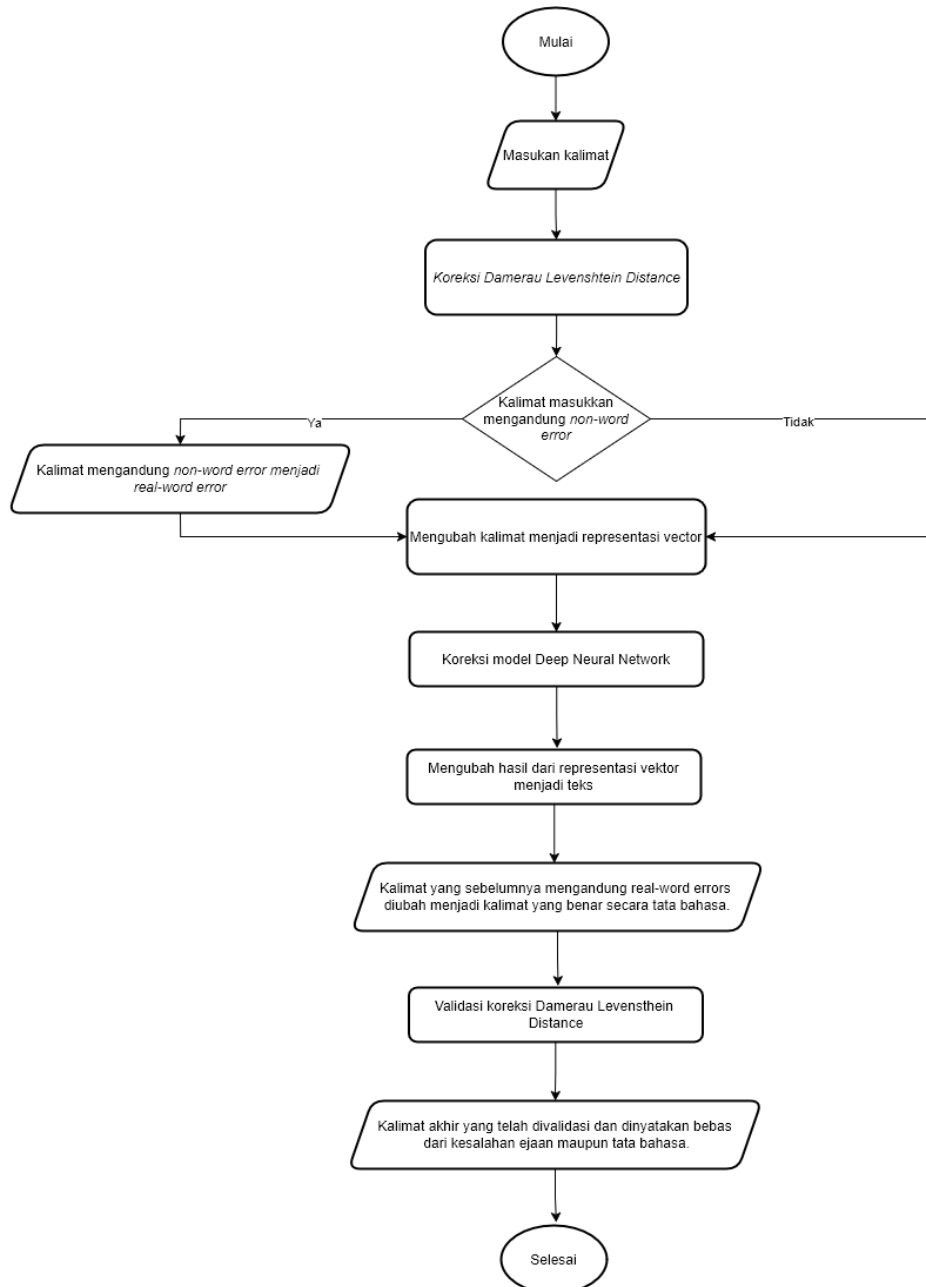
Dengan langkah-langkah ini, sistem dirancang untuk mendeteksi dan memperbaiki kesalahan penulisan secara akurat, sehingga pengguna memperoleh saran perbaikan yang sesuai tanpa mengubah makna yang diinginkan.

3.4 Perancangan Alur Sistem

Perancangan alur sistem merupakan langkah penting dalam membangun sistem. Tahap ini mencakup pemetaan rinci langkah operasional untuk memastikan setiap komponen berfungsi secara terorganisir. Alur dimulai dengan pengumpulan data menggunakan teknik *web scraping* untuk membentuk *parallel corpus* sebagai sumber data utama. Data yang diperoleh kemudian diproses melalui tahap *pre-processing*, termasuk *data cleaning* untuk menghapus karakter tidak relevan, *case folding* untuk menyeragamkan huruf kecil, *tokenization* untuk memecah teks menjadi kata, dan *word-to-integer* untuk mengonversi kata menjadi representasi numerik.

Pada tahap pengolahan utama, algoritma *Damerau-Levenshtein Distance* digunakan untuk menghitung jarak edit dan mendeteksi kesalahan, sementara model *Deep Neural Network* (DNN) menganalisis konteks kalimat untuk

mendeteksi *real-word errors*. Sistem kemudian mengevaluasi hasil untuk menentukan metode koreksi, apakah menggunakan model DNN yang telah dilatih atau algoritma *Damerau-Levenshtein* sebagai alternatif. Dengan pendekatan ini, sistem dirancang untuk menghasilkan koreksi ejaan yang relevan dan akurat tanpa mengubah makna asli pesan. Diagram alur sistem ditampilkan pada Gambar 3.2.



Gambar 3. 2 Diagram Alir Sistem

Berdasarkan diagram alur, proses koreksi ejaan pada sistem berlangsung sebagai berikut:

1. Mulai.
2. Pengguna memasukkan kalimat yang akan diperiksa dan dikoreksi ke dalam sistem.
3. Kalimat *input* akan diperiksa untuk mengetahui apakah terdapat *non-word errors*, jika terdapat *non-words errors*, kalimat akan dikoreksi dengan algoritma *Damerau Levensthein Distance*, dan jika tidak terdapat *non-word errors*, kalimat akan langsung dikonversi menjadi representasi vektor.
4. Koreksi Tahap Pertama: Sistem menggunakan algoritma *Damerau Levensthein Distance* untuk mengoreksi kalimat. Tujuannya adalah meminimalkan kesalahan *non-word* dan kata yang tidak ada dalam kosakata (*out-of-vocabulary* atau OOV), serta memfokuskan koreksi pada *real-word errors*. Pada tahap ini, sistem membangkitkan kandidat koreksi untuk setiap kata dengan jarak satu operasi dan maksimal lima kandidat. Bobot yang digunakan untuk operasi seperti *delete* berbobot 0.5, *insert* berbobot 0.5, *substitution* berbobot 1 dan *transposition* berbobot 1.
5. Hasil koreksi tahap pertama disimpan untuk dokumentasi dan dilanjutkan ke langkah berikutnya.
6. Konversi Representasi Vektor: Hasil koreksi tahap pertama diubah menjadi representasi vektor berdasarkan kamus yang telah disiapkan sebelumnya.
7. Koreksi Tahap Kedua: Representasi vektor dikirim ke model *Deep Neural Network* (DNN) yang telah dilatih untuk diproses lebih lanjut.
8. Output dari model DNN dikonversi kembali dari representasi vektor menjadi teks.
9. Hasil koreksi tahap kedua disimpan sebagai dokumentasi dan proses dilanjutkan ke langkah berikutnya.
10. Koreksi Tahap Ketiga: Hasil koreksi tahap kedua divalidasi kembali menggunakan algoritma *Damerau Levenshtein Distance*. Tujuannya adalah meminimalkan perubahan yang tidak relevan atau terlalu jauh yang mungkin terjadi akibat proses pada model *Deep Neural Network*. Pada tahap ini, kandidat koreksi setiap kata dibangkitkan dengan jarak maksimal 1,5 bobot operasi dan lima kandidat per kata. Bobot operasi yang digunakan adalah 0,5 untuk

