DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Proses pembelajaran yang efektif tidak dapat dilepaskan dari mekanisme evaluasi yang baik. Salah satu komponen penting dalam evaluasi adalah penilaian terhadap kemampuan siswa dalam menjawab soal esai, terutama pada mata pelajaran yang bersifat subjektif seperti Ilmu Pengetahuan Sosial (IPS). Selama ini, penilaian esai umumnya dilakukan secara manual oleh guru, yang membutuhkan estimasi tenaga dari guru dan waktu untuk mengoreksi. Selain itu, penilaian manual juga rentan terhadap subjektivitas, sehingga hasil penilaian dapat bervariasi antar satu penilai dengan penilai lainnya.

Ketidaksesuaian antara tuntutan zaman akan efisiensi dan efektivitas proses pembelajaran yang keterbatasan dalam sistem penilaian esai manual menjadi alasan utama pemilihan masalah ini sebagai objek penelitian. Dengan seiring berkembangnya watu dan jaman teknologi serta informasi komunikasi, khususnya dalam bidang *Artificial Intelligence (AI)*, muncul harapan untuk dapat mengembangkan sistem penilaian yang lebih objektif, efisien, dan akurat.

Metode *Recurrent Neural Network (RNN)* dengan *Long Short Term Memory (LSTM)* adalah salah satu model *deep learning* yang efektif untuk tugas klasifikasi sentimen. Model ini mampu memproses data secara berurutan, seperti teks, audio, dan video. Konsep ini didasarkan pada kemampuan model *MaLSTM* dan *LSTM* dalam memproses data teks secara berurutan dan menangkap konteks dalam sebuah kalimat atau paragraf.(Cahyadi et al., 2020)

Meskipun telah ada beberapa penelitian yang membahas tentang pemanfaatan AI dalam bidang pendidikan, namun penelitian yang fokus pada pengembangan sistem penilaian otomatis untuk soal esai IPS masih relatif terbatas. Kebanyakan penelitian sejenis lebih banyak berfokus pada bahasa Inggris atau mata pelajaran yang bersifat eksakta. Oleh karena itu, terdapat gap antara kebutuhan akan sistem penilaian yang lebih efisien dan objektif dengan ketersediaan solusi yang sesuai untuk konteks pendidikan di Indonesia, khususnya pada mata pelajaran IPS.

Dalam penilaian soal esai, guru biasanya harus mengoreksi jawaban secara manual,

satu per satu. Untuk mempermudah dan mempercept proses ini, dapat diterapkan algoritma dalam sebuah sistem penilaian otomatis. Algoritma tersebut bekerja dengan membandingkan teks jawaban siswa dengan kunci jawaban siswa dengan kunci jawaban yang telah disediakan. Berdasarkan penjelasan tersebut, sistem evaluasi otomatis dapat didefinisikan debagai metode untuk menilai hasil dan memastikan keakuratan dengan menggunakan prinsip-prinsip teknologi dalam pengembangannya.(Salim et al., 2023)

Dari penjelesan di atas, dapat disimpulkan bahwa penelitian yang akan dilakukan ini sangat relevan dengan permasalahan yang teradi lapangan. Pengembangan sistem penilaian otomatis untuk soal esai ips diharapkan mampus memberikan kontribusi besar dalam meningkatkan kualitas Pendidikan, terutama dalam aspek evaluasi pembelajaran.

1.2. Rumusan Masalah

1. Bagaimana cara mengimplementasikan *Long Short-Term Memory (LSTM)* dan *Manhattan Long Hort-Term Memory* pada sistem Penilaian Otomatis untuk Soal Esai Mata Pelajaran Ilmu Pengetahuan Sosial?
2. Bagaimana hasil performa model *Long Short-Term Memory (LSTM)* dan *Manhattan Long Short-Term Memory* untuk mengevaluasi akurasinya dalam menilai jawaban esai siswa secara otomatis?

1.3. Tujuan Penelitian

1. Mengimplementasikan model *Long Short-Term Memory (LSTM)* dan *Manhattan Long Short-Term Memory* untuk membangun sistem Penilaian Otomatis pada Soal Esai Mata Pelajaran Ilmu Pengetahuan Sosial, sehingga mampu memahami hubungan antar kata dalam teks jawaban siswa secara efektif.
2. Melakukan pengujian performa model *Long Short-Term Memory (LSTM)* dan *Manhattan Long Short-Term Memory* dengan menggunakan berbagai metrik evaluasi, seperti akurasi, *precision, dan recall,* untuk menilai keandalan model dalam memberikan penilaian otomatis terhadap jawaban esai siswa.