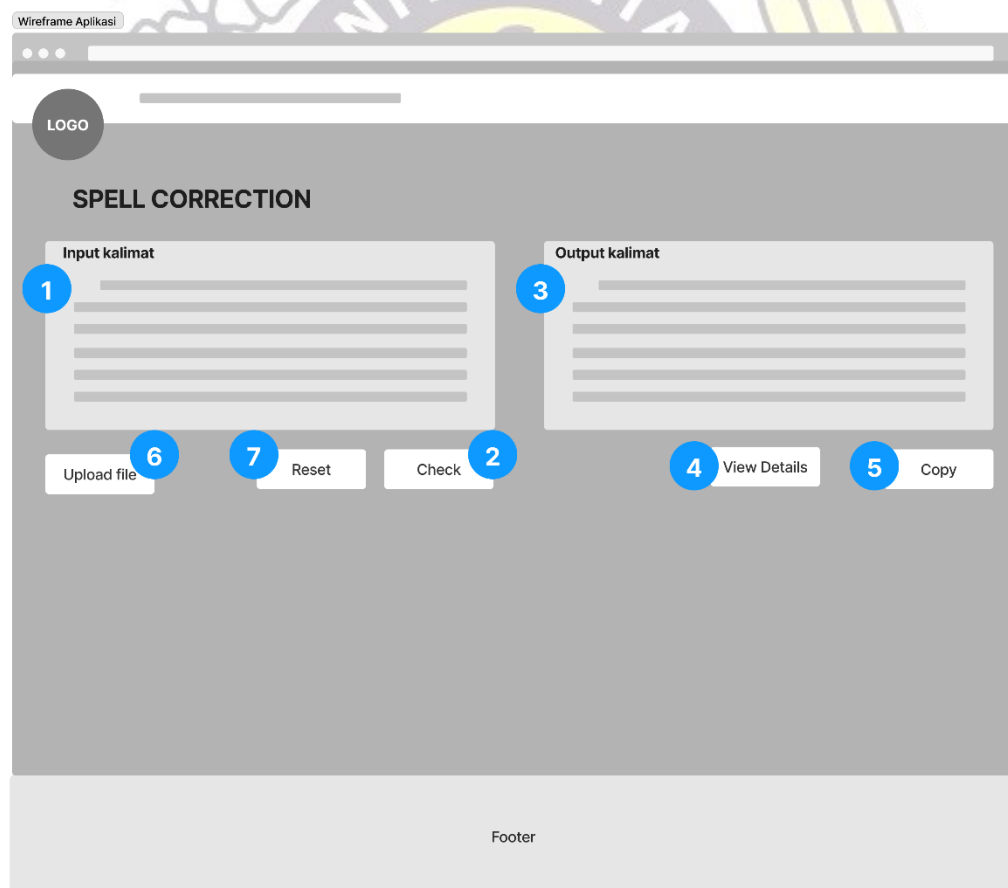
penghapusan, 0,5 untuk penyisipan, satu untuk penggantian dan satu untuk transposisi.

11. Hasil koreksi tahap ketiga disimpan sebagai dokumentasi serta hasil akhir dari proses koreksi.

12. Selesai.

3.4.1 Desain Halaman Antarmuka Aplikasi

Sistem ini akan diimplementasikan dalam bentuk aplikasi web. Aplikasi tersebut akan dikembangkan menggunakan *library Flask* sebagai API untuk menghubungkan *Javascript*, yang berfungsi sebagai inti pengoperasian web di *browser*, dengan *Python*, yang menjadi dasar model *Deep Neural Network*. Desain aplikasi dirancang seperti yang ditampilkan pada gambar 3.3.



Gambar 3. 3 Halaman antarmuka aplikasi

Dari desain antarmuka pada gambar 3.3, terlihat bahwa aplikasi terdiri dari

tujuh komponen, dengan fungsi masing-masing sebagai berikut :

1. Bidang teks masukan

Berfungsi sebagai area di mana pengguna dapat memasukkan teks yang ingin diperiksa dan diperbaiki ejaannya. Teks yang dimasukkan dapat berupa satu atau lebih kalimat.

2. Tombol *check*

Merupakan komponen yang digunakan untuk memproses kalimat-kalimat yang ada pada bidang teks masukan sesuai dengan alur dan langkah-langkah yang telah dirancang dan menampilkan hasilnya pada bidang area keluaran.

3. Bidang teks keluaran

Merupakan area di mana tampilan keluaran berupa kalimat yang telah diperbaiki.

4. Tombol *view detail*

Digunakan untuk menampilkan bidang teks detail koreksi beserta detail koreksi dari hasil proses pengecekan ejaan yang telah dilakukan.

5. Tombol *copy*

Mempermudah pengguna untuk menyalin teks yang ditampilkan di bidang teks keluaran.

6. Tombol *upload*

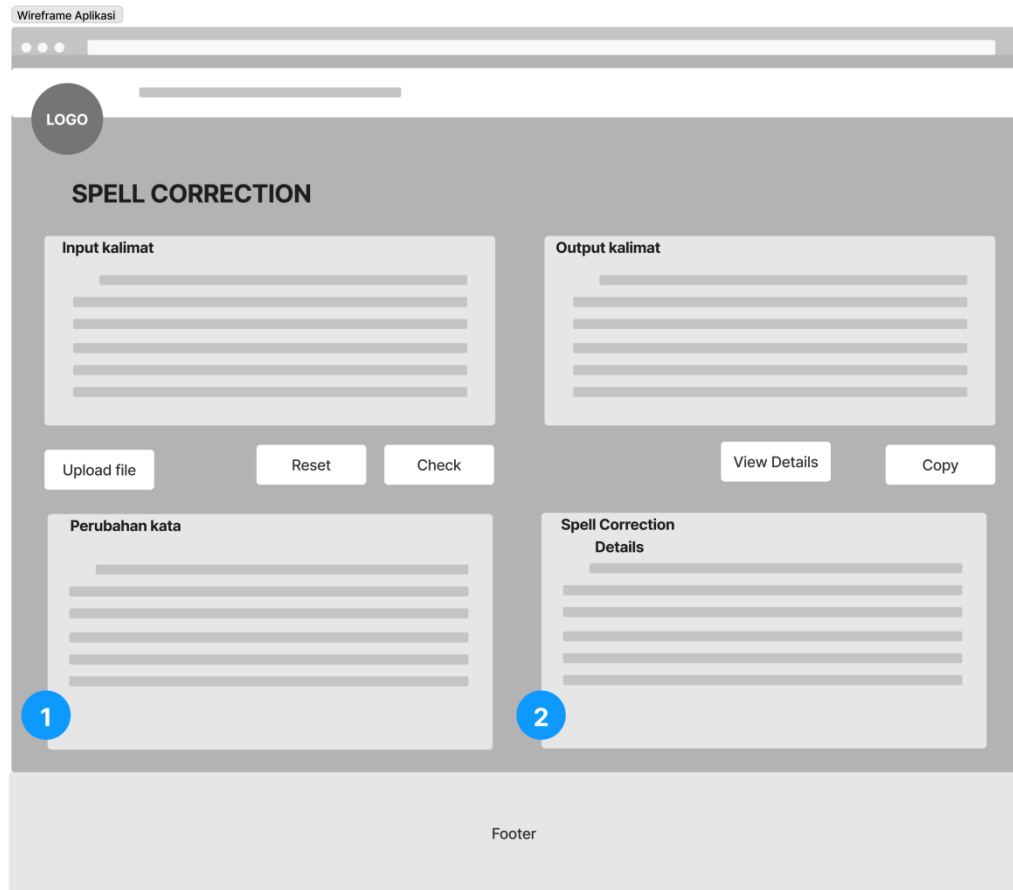
Memungkinkan pengguna mengunggah file berformat *.txt* dan *.pdf* untuk memudahkan input teks dalam jumlah banyak ke dalam bidang teks masukan.