1.4. Batasan dan Asumsi Penelitian

1. **Jenis soal esai:** Penelitian ini akan fokus pada soal esai yang bersifat terbuka, dimana siswa diminta untuk memberikan pendapat atau argumen.
2. **Mata pelajaran:** Penelitian ini akan dibatasi pada mata pelajaran IPS.

3. **Tingkat pendidikan:** Penelitian ini akan dilakukan pada tingkat SMP.
4. **Bahasa:** Sistem akan dilatih dan diuji pada bahasa Indonesia.
5. **Jumlah data:** Jumlah data latih yang digunakan akan disesuaikan dengan ketersediaan data.
6. **Perangkat terbatas :** Hanya dapat dijalankan di browser.
7. **Penilaian :** Tidak mempertimbangkan bobot nilai siswa per jawaban. DONE
8. **Sistem** : Tidak sepenuhnya memahami konteks atau makna implisit dalam jawaban, sehingga berpotensi salah menilai jika struktur kalimat berbeda dari pola pelatihan.

1.5. Manfaat Penelitian

1. Efisiensi bagi Guru: Mengurangi beban kerja guru dalam memeriksa soal esai secara manual, sehingga mereka dapat fokus pada kegiatan pengajaran lainnya.
2. Keakuratan Penilaian: Meningkatkan keakuratan dan konsistensi penilaian dibandingkan dengan koreksi manual yang bisa subjektif.
3. Kemudahan bagi Sekolah: Membantu sekolah memiliki sistem penilaian otomatis yang mendukung pengelolaan bank soal esai secara digital.
4. Pengembangan Pendidikan Berbasis Teknologi: Mendorong penggunaan teknologi dalam proses pendidikan, sejalan dengan konsep pendidikan berbasis digital.

1.6. Sistematika Penulisan

Rencana untuk sistematika dari penulisan proposal penelitian ini akan disusun dengan rincian sebagai berikut:

BAB I PENDAHULUAN
Bab ini berisi penjelasan mengenai latar belakang permasalahan yang melandasi dilaksanakannya penelitian, rumusan masalah yang ingin diselesaikan, tujuan dari penelitian yang ingin dicapai, serta batasan masalah dan asumsi yang digunakan untuk menjaga ruang lingkup penelitian tetap fokus. Selain itu, bab ini juga mencakup manfaat dari penelitian baik secara teoritis maupun praktis, serta sistematika penulisan sebagai panduan keseluruhan dokumen.

BAB II LANDASAN TEORI
Bab ini memuat teori-teori dan literatur yang relevan sebagai dasar dalam membangun penelitian. Materi yang dibahas mencakup konsep soal esai dalam pendidikan, dasar-dasar Natural Language Processing (NLP), teori Long Short-Term Memory (LSTM) dan varian Manhattan LSTM (MaLSTM), metode pengukuran kemiripan teks (*text similarity*), serta pembahasan mengenai penelitian terdahulu yang berkaitan dengan sistem penilaian otomatis berbasis teks.

BAB III METODOLOGI PENELITIAN

Bab ini menjelaskan secara rinci tahapan pelaksanaan penelitian, mulai dari studi literatur, pengumpulan dan pemrosesan dataset soal esai, hingga proses perancangan serta implementasi model LSTM dan MaLSTM. Selain itu, dijelaskan pula proses *preprocessing* data (tokenisasi, *stopword removal*, dan *stemming*), perancangan arsitektur model, skenario pengujian, serta evaluasi performa model menggunakan metrik seperti akurasi, precision, recall, dan f1-score. Jadwal pelaksanaan penelitian juga disertakan dalam bentuk tabel guna memberikan gambaran runtut terkait waktu pengerjaan.

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisi hasil implementasi dan pengujian model yang telah dikembangkan, baik model LSTM Classifier maupun MaLSTM. Hasil yang ditampilkan mencakup analisis performa model berdasarkan metrik evaluasi, visualisasi seperti confusion matrix, histogram skor similarity, dan hasil thresholding. Selain itu, disertakan pula pembahasan mendalam terkait pengaruh parameter model (unit memori, jumlah layer, dan *learning rate*), rasio data latih-uji, serta dampak preprocessing terhadap kinerja model. Interpretasi hasil dilakukan dengan mengaitkan temuan dengan tujuan penelitian serta kelebihan dan keterbatasan sistem.