7. Tombol *reset*

Berfungsi untuk menghapus seluruh isi dari bidang teks yang ada.

3.4.2 Desain Halaman *View Details*

Halaman *view details* adalah wadah yang mengizinkan pengguna untuk melihat informasi lebih rinci terkait perubahan kalimat yang sudah diproses oleh sistem, seperti data hasil koreksi, dan rincian proses, yang bertujuan untuk memberikan pengalaman pengguna yang lebih baik dengan akses langsung ke informasi tambahan. Desain Halaman *View Details* ditampilkan pada gambar 3.4.



Gambar 3. 4 Halaman *view details* aplikasi

Tampilan halaman *view details* ini hanya dapat diakses ketika pengguna menekan tombol *view details*. Halaman ini terdiri dari 2 komponen dengan fungsi masing-masing sebagai berikut :

1. Bidang teks perubahan kata

Sebagai area yang menampilkan perubahan dari kata awal dengan kata yang sudah diperbaiki yang diurut berdasarkan urutan kalimatnya

2. Bidang teks detail koreksi

Sebagai area yang menjelaskan detail hasil koreksi kalimat setiap langkahnya, mulai dari awal hingga hasil akhir.

3.5 Metode Evaluasi

Setelah aplikasi selesai dikembangkan, pengujian dilakukan baik secara otomatis maupun langsung. Berdasarkan hasil pengujian, sistem dievaluasi menggunakan beberapa metrik untuk menilai kemampuan dalam memperbaiki kata atau token

secara individual. Metrik yang dihitung meliputi akurasi dan tingkat koreksi kata, guna mengukur efektivitas sistem dalam memperbaiki setiap kata yang mengalami kesalahan. Evaluasi ini mencakup parameter seperti *precision*, *recall*, *F1-score*, dan akurasi, yang memberikan wawasan tentang kinerja model, dengan menggunakan analisis *confusion matrix*. Penerapan Confusion Matrix disajikan dalam Tabel 3.8 sebagai ilustrasi dari evaluasi yang akan dilakukan.

Tabel 3. 13 Ilustrasi *Confusion Matrix*

	Aktual Positif	Aktual Negatif
Identifikasi Positif	TP (<i>True Positive</i>)	FP (<i>False Positive</i>)
Identifikasi Negatif	FN (<i>False Negative</i>)	TN (<i>True Negative</i>)

Keterangan :

TP = Volume data yang teridentifikasi sebagai positif dan serasi dengan kelas aktual yang bernilai positif.

TN = Volume data yang teridentifikasi sebagai negatif dan serasi dengan kelas aktual yang bernilai negatif.

FP = Volume data yang teridentifikasi sebagai positif dan tidak serasi dengan kelas aktual yang bernilai negatif.

FN = Volume data yang teridentifikasi sebagai negatif dan tidak serasi dengan kelas aktual yang bernilai positif.

BAB 4

IMPLEMENTASI DAN PENGUJIAN

Pada bab ini berisikan tentang penjelasan mengenai proses implementasi dan hasil uji coba algoritma *Damerau-Levenshtein Distance* dan pendekatan dari model *Deep Neural Network* sesuai dengan rancangan sistem pada bab sebelumnya.

4.1 Implementasi Sistem

Pada tahap ini akan dilakukan implementasi algoritma *Damerau-Levenshtein Distance* dan pendekatan *Deep Neural Network* untuk mendeteksi dan memperbaiki kesalahan eja atau *real-word errors* dalam Bahasa Indonesia.

4.1.1 Perangkat Keras dan Perangkat Lunak

Komponen perangkat keras yang digunakan penulis dalam merancang sistem tersebut adalah komputer dengan spesifikasi sebagai berikut:

- 1) CPU: AMD Ryzen 7 5800H
- 2) RAM: 32GB
- 3) Storage: 500GB
- 4) GPU: Nvidia GeForce RTX 3060
- 5) OS: Windows 10 Home Single Language 64-bit

Komponen perangkat lunak dan *library* yang digunakan untuk membangun sistem yaitu sebagai berikut:

- 1) Visual Studio Code
- 2) Google Colab
- 3) Python 3.12.7
- 4) Flask
- 5) Excel
- 6) *Library: beautifulsoup, numpy, os, pandas, pickle, regex, time, torch, tqdm,*

typing

4.1.2 Implementasi Perancangan Antarmuka

Dari perancangan yang dilakukan pada bab 3, maka berikut ini merupakan implementasi dari rancangan tersebut :

1. Halaman *Spell Correction*

Laman ini merupakan tataran awal yang pertama kali diakses oleh pengguna sewaktu memulai operasi sistem *web application* dengan bantuan kerangka kerja Flask yang sudah diimplementasi Model dan *vocabulary* didalamnya. Pada halaman ini, disediakan *input box* untuk pengguna untuk mengoreksi kalimat, dan jika ingin mengunggah file dengan format *.txt* maupun *.pdf* yang berisi kalimat yang ingin dikoreksi dengan tombol ‘Upload’. Kemudian, pengguna dapat memulai proses pelatihan model melalui tombol ‘Check’. Tampilan dari halaman *Spell Correction* diperlihatkan oleh Gambar 4.1.

Gambar 4. 1 Halaman *Spell Correction*

Selepas proses pelatihan berhasil, hasil koreksi akan dipresentasikan pada *textbox* ‘Hasil perbaikan disarankan’. Jika ingin melihat detail koreksi sistem yang berjalan, pengguna nantinya bisa menekan tombol ‘*View Detail*’ yang kemudian akan muncul dua *textbox* yaitu ‘Perubahan Kata’ dan ‘Detail Koreksi’. Di samping tombol ‘*View Detail*’ yang menyajikan detail keseluruhan tentang perubahan kalimat yang dikoreksi, laman ini juga menyediakan tombol ‘*Copy*’ untuk menyalin hasil perbaikan kalimat yang sudah dibuat oleh sistem, dan tombol ‘*Clear*’ untuk

membersihkan kalimat. Berikut tampilan aplikasi setelah proses koreksi pada gambar 4.2.

The screenshot displays the 'Real Word Spell Correction Bahasa Indonesia' web application. It features a main input area with a text area containing a paragraph about computer science and digital innovation. Below the text area are buttons for 'Upload File', 'Clear', and 'Check →'. To the right, there's a 'Detail koreksi:' section showing the correction steps: [SENTENCE 1], [STEP 1 - DLD CORRECTION], [STEP 2 - NN CORRECTION], and [STEP 3 - DLD VALIDATION]. Below the main text area, a 'Perubahan Kata:' section lists the corrected words for two sentences. The interface is clean and user-friendly, with a dark header bar and a light main content area.

Masukan kalimat*:

Ilmu komputer adalah b@idang y`ang te=rus berkembang; dann mem/puyai peran penting dalam hampir setiap aspek kehidupan. Dengan kemajuan pesat daslam tknology informasi, ilmu komputer telah menjadi dasar bagi banyak inovasi, termasuk salah satunya keamanan syber. Dalam dunia digital ini, keterampilan dalam ilmu komputer sangat di perlukan. Namun, masih banyak orang yang belum sepenuhnya memahami betapa pentngnya bidang ini. Pendidikan dalam ilmu komputer dapat membantu meningkatkan kemampuann analisis serta keterampilan teknis yang diperlukan untuk berinovasi dalam dunia yang terus berkembang. Bnvak pekerjaan yang membutuhkan

Hasil perbaikan disarankan:

inovasi, termasuk salah satunya keamanan siber. dalam dunia digital ini, keterampilan dalam ilmu komputer sangat di diperlukan. namun, masih banyak orang yang belum sepenuhnya memahami betapa penting bidang ini. pendidikan dalam ilmu komputer dapat membantu meningkatkan kemampuan analisis serta keterampilan teknik yang diperlukan untuk berinovasi dalam dunia yang terus berkembang. banyak pekerjaan yang membutuhkan ahli dalam bidang ini.

Upload File Clear Check → Close Detail Copy

Perubahan Kata:

Kalimat 1:

1. komputer → komputer
2. adalah! → adalah

Kalimat 2:

1. b@idang → bidang
2. y`ang → yang
3. te=rus → terus
4. berkembang; → berkembang
5. dann → dan
6. mem/puyai → meminuai

Detail koreksi:

[SENTENCE 1]
ilmu komputer adalah

[STEP 1 - DLD CORRECTION]
ilmu komputer adalah

[STEP 2 - NN CORRECTION]
ilmu komputer adalah

[STEP 3 - DLD VALIDATION]
ilmu komputer adalah

Gambar 4. 2 Tampilan setelah proses koreksi

4.2 Pengujian Sistem

Pengujian dilakukan secara langsung dengan memasukkan sejumlah kalimat ke dalam aplikasi web. Contoh beberapa kalimat yang telah diuji secara langsung disajikan pada Tabel 4.1.

Tabel 4. 1 Daftar Kalimat Pengujian Langsung

Kalimat salah	Kalimat benar	Hasil koreksi
media online juga berusaha menarik perhatian pembaca lengan memberikan informasi uang dapat menjawab kebutuhan dana menarik perhatian audiens.	media online juga berusaha menarik perhatian pembaca dengan memberikan informasi yang dapat menjawab kebutuhan dan menarik perhatian audiens.	media online juga berusaha menarik perhatian pembaca dengan memberikan informasi yang dapat menjawab kebutuhan dan menarik perhatian audiens.

clickbait merupakan bara yang digunakan oleh media online salam menaikkan jumlah pengunjung, traffic dan pageviews.	clickbait merupakan cara yang digunakan oleh media online dalam menaikkan jumlah pengunjung, traffic dan pageviews.	clickbait merupakan cara yang digunakan oleh media online dalam menaikkan jumlah pengunjung, traffic dan pageviews.
kata yang digunakan salam penelitian kini diperoleh melalui kuesioner dana observasi.	data yang digunakan dalam penelitian ini diperoleh melalui kuesioner dan observasi.	data yang digunakan dalam penelitian ini diperoleh melalui kuesioner dan observasi.
penelitian kini memberikan akurasi sekitar 72,3% @ dana modem sang dihasilkan rapat melakukan identifikasi judul cerita clickbait atau non-clickbait.	penelitian ini memberikan akurasi sekitar 72,3% dan model yang dihasilkan dapat melakukan identifikasi judul berita clickbait atau non-clickbait.	penelitian ini memberikan akurasi sekitar 72,3% dan model yang dihasilkan dapat melakukan identifikasi judul berita clickbait atau non-clickbait
semua cara pada akhirnya akan menghasilkan jawaban yang sama tetapi pasti mempunyai warga yang berbeda-beda, seperti misalnya kecepatan waktu untuk merespons data.	semua cara pada akhirnya akan menghasilkan jawaban yang sama tetapi pasti mempunyai harga yang berbeda-beda, seperti misalnya kecepatan waktu untuk merespons data.	semua cara pada akhirnya akan menghasilkan jawaban yang sama tetapi pasti mempunyai warna yang berbeda-beda, seperti misalnya kecepatan waktu untuk merespons data.

Berdasarkan hasil pengujian yang tercantum dalam Tabel 4.1, sistem menunjukkan potensi yang baik dalam mengidentifikasi dan memperbaiki kesalahan ejaan pada kalimat yang mengandung *real-word errors*. Dari total 5

kalimat yang diuji, ditemukan 16 kata dengan kesalahan ejaan. Secara keseluruhan, sistem berhasil memperbaiki 4 dari 5 kalimat secara sempurna, sementara 1 kalimat masih memiliki kesalahan yang belum dapat diperbaiki dengan optimal. Hal ini menunjukkan bahwa tingkat keberhasilan sistem dalam pengujian kalimat mencapai 80%. Jika dianalisis dari jumlah kata, sistem mampu mendeteksi dan memperbaiki 15 dari 16 kata yang salah. Namun, terdapat 1 kata yang belum berhasil dikoreksi.

Ketidaktepatan ini terutama disebabkan oleh keterbatasan dataset pelatihan yang kurang mencakup variasi pola dan kosakata untuk menangani kompleksitas bahasa Indonesia. Kata-kata dengan makna ganda atau konteks spesifik sering kali tidak terwakili dengan baik, sehingga model kesulitan mengenali dan memperbaikinya secara akurat. Frekuensi kemunculan kata dalam dataset juga menjadi faktor penting. Kata-kata yang jarang muncul cenderung diabaikan selama pelatihan, menghambat model dalam memahami pola dan konteksnya. Selain itu, dataset yang terbatas pada jenis teks tertentu, seperti teknis atau ilmiah, membatasi kemampuan model untuk menangani konteks yang lebih umum.

Diversifikasi dataset menjadi kunci. Dataset yang lebih beragam perlu mencakup pola kalimat, jenis teks, dan konteks yang luas, termasuk kosakata dari berbagai domain, seperti sastra dan komunikasi sehari-hari. Distribusi kata yang seimbang juga diperlukan untuk membantu model mengenali variasi bahasa dengan lebih baik dan meningkatkan akurasi koreksi ejaan secara keseluruhan. Secara umum, sistem menunjukkan kinerja yang memuaskan dalam memperbaiki kesalahan ejaan pada tingkat kalimat dengan tingkat keberhasilan yang tinggi. Namun, untuk menghasilkan performa yang lebih konsisten dan akurat, diperlukan pengembangan dataset yang lebih besar dan lebih beragam. Sistem ini memberikan kontribusi yang signifikan dalam meningkatkan kualitas teks berbahasa Indonesia, khususnya dalam konteks ilmiah dan teknologi, serta menawarkan peluang untuk penelitian lebih lanjut di bidang pengolahan bahasa alami (NLP) berbasis konteks.