DAFTAR PUSTAKA

Bab ini memuat seluruh referensi dan sumber pustaka yang digunakan selama proses penyusunan proposal, baik berupa buku, jurnal, artikel ilmiah, maupun sumber digital lainnya, sesuai dengan standar penulisan ilmiah yang berlaku.

BAB II
TINJAUAN PUSTAKA

2.1. Penelitian Terkait

Tabel II.1 Penelitian Terkait Jurnal 1

Judul Jurnal	Penulis	Metode	Hasil Penelitian	Keterkaitan
Implementasi Metode *Recurrent Neural Network* pada Text Summarizatio n dengan Teknik Abstrakti	Kasyfi Ivanedra, Metty Mustikasar i (2020)	*RNN*, *LSTM*, Attention Mechanis m	Peringkasan teks abstraktif dengan *LSTM* dan Attention meningkatka n F-Measure dari 54,27% ke 58,20%.	Memberika n dasar penggunaan dan *LSTM* untuk tugas-tugas *NLP* seperti peringkasan teks

Pada *Tabel II. 1* berjudul "Implementasi metode *Recurrent Neural Network* pada *Text Summarization* dengan Teknik Abstrakti" oleh Kasyfi Ivanedra dan Metty Mustikasari (2020) menggunakan metode *RNN* dan *LSTM* untuk menghasilkan model peringkasan teks abstraktif yang dilengkapi dengan Attention, sehingga mampu meningkatkan nilai F-Measure dari 54,27% menjadi 58,20%. Penelitian ini memiliki keterkaitan dengan penelitian yang akan dilakukan, yaitu penggunaan metode *RNN* dan *LSTM* untuk penilaian otomatis soal esai mata pelajaran IPS pada tingkat SMP. Namun, penelitian ini memiliki kekurangan karena berfokus pada peringkasan teks berita, yang kurang relevan untuk esai dengan variasi struktur kalimat yang menjadi fokus penelitian soal esai IPS. Selain itu, hasil ringkasan dalam penelitian tersebut berupa gabungan kalimat-kalimat terpilih berdasarkan kemunculan yang paling sering. Di sisi lain, kelebihan penelitian ini adalah penggunaan kombinasi Attention dan *LSTM* yang memberikan hasil peringkasan lebih natural. Hal ini menunjukkan bahwa metode *RNN* dengan *LSTM* berpotensi menghasilkan model yang efektif untuk tugas serupa, termasuk dalam penelitian penilaian otomatis soal esai IPS.(Ivanedra & Mustikasari, 2019)

Tabel II.2 Penelitian Terkait Jurnal 2

Judul Jurnal	Penulis	Metode	Hasil Penelitian	Keterkaitan
Penilaian Otomatis Jawaban Tes Uraian Lisan Menggunakan Metode *Recurrent Neural Network*	Philip Antoni Siahaan (2021)	*RNN, Automated Speech Recognition (ASR)*	Evaluasi berbasis regresi menggunakan *RNN* menghasilkan nilai Quadratic Weighted Kappa (QWK) sebesar 0,971.	Relevan dalam menerapkan *RNN* untuk penilaian *otomatis* esai berbasis teks yang berasal dari hasil konversi suara.

Pada tabel *II. 2* penelitian ini dengan judul Penilaian Otomatis Jawaban Tes Uraian Lisan Menggunakan Metode *Recurrent Neural Network*, Philip Antoni Siahaan (2021) menggunakan methode *RNN* dan *Automated Speech Recognition (ASR)* dengan hasil penelitian Evaluasi berbasis regresi menggunakan *RNN* menghasilkan nilai *Quadratic Weighted Kappa (QWK)* sebesar 0,971. pada penelitian ini memiliki keterkaitan dengan penelitian yang sedang dilakukan yaitu Relevan dalam menerapkan *RNN* untuk penilaian otomatis esai berbasis teks yang berasal dari hasil konversi suara dengan perbedaannya memberikan penilaian otomatis pada mata pelajaran soal essai IPS jenjang SMP menggunakan *text similarity* dari jawaban siswa dengn guru. Pada penelitian ini juga memiliki kekurangan dalam penelitian nya yaitu fokus pada jawaban berbahasa Inggris dengan dataset tertentu, sehingga perlu adaptasi untuk bahasa lain. Dengan kelebihan dari methode yang digunakan mencakup pengolahan suara ke teks *(ASR)* hingga penilaian otomatis berbasis teks dengan performa akurasi tinggi sangat menguntungkan bagi penelitian ini karena mungkin apkurasi yang dihasilkan dari methode text to text lebih akurat dari suara to text.(Siahaan, 2021)