4.3 Implementasi Model

4.3.1 Parallel Corpus

Untuk melatih model deep neural network dalam mendeteksi dan memperbaiki kesalahan ejaan pada teks dengan efektif, diperlukan pembentukan parallel corpus.

Berdasarkan data kalimat yang terkumpul dari abstrak karya ilmiah di repositori Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara, dibuat parallel corpus dalam format pasangan kalimat yang benar dan salah secara terpisah.

1	correct_sentence	incorrect_sentence
2	Dalam penelitian ini penulis menerapkan proses text mining dan proses n-gram karakter untuk seleksi fitur serta menggunakan algoritma Naive Bayes Classifier untuk mengklasifikasi sentimen secara otomatis.	Salam penelitian dini penulis menerapkan proses text mining dan proses n-gram karakter untuk seleksi fitur serta menggunakan algoritma Naive Bayes Classifier untuk mengklasifikasi sentimen secara otomatis.
3	Penulis menggunakan 3300 data tweet tentang sentimen kepada provider telekomunikasi.	Penulis menggunakan 3300 data tweet tentang sentimen kepada provider telekomunikasi.
4	Data tersebut diklasifikasi secara manual dan dibagi ke dalam masing-masing 1000 data untuk sentimen positif, negatif dan netral.	Data tersebut diklasifikasi secara manual dan dibagi ke dalam masing-masing 1000 data untuk sentimen positif, negatif dan netral.
5	Hasil penelitian ini menghasilkan sebuah sistem yang dapat mengklasifikasi sentimen secara otomatis dengan hasil pengujian 100 tweet mencapai 93% dengan 2700 data training.	Hasil penelitian ini menghasilkan sebuah sistem yang dapat mengklasifikasi sentimen secara otomatis dengan hasil pengujian 100 tweet mencapai 93% dengan 2700 data training.

Gambar 4. 3 Pasangan Kalimat Benar dan Salah

4.3.2 Pre-processing

Penelitian ini melakukan *pre-processing* untuk memastikan bahwa data yang digunakan dalam pelatihan model berada dalam format yang bersih, konsisten, dan siap untuk diproses lebih lanjut oleh algoritma yang digunakan.

4.3.2.1 Record Punctuation

Kalimat	Record Punctuation
Twitter salah satu situs microblogging memungkinkan penggunanya untuk menulis tentang berbagai topik dan membahas isu-isu yang terjadi pada saat ini.	[('!', 20)]

Pada tahap pertama, *record punctuation* diterapkan pada kalimat untuk memeriksa posisi tanda baca pada semua huruf, lalu dicatat posisi tanda baca tersebut bersama posisinya dan dihilangkan sementara dari kalimat.

4.3.2.2 Data Cleaning

Kalimat	Data Cleaning
Twitter salah satu situs microblogging memungkinkan penggunanya untuk	twitter salah satu situs microblogging memungkinkan penggunanya untuk

menulis tentang berbagai topik dan membahas isu-isu yang terjadi pada saat ini	menulis tentang berbagai topik dan membahas isu-isu yang terjadi pada saat ini
--	--

Setelah itu, dilakukan *data cleaning*, yang bertujuan untuk membersihkan teks dari elemen-elemen yang tidak diperlukan atau mengoreksi kesalahan. Misalnya, proses ini menghapus tanda baca yang tidak relevan, mengubah huruf kapital menjadi huruf kecil (*case folding*), menghapus spasi berlebih dan menyederhanakan kata-kata.

4.3.2.3 Tokenization

Kalimat	Tokenization
twitter salah satu situs microblogging memungkinkan penggunaanya untuk menulis tentang berbagai topik dan membahas isu-isu yang terjadi pada saat ini	["twitter", "salah", "satu", "situs", "microblogging", "memungkinkan", "penggunaanya", "untuk", "menulis", "tentang", "berbagai", "topik", "dan", "membahas", "isu-isu", "yang", "terjadi", "pada", "saat", "ini"]

Selanjutnya, dilakukan *tokenization*, yang memecah kalimat menjadi potongan-potongan yang lebih kecil, yang disebut token. Token ini bisa berupa kata atau simbol yang lebih kecil. Proses tokenization ini memisahkan kata-kata sehingga model dapat memproses setiap unit secara terpisah.

4.3.2.4 Word to integer

Kalimat tokenization	Word to Integer
["twitter", "salah", "satu", "situs", "microblogging", "memungkinkan", "penggunaanya", "untuk", "menulis", "tentang", "berbagai", "topik", "dan", "membahas", "isu-isu", "yang", "terjadi", "pada", "saat", "ini"]	{ 'twitter': 45, 'salah': 12, 'satu': 88, 'situs': 67, 'microblogging': 34, 'memungkinkan': 29,

	'penggunanya': 74, 'untuk': 53, 'menulis': 92, 'tentang': 21, 'berbagai': 81, 'topik': 36, 'dan': 11, 'membahas': 60, 'isu-isu': 99, 'yang': 5, 'terjadi': 63, 'pada': 48, 'saat': 17, 'ini': 84 }
--	--

Proses *word to integer* dilakukan, yang mengubah setiap kata yang telah di-tokenize menjadi ID numerik. Ini bertujuan agar model *machine learning* bisa bekerja dengan data dalam bentuk numerik, sehingga model dapat memproses data dalam bentuk angka yang mudah dipahami dan dianalisis.

4.3.2.5 Labelize

<i>Index Numerik</i>	<i>Labelize</i>
{ 'twitter': 45, 'salah': 12, 'satu': 88, 'situs': 67, 'microblogging': 34, 'memungkinkan': 29, 'penggunanya': 74, 'untuk': 53,	[45, 12, 88, 67, 34, 29, 74, 53, 92, 21, 81, 36, 11, 60, 99, 5, 63, 48, 17, 84]

'menulis': 92, 'tentang': 21, 'berbagai': 81, 'topik': 36, 'dan': 11, 'membahas': 60, 'isu-isu': 99, 'yang': 5, 'terjadi': 63, 'pada': 48, 'saat': 17, 'ini': 84 }	
--	--

Pada proses ini mengonversi label teks menjadi representasi numerik yang sesuai dengan kosakata, yang digunakan sebagai pemetaan setiap token label ke indeks.

4.3.2.6 Untokenize

<i>Index</i>	<i>Konversi Index ke Token</i>	<i>Untokenize</i>
[45, 12, 88, 67, 34, 29, 74, 53, 92, 21, 81, 36, 11, 60, 99, 5, 63, 48, 17, 84]	["twitter", "salah", "satu", "situs", "microblogging", "memungkinkan", "penggunanya", "untuk", "menulis", "tentang", "berbagai", "topik", "dan", "membahas", "isu-isu", "yang", "terjadi", "pada", "saat", "ini"]	twitter salah satu situs microblogging memungkinkan penggunanya untuk menulis tentang berbagai topik dan membahas isu-isu yang terjadi pada saat ini

Proses ini memecah index numerik menjadi kalimat dengan mencari kata yang sesuai dengan nomor indexnya.