Tabel II.3 Penelitian Terkait Jurnal 3

Judul Jurnal	Penulis	Metode	Hasil Penelitian	Keterkaitan
Implementation of Recurrent Neural Network (RNN) for Question Similarity Identification in Indonesian Language	Muhammad Iqbal, Hasmawati, Ade Romadhony (2023)	*RNN, Manhattan Distance, Word2Vec*	*RNN* dengan pendekatan Manhattan Distance mencapai akurasi 76%, precision 74,7%, recall 98,8%, F1-score 85,1%.	Menunjukk an efektivitas *RNN* untuk pengukuran kesamaan teks dalam bahasa Indonesia.

Pada tabel *II. 3* penelitian ini dengan judul *Implementation of Recurrent Neural Network (RNN) for Question Similarity Identification in Indonesian Language*, Muhammad Iqbal, Hasmawati, Ade Romadhony (2023) dengan methode *RNN*, *Manhattan Distance, Word2Vec* dengan hasil dari penelitian adalah *RNN* dengan pendekatan *Manhattan Distance* mencapai akurasi 76%, precision 74,7%, recall 98,8%, F1-score 85,1%. penelitian ini memeiliki keterkaitan dengan penelitian yang sedang dilakukan menunjukkan efektivitas *RNN* untuk pengukuran kesamaan teks dalam bahasa Indonesia karena penilaian otomatis pada soal essai dilakukan juga dalam bahasa indonesia dalam bentuk text yang terstruktur. penelitian ini juga memiliki kekurangan yaitu dataset terbatas (621 pasang soal), sehingga hasil penelitian kurang dapat digeneralisasi untuk data berskala besar dengan hal ini mungkin pengembangan dalpam penelitian ini hanya kurang di bagian data yang di penelitian yang sedang dilakukan akan menggunakan data sejumlah 1000 soal mata pelajaran IPS pada jenjang SMP. Kelebihan dari penelitian ini adalah menggunakan dataset bahasa Indonesia dengan proses *preprocessing* yang efisien seperti stopword removal yang dengan penelitian sedang berlangsung adanya stopword

untuk penilaian otomatis pada soal essai sangat bergna dan membantu sistem dalam penilaian berlangsung.(Iqbal et al., 2023)

Tabel II.4 Penelitian Terkait Jurnal 4

Judul Jurnal	Penulis	Metode	Hasil Penelitian	Keterkaitan
Implementasi *Long-Short Term Memory (LSTM)* untuk Generasi Feedback Berbahasa Indonesia pada Sistem Penilaian Esai	Delvia Lana Semba	*LSTM, Automated Essay Scoring (AES)*	LSTM digunakan untuk menghasilkan skor otomatis (1–4) dan kalimat umpan balik. Akurasi model AES mencapai 94%, umpan balik 87%.	Menjelaskan cara pengembangan sistem penilaian esai berbasis *LSTM* untuk bahasa Indonesia dengan feedback otomatis.

Pada tabel *II.4* penelitian ini dengan judul Implementasi *Long-Short Term Memory (LSTM)* untuk Generasi Feedback Berbahasa Indonesia pada Sistem Penilaian Esai, Delvia Lana Semba menggunakan metode *LSTM* dan *Automated Essay Scoring (AES)* untuk menghasilkan skor otomatis dan umpan balik berbasis teks. Hasil penelitian menunjukkan bahwa model *LSTM* berhasil memberikan skor otomatis dengan akurasi mencapai 94% dan tingkat keberhasilan dalam menghasilkan kalimat umpan balik sebesar 87%. Kelebihan dari penelitian ini adalah integrasi antara sistem penilaian otomatis dan generasi umpan balik, yang menjadikannya relevan untuk studi terkait otomatisasi evaluasi esai berbahasa Indonesia. Namun, penelitian ini juga memiliki kekurangan, yaitu skor BLEU untuk kualitas kalimat umpan balik masih rendah (<0.3), sehingga kualitas umpan balik yang dihasilkan belum optimal dibandingkan dengan umpan balik manusia. Penelitian ini

mendukung studi yang sedang dilakukan karena menawarkan pendekatan berbasis *LSTM* yang relevan untuk tugas penilaian esai otomatis. (Lana Semba, 2024)