4.3.2.7 Restore Punctuation

<i>Record Punctuation</i>	<i>Kalimat</i>	<i>Restore Punctuation</i>
[('!', 20)]	twitter salah satu situs microblogging memungkinkan penggunanya untuk menulis tentang berbagai topik dan membahas isu-isu yang terjadi pada saat ini	twitter salah satu situs microblogging memungkinkan penggunanya untuk menulis tentang berbagai topik dan membahas isu-isu yang terjadi pada saat ini.

Mengembalikan posisi tanda baca sesuai dengan posisi yang sudah direkam pada proses *record punctuation*, pada urutan setelah kata ke-20.

Dengan urutan proses ini, data teks yang awalnya tidak terstruktur dan dalam format string dapat diproses menjadi format yang lebih terstruktur dan dapat digunakan oleh algoritma machine learning, baik untuk penghitungan jarak (seperti *Weighted Levenshtein Distance*) atau untuk pelatihan model seperti *Deep Neural Networks*.

4.3.3 Deep Neural Network

Arsitektur model *Deep Neural Network* dirancang berdasarkan desain awal. LSTM digunakan sebagai lapisan input, sementara LSTM dan lapisan dropout berfungsi sebagai lapisan tersembunyi. Selanjutnya, lapisan dense berperan sebagai lapisan output dari model deep neural network.

4.3.4 Train dan Modelling

Untuk melatih model *deep neural network*, pertama-tama perlu ditentukan nilai-nilai *hyperparameter*. *Hyperparameter* yang digunakan dalam penelitian ini untuk melatih model *deep neural network* adalah sebagai berikut:

1. Epoch: 12
2. Batch size: 32

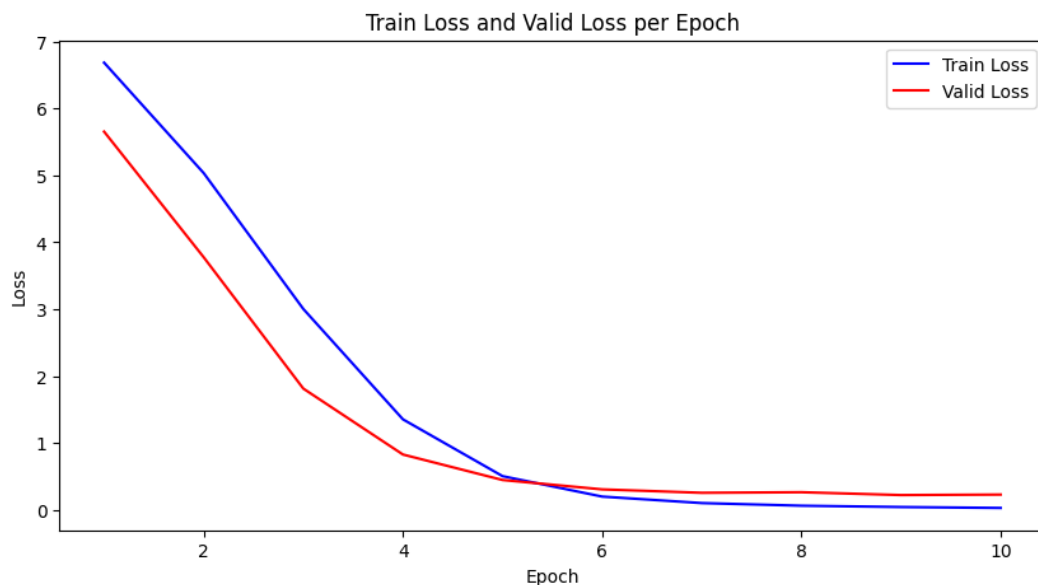
3. Learning rate: 0.005
4. Dropout: 0.5

Pemilihan nilai *learning rate* dan *dropout* didasarkan pada penelitian yang relevan serta melalui eksperimen yang disesuaikan dengan konteks penelitian ini (Jayanthi, 2020; Mammadov, 2020). Nilai *batch size* dipilih karena jumlah dataset yang digunakan adalah 4000 kalimat, sehingga tidak memerlukan *batch size* yang terlalu besar. pengujian nilai epoch ditunjukkan oleh Tabel 4.2.

Tabel 4. 2 Kinerja model per *epoch*

Epoch	Train Loss	Valid Loss
1	6.4636	5.1514
2	4.1305	2.5752
3	1.8351	1.0161
4	0.6278	0.4866
5	0.2101	0.3199
6	0.0940	0.3043
7	0.0568	0.2751
8	0.0349	0.2854
9	0.0344	0.2768
10	0.0340	0.2334
11	0.0322	0.2874
12	0.0287	0.2828

Nilai epoch juga ditentukan berdasarkan hasil percobaan, di mana pada epoch ke-10, nilai *loss* yang diperoleh sudah cukup rendah dan stabil. Dapat dilihat *Train loss* pada epoch ke-10 adalah 0.034, dan valid loss-nya adalah 0.233. Grafik *loss* untuk *train* dan *valid* pada setiap epoch dapat dilihat pada gambar 4.4.



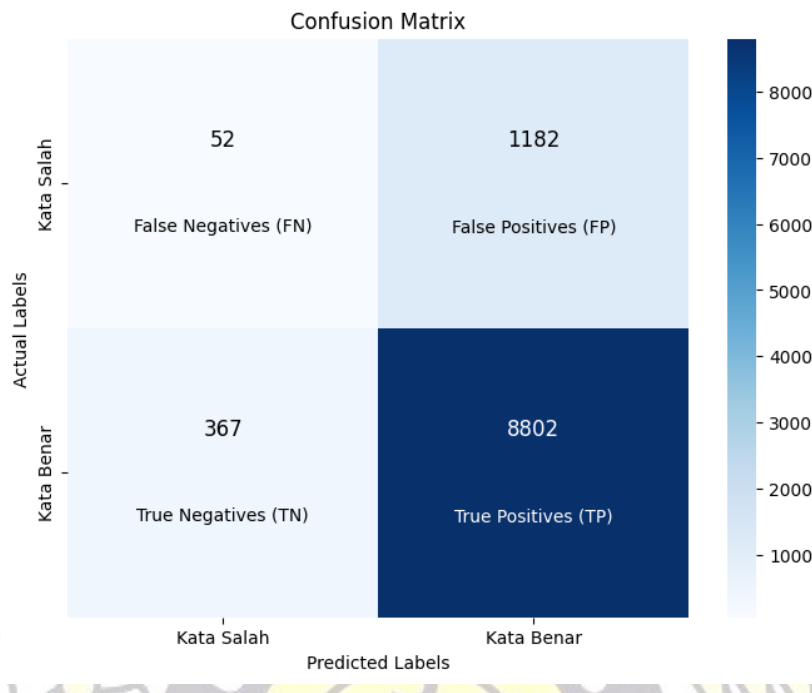
Gambar 4. 4 Grafik *loss* untuk *train* dan *valid* setiap epoch

Output yang dihasilkan berupa grafik yang menampilkan dua garis, yaitu *Train Loss* (berwarna biru) dan *Valid Loss* (berwarna merah). Grafik tersebut menunjukkan penurunan *loss* yang signifikan sepanjang 10 epoch. Penurunan ini mengindikasikan bahwa model yang diterapkan berfungsi dengan baik, karena baik *train loss* maupun *valid loss* terus menurun secara konsisten.