Tabel II.5 Penelitian Terkait Jurnal 5

Judul Jurnal	Penulis	Metode	Hasil Penelitian	Keterkaitan
Sentimental Analysis using Recurrent Neural NetworkImplementasi Library Deep Learning Keras pada Sistem Ujian Essay Online	Ekojono, Faisal Rahutomo, Dhiana Novita Sari (2020)	*Siamese LSTM, Keras*	Menggunakan Siamese *LSTM* untuk menghitung kemiripan teks pada ujian esai daring. Tingkat error pada kategori soal politik: 61%, lifestyle: 112%, olahraga: 162%, teknologi: 175%.	Memberika n penerapan langsung Siamese *LSTM* untuk sistem penilaian esai daring berbasis Bahasa Indonesia.

Pada tabel *II.5* penelitian ini dengan judul Sentimental *Analysis using Recurrent Neural NetworkImplementasi Library Deep Learning Keras* pada Sistem Ujian Essay Online, Ekojono, Faisal Rahutomo, Dhiana Novita Sari (2020) dengan methode Siamese *LSTM*, Keras dengan hasil penelitian menggunakan *Siamese LSTM* untuk menghitung kemiripan teks pada ujian esai daring. Tingkat error pada kategori soal politik: 61%, lifestyle: 112%, olahraga: 162%, teknologi: 175%. Pada penelitian ini keterkaitan dengan penelitian yang akan dilakukan adalah memberikan penerapan langsung *Siamese LSTM* untuk sistem penilaian esai daring berbasis Bahasa Indonesia. Penelitian ini juga memiliki kekurangan hasil error masih cukup tinggi dibandingkan metode lain, menunjukkan perlunya pengolahan

dataset lebih baik dengan klasifikasi yang lebih teliti. Penelitian ini juga memiliki kelebihan menggunakan pendekatan yang mendalam dengan *Siamese LSTM* untuk penilaian otomatis pada teks berbahasa Indonesia yang berhubungan dengan penelitian yang sedang berlangsung dengan hal yang lebih menjurus yaitu mata pelajaran IPS pada jenjang SMP yang dimana sama-sama berbahasa Indonesia(Yosia Wibowo et al., 2024)

Tabel II.6 Penelitian Terkait Jurnal 6

Judul Jurnal	Penulis	Metode	Hasil Penelitian	Keterkaitan
Stance Classification Pada Berita Berbahasa Indonesia Berbasis Bidirectional *LSTM*	Esther Irawati Setiawan, Ika Lestari (2022)	*BiLSTM, GRU, FastText Embedding*	Menggunakan Bi*LSTM* dan GRU untuk stance classification. Akurasi F1-Score tertinggi mencapai 64% dengan FastText embedding.	Menunjukkan efektivitas Bi*LSTM* dan GRU dalam klasifikasi sikap untuk berita berbahasa Indonesia.

Pada tabel *II.6* Penelitian berjudul "*Stance Classification Pada Berita Berbahasa Indonesia Berbasis Bidirectional LSTM*" oleh Esther Irawati Setiawan dan Ika Lestari (2022) menggunakan metode *BiLSTM, GRU*, dan *FastText Embedding* untuk melakukan klasifikasi sikap (*stance classification*) pada berita berbahasa Indonesia. Hasil penelitian menunjukkan bahwa model Bi*LSTM* dan GRU memiliki kinerja yang sebanding, dengan F1-score tertinggi mencapai 64% saat menggunakan FastText sebagai word embedding. Penelitian ini relevan dengan penelitian yang akan dilakukan karena keduanya menggunakan pendekatan deep learning berbasis *LSTM* untuk analisis teks dalam bahasa Indonesia. Salah satu keunggulan penelitian ini adalah kemampuan FastText dalam menangani masalah *out-of-vocabulary (OOV)* yang sering terjadi pada *NLP,* serta performa konsisten dari *BiLSTM dan GRU*. Namun, penelitian ini memiliki kelemahan berupa akurasi

yang masih terbatas pada 64%, sehingga diperlukan peningkatan lebih lanjut, misalnya dengan memanfaatkan embedding yang lebih canggih seperti *BERT* untuk meningkatkan kinerja model. (Esther Irawati Setiawan & Ika Lestari, 2022)

Tabel II.7 Penelitian Terkait Jurnal 7

Judul Jurnal	Penulis	Metode	Hasil Penelitian	Keterkaitan
A Hybrid Approach of Weighted Fine-Tuned BERT Extraction with Deep Siamese Bi-LSTM Model for Semantic Text Similarity Identification	D. Viji, S. Revathy (2022)	*Fine-Tuned BERT, Siamese Bi-LSTM*	Kombinasi BERT Fine-Tuning dan Siamese Bi-*LSTM* mencapai akurasi 91% pada dataset Quora Question Pairs.	Memberikan pendekatan kombinasi deep learning untuk mengidentifikasi kesamaan teks semantik.

Pada tabel *II. 7* penelitian ini dengan judul *A Hybrid Approach of Weighted Fine-Tuned BERT Extraction with Deep Siamese Bi-LSTM Model for Semantic Text Similarity Identification*, D. Viji dan S. Revathy (2022) mengimplementasikan pendekatan BERT Fine-Tuning yang dikombinasikan dengan *Siamese Bi-LSTM* untuk mengidentifikasi kesamaan teks semantik. Penelitian ini menggunakan dataset dari Quora Question Pairs, di mana fitur teks diekstraksi menggunakan *BERT*, kemudian diproses melalui model *Siamese Bi-LSTM* untuk menghitung skor kesamaan semantik. Hasil penelitian menunjukkan akurasi 91% dalam prediksi kesamaan teks, yang lebih unggul dibandingkan metode seperti *CNN*, *Shingling*, dan *MLP*. Kelebihan dari penelitian ini adalah penggunaan *BERT* untuk menangkap fitur kontekstual yang lebih mendalam serta *Siamese Bi-LSTM* yang efektif dalam menangani teks dengan hubungan semantik. Namun, penelitian ini masih memiliki

kekurangan, seperti peningkatan waktu komputasi ketika ukuran dataset meningkat, meskipun eksekusi pada file berukuran kecil cukup efisien. Penelitian ini relevan dengan studi yang akan dilakukan karena menawarkan pendekatan kombinasi deep learning untuk penilaian teks, yang dapat diadaptasi dalam sistem penilaian otomatis soal esai berbahasa Indonesia(Viji & Revathy, 2022)

Tabel II.8 Penelitian Terkait Jurnal 8

Judul Jurnal	Penulis	Metode	Hasil Penelitian	Keterkaitan
Deteksi Pertanyaan Serupa di Quora Menggunakan *CNN dan LSTM*	Mansoor, M., Rehman, Z. U., Shaheen, M., Khan, M. A., Habib, M. (2020)	*CNN, LSTM, Word2Vec*	Model mencapai akurasi 87.50%, dengan precision 83.15, recall 87.02, dan F1 score 85.04.	Menunjukkan efektivitas kombinasi CNN dan *LSTM* dalam mendeteksi kesamaan pertanyaan di Quora. | Menggunakan dataset besar yang relevan, serta kombinasi model yang efektif untuk menangkap fitur lokal dan ketergantungan jangka panjang.

Pada tabel *II. 8* penelitian ini dengan judul Deteksi Pertanyaan Serupa di Quora Menggunakan *CNN* dan *LSTM*, Mansoor, M., Rehman, Z. U., Shaheen, M., Khan, M. A., dan Habib, M. (2020) mengimplementasikan metode *CNN, LSTM*, dan *Word2Vec* untuk mendeteksi kesamaan pertanyaan di platform Quora. Hasil penelitian menunjukkan bahwa kombinasi *CNN* dan *LSTM* mampu mencapai