4.3.5 Evaluasi

Pengukuran kinerja model untuk menilai efektivitas model dan sistem yang telah dikembangkan dengan melakukan proses pengujian dan evaluasi. Evaluasi dilakukan dengan membandingkan hasil prediksi terhadap 600 pasangan kalimat dari data uji yang dihasilkan melalui proses pemisahan data (*data splitting*). Proses ini berfokus pada evaluasi kata atau token untuk menilai kemampuan sistem dalam memperbaiki setiap kata. Dengan menggunakan rumus evaluasi yang telah dijelaskan pada tahap analisis dan perancangan, akurasi dan tingkat koreksi kata dari model dan sistem dihitung.

Dalam penelitian ini, evaluasi performa dilakukan dengan memanfaatkan *Confusion Matrix*. Hasil tersebut ditampilkan dalam visualisasi pada Gambar 4.5.



Gambar 4.5 Hasil *Confusion Matrix*

Model *Deep Neural Network* berhasil dilatih, sehingga menghasilkan akurasi sebesar 95.96% dan tingkat koreksi kata sebesar 95.75%. Berikut adalah rincian perhitungan evaluasi untuk model *Deep Neural Network* :

1. *True Positive* (TP) : 8802
2. *True Negative* (TN) : 367
3. *False Positive* (FP) : 1182
4. *False Negative* (FN) : 52
5. *Total* : 10403

Kemudian, data perhitungan tersebut digunakan dalam perhitungan nilai *Accuracy* dan Tingkat koreksi kata dengan memanfaatkan model matematis seperti berikut :

$$\text{Akurasi} = \frac{TP + FP}{\text{total}} \times 100\% = \frac{8802 + 1182}{10403} \times 100\% = \mathbf{95.97\%}$$

$$\text{Tingkat Koreksi kata} = \frac{FP}{FP + FN} \times 100\% = \frac{1182}{1182 + 52} \times 100\% = \mathbf{95.78\%}$$

Evaluasi juga dilakukan terhadap sistem yang telah dikembangkan, di mana proses koreksi tidak hanya memanfaatkan model *Deep Neural Network*, tetapi juga

mengintegrasikan algoritma *Damerau-Levenshtein Distance* sebagai langkah koreksi awal dan validasi akhir. Sistem ini berhasil dirancang dengan menghasilkan akurasi sebesar 98.56% dan tingkat keberhasilan koreksi kata sebesar 94.17%. Rincian hasil evaluasi untuk keseluruhan sistem adalah sebagai berikut:

1. *True Positive* (TP) : 9091
2. *True Negative* (TN) : 77
3. *False Positive* (FP) : 1163
4. *False Negative* (FN) : 72
5. Total : 10403

$$\text{Precision} = \frac{TP}{TP + FP} \times 100\% = \frac{9091}{9091 + 1163} \times 100\% = 88.65\%$$

$$\text{Recall} = \frac{TP}{TP + FN} \times 100\% = \frac{9091}{9091 + 72} \times 100\% = 99.21\%$$

$$\text{F1 Score} = \frac{(2 \times \text{precision} \times \text{recall})}{(\text{precision} + \text{recall})} \times 100\% = \frac{2 \times 0.88 \times 0.99}{(0.88 + 0.99)} \times 100\% = 93.56\%$$

$$\text{Akurasi} = \frac{TP + FP}{\text{total}} \times 100\% = \frac{9091 + 1163}{10403} \times 100\% = 98.56\%$$

$$\text{Tingkat Koreksi kata} = \frac{FP}{FP + FN} \times 100\% = \frac{1163}{1163 + 72} \times 100\% = 94.17\%$$

Berdasarkan pengujian yang dilakukan, diketahui bahwa keterbatasan dataset merupakan salah satu penyebab utama mengapa akurasi dan tingkat koreksi kata pada sistem ini belum mencapai tingkat sempurna, yaitu 100%. Dataset awal pelatihan memiliki cakupan yang terbatas, terutama dalam hal variasi kosakata dan pola konteks, sehingga memengaruhi kemampuan model untuk mendeteksi dan memperbaiki kesalahan ejaan. Selain itu, peningkatan akurasi juga didukung oleh adanya integrasi tambahan koreksi akhir menggunakan algoritma *Damerau-Levenshtein Distance* sebagai validasi akhir, yang memberikan kontribusi signifikan pada kinerja sistem.

Hasil pengujian juga menegaskan bahwa diversifikasi dan perluasan dataset adalah langkah krusial untuk meningkatkan akurasi dan tingkat keberhasilan

koreksi kata. Dengan dataset yang lebih beragam, sistem mampu mengatasi tantangan dalam mendeteksi dan memperbaiki kesalahan ejaan, terutama pada teks berbahasa Indonesia yang memiliki variasi konteks yang kompleks. Diversifikasi ini tidak hanya memperluas cakupan pola yang dapat dikenali oleh model tetapi juga meningkatkan kemampuan sistem dalam memahami dan mengoreksi kesalahan secara lebih akurat.



BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

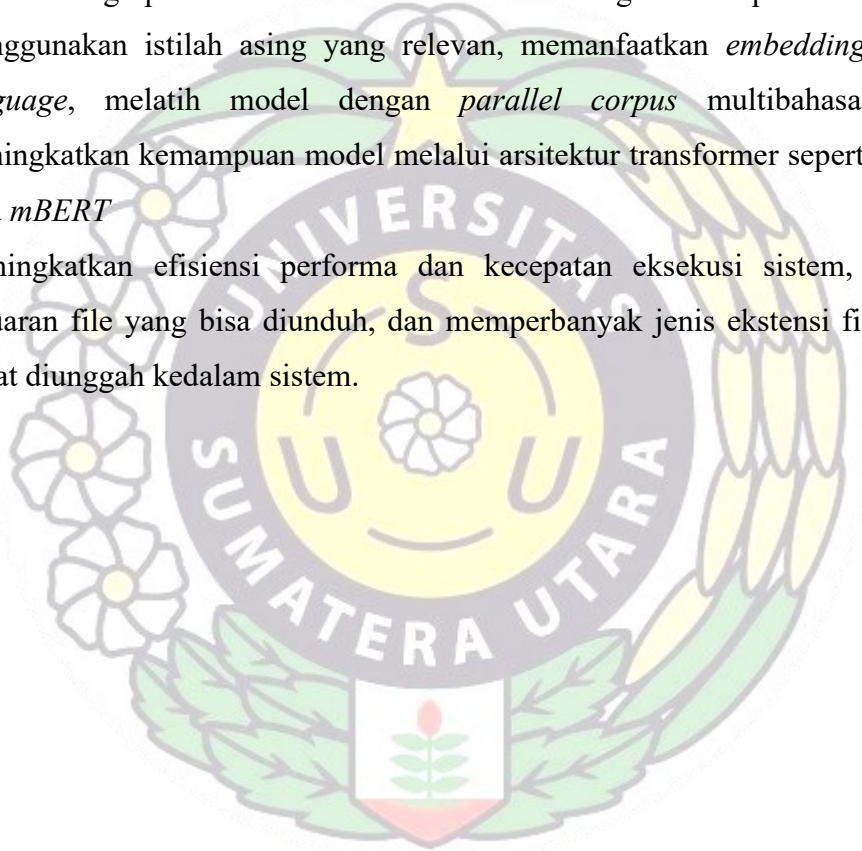
Berdasarkan hasil penelitian dan pengujian yang dilakukan, berikut adalah kesimpulan yang dapat diambil:

1. Identifikasi dan perbaikan *real-word errors* memiliki tingkat kompleksitas yang lebih tinggi dibandingkan dengan *non-word errors*. Hal ini dibuktikan dengan perlunya tiga tahap koreksi: tahap pertama menggunakan algoritma *Damerau-Levenshtein Distance*, tahap kedua menggunakan model *Deep Neural Network*, dan tahap terakhir kembali menggunakan algoritma *Damerau-Levenshtein Distance* sebagai validasi akhir.
2. Sistem koreksi ejaan *real-word errors* dalam bahasa Indonesia yang menggunakan algoritma *Damerau-Levenshtein Distance* dan pendekatan *Deep Neural Network* telah berhasil dikembangkan. Sistem ini mampu mendeteksi dan memperbaiki *real-word errors* dalam teks bahasa Indonesia dengan akurasi sebesar 98.56% dan tingkat keberhasilan koreksi kata sebesar 94.17%.
3. Jumlah data dan luasnya cakupan konteks dalam dataset memiliki pengaruh signifikan terhadap kualitas hasil keluaran sistem yang diharapkan.

5.2 Saran

Berdasarkan hasil penelitian dan pengujian yang telah dilakukan, berikut beberapa saran yang dapat dipertimbangkan untuk pengembangan sistem dan penelitian di masa depan:

1. Meningkatkan jumlah dataset untuk pelatihan. Penambahan dataset dapat meningkatkan performa model *Deep Neural Network*, sehingga menghasilkan output yang lebih akurat.
2. Memastikan kondisi huruf kapital dan tanda baca lainnya tetap diperhatikan sebelum proses case folding dilakukan.
3. Memperluas cakupan konteks dalam dataset dengan menambah variasi konteks kalimat, sistem diharapkan dapat digunakan untuk berbagai kebutuhan.
4. Mengoptimalkan kemampuan sistem dalam mendeteksi dan memperbaiki istilah asing pada teks bahasa Indonesia dengan memperluas dataset menggunakan istilah asing yang relevan, memanfaatkan *embedding multi-language*, melatih model dengan *parallel corpus* multibahasa, serta meningkatkan kemampuan model melalui arsitektur transformer seperti *BERT* atau *mBERT*.
5. Meningkatkan efisiensi performa dan kecepatan eksekusi sistem, seperti keluaran file yang bisa diunduh, dan memperbanyak jenis ekstensi file yang dapat diunggah kedalam sistem.



DAFTAR PUSTAKA

- Adiwidya, Bernardino Madaharsa Dito. (2009). *Algoritma Levenshtein dalam Pendekatan Approximate String Matching*. Bandung: Institut Teknologi Bandung.
- Adinugroho, R. (2022). *Perbandingan Rasio Split Data Training Dan Data Testing Menggunakan Metode Lstm Dalam Memprediksi Harga Indeks Saham Asia* (Bachelor's thesis, Fakultas Sains dan Teknologi UIN Syarif Hidayatullah Jakarta).
- Ahmadzade, A., & Malekzadeh, S. (2021). Spell correction for azerbaijani language using deep neural networks. *arXiv preprint arXiv:2102.03218*.
- Aritomatika, E. J., Sanjaya, A., & Widodo, D. W. (2021, August). Implementasi Algoritma Jaro Winkler Distance Untuk Pendeteksi Kesamaan Kata Dalam Pengembangan Aplikasi English Conversation. In *Prosiding SEMNAS INOTEK (Seminar Nasional Inovasi Teknologi)* (Vol. 5, No. 1, pp. 104-110).
- Arsyta, F. W., & Putra, R. E. (2023). Penerapan Algoritma Damerau Levenshtein Distance Pada Pencarian Arsip Desa Jerukseger Pendukung ISO. *Journal of Informatics and Computer Science (JINACS)*, 4(04), 423-435.
- Aziz, R., Anwar, M. W., Jamal, M. H., Bajwa, U. I., Castilla, Á. K., Rios, C. U., ... & Ashraf, I. (2023). Real Word Spelling Error Detection and Correction for Urdu Language. *IEEE Access*.
- Christanti, V. M., & Naga, D. S. (2018). Fast and accurate spelling correction using trie and Damerau-levenshtein distance bigram. *Telkomnika*, 16(2), 827-833.
- Fu'adi, F. (2015). Implementasi Perintah Menampilkan Data Menggunakan Bahasa Indonesia Dengan Natural Language Processing. *Proceeding of KMICE*, 8, 1-8.

- Hamidah, N., Yusliani, N., & Rodiah, D. (2020). Spelling Checker using Algorithm Damerau Levenshtein Distance and Cosine Similarity. *Sriwijaya Journal of Informatics and Applications*, 1(1).
- Herpindo, H., Astuty, A., Ekawati, M., Arvianti, G. F., Nikmatullah, M. R., & Afiq, M. N. (2023). Pembelajaran Dan Pengajaran Tata Bahasa Berdasarkan Korpus. *Risenologi*, 8(2), 25-37.
- Huang, G., Chen, J., & Sun, Z. (2020, December). A correction method of word spelling mistake for English text. In *Journal of Physics: Conference Series* (Vol. 1693, No. 1, p. 012118). IOP Publishing.
- Kusuma, A. T. A., & Ratnasari, C. I. (2023). Comparison Of Spell Correction In Bahasa Indonesia: Peter Norvig, LSTM, And N-Gram. *JIKO (Jurnal Informatika dan Komputer)*, 6(3), 214-220.
- Lee, J. H., Kim, M., & Kwon, H. C. (2020). Deep learning-based context-sensitive spelling typing error correction. *IEEE Access*, 8, 152565-152578.
- Londo, G. L. Y., Purnomo WP, Y. S., & Maslim, M. (2020). Pembangunan Aplikasi Identifikasi Kesalahan Ketik Dokumen Berbahasa Indonesia Menggunakan Algoritma Jaro-Winkler Distance. *Jurnal Informatika Juita*, 8(1), 19-27.
- Marpaung, Sari Wahyuni. (2019). Analisis Akurasi Saran Perbaikan Kata pada Dokumen Berbahasa Indonesia dengan Algoritma Levenshtein Sesuai Topik Dokumen Menggunakan Metode Term Weighting. Medan: Universitas Sumatera Utara.
- Naing, I., Aung, S. T., Wai, K. H., & Funabiki, N. (2024). A Reference Paper Collection System Using Web Scraping. *Electronics*, 13(14), 2700.
- Samanta, P., & Chaudhuri, B. B. (2013, October). A simple real-word error detection and correction using local word bigram and trigram. In *Proceedings of the 25th conference on computational linguistics and speech processing (ROCLING 2013)* (pp. 211-220).

- Santoso, P., Yuliawati, P., Shalahuddin, R., & Wibawa, A. P. (2019). Damerau levenshtein distance for indonesian spelling correction. *J. Inform*, 13(2), 11.
- Sujaini, H., Muhardi, H., & Simanjuntak, J. H. Aplikasi Pengoreksi Ejaan (Spelling Correction) pada Naskah Jurnal Bidang Informatika dengan N-Gram dan Jaro-Winkler Distance. *JEPIN (Jurnal Edukasi dan Penelitian Informatika)*, 8(2), 235-244.