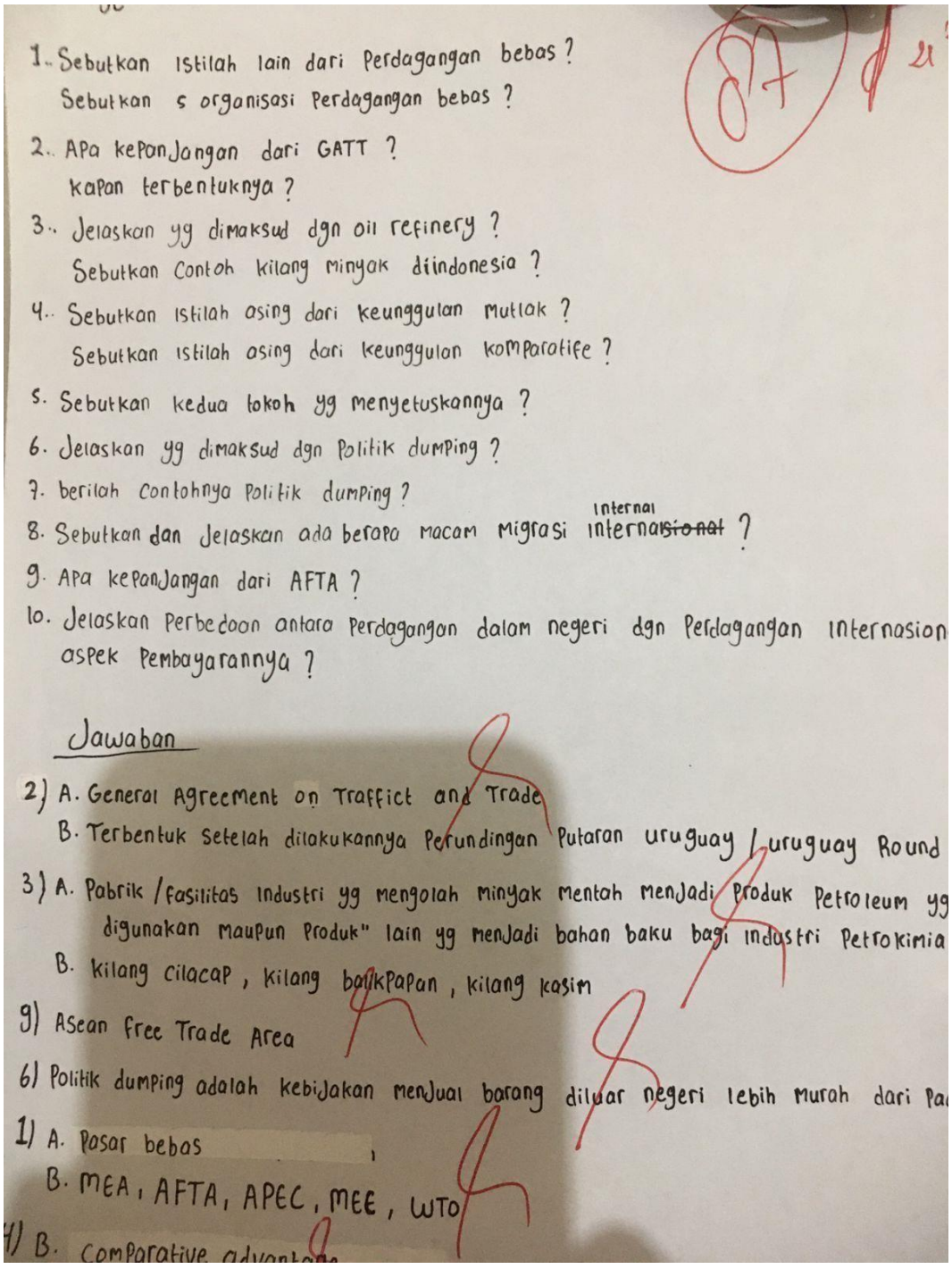
akurasi sebesar 87.50%, dengan precision 83.15, recall 87.02, dan F1-score 85.04. Kelebihan dari penelitian ini adalah penggunaan dataset besar yang relevan dan kombinasi model *CNN* serta *LSTM* yang efektif dalam menangkap fitur lokal serta ketergantungan jangka panjang dalam teks. Namun, penelitian ini memiliki kekurangan yaitu hasil mungkin tidak dapat digeneralisasi ke dataset lain tanpa penyesuaian lebih lanjut, serta tidak adanya analisis mendalam mengenai metrik evaluasi yang digunakan. Penelitian ini relevan dengan studi yang akan dilakukan karena menunjukkan efektivitas penggunaan *CNN* dan *LSTM* dalam mengidentifikasi kesamaan teks, yang dapat diadaptasi untuk penilaian otomatis pada soal esai berbasis teks.(Mansoor et al., 2020)

2.2. Dasar Teori

2.1.1. *Penilaian Otomatis*

Automated Essay Scoring (AES) merupakan sebuah proses penilaian yang dilakukan terhadap tes uraian tanpa adanya campur tangan dari manusia. Dalam sistem ini, *AES* mengolah masukan dari jawaban yang diberikan oleh peserta dan memberikan skor berdasarkan beberapa kriteria, seperti kemiripan konten, susunan kata, serta arti kata terhadap jawaban awal yang telah ditentukan oleh pemberi soal. Implementasi *AES* menjadi solusi inovatif dalam sistem penilaian, karena tidak hanya mengurangi penggunaan kertas, tetapi juga efisiensi waktu dalam menilai setiap jawaban. Salah satu ilmu yang berperan penting dalam pengembangan *AES* adalah *Natural Language Processing (NLP),* yang memungkinkan analisis dan pemrosesan masukan berbasis teks atau dokumen secara otomatis. Dengan demikian, *AES* dan *NLP* saling melengkapi dalam menciptakan sistem penilaian yang lebih cepat, akurat, dan efisien. (Siahaan, 2021)

2.1.2. *Soal Esai*



1. Sebutkan istilah lain dari perdagangan bebas ?
   Sebutkan 5 organisasi perdagangan bebas ?
2. Apa kepanjangan dari GATT ?
   kapan terbentuknya ?
3. Jelaskan yg dimaksud dgn oil refinery ?
   Sebutkan contoh kilang minyak diindonesia ?
4. Sebutkan istilah asing dari keunggulan mutlak ?
   Sebutkan istilah asing dari keunggulan komparatife ?
5. Sebutkan kedua tokoh yg menyetuskannya ?
6. Jelaskan yg dimaksud dgn politik dumping ?
7. berilah contohnya politik dumping ?
8. Sebutkan dan jelaskan ada berapa macam migrasi internasional ~~sional~~ ?
9. Apa kepanjangan dari AFTA ?
10. Jelaskan perbedaan antara perdagangan dalam negeri dgn perdagangan internasion aspek pembayarannya ?

Jawaban

2) A. General Agreement on Traffict and Trade
   B. Terbentuk setelah dilakukannya perundingan putaran uruguay / uruguay Round
3) A. Pabrik / fasilitas industri yg mengolah minyak mentah menjadi produk petroleum yg digunakan maupun produk" lain yg menjadi bahan baku bagi industri petrokimia
   B. kilang cilacap, kilang balikpapan, kilang kasim
9) Asean free Trade Area
6) Politik dumping adalah kebijakan menjual barang diluar negeri lebih murah dari pa
1) A. pasar bebas
   B. MEA, AFTA, APEC, MEE, WTO
4) B. comparative advanta

Gambar II.1 Contoh Soal Esai Siswa

Pada gambar II.1 terdapat contoh dari soal esai siswa dengan mata Pelajaran IPS. Soal esai merupakan salah satu metode evaluasi yang efektif untuk mengukur kemampuan siswa. Menurut Sukardi (2009), soal essay mengandung permasalahan yang menuntut siswa untuk memberikan jawaban dalam bentuk uraian yang

mencerminkan kemampuan berpikir mereka. Berbeda dengan soal pilihan ganda, soal essay menggunakan pertanyaan terbuka yang memun.gkinkan siswa menjawab sesuai dengan pengetahuan dan pemahaman yang dimiliki. Penggunaan soal essay sangat relevan dalam mengukur kemampuan berpikir kritis siswa.

Dirman dan Juarsih (2014: 57) mengidentifikasi beberapa kelebihan dari soal essay, antara lain: siswa tidak dapat mengkordinisir seperti apa jawaban yang diinginkan atau jawaban yang persis seperti kunci jawaban milik guru, soal esai sangat cocok untuk mengukur seberapa kemampuan siswa dalam memahami mata Pelajaran tersebut, dan (e) Soal essay dapat melatih siswa untuk lebih kritis dan memilih fakta yang relevan dengan permasalahan yang soal esai sajikan. Dengan demikian, soal esai tidak hanya sebagai wadah untuk guru menilai kemampuan siswa dalam memberikan jawaban tapi melatih siswa lebih kritis dan terbuka atas argumen kebebasan berpendapat sesuai fakta.(Handayani et al., 2021)

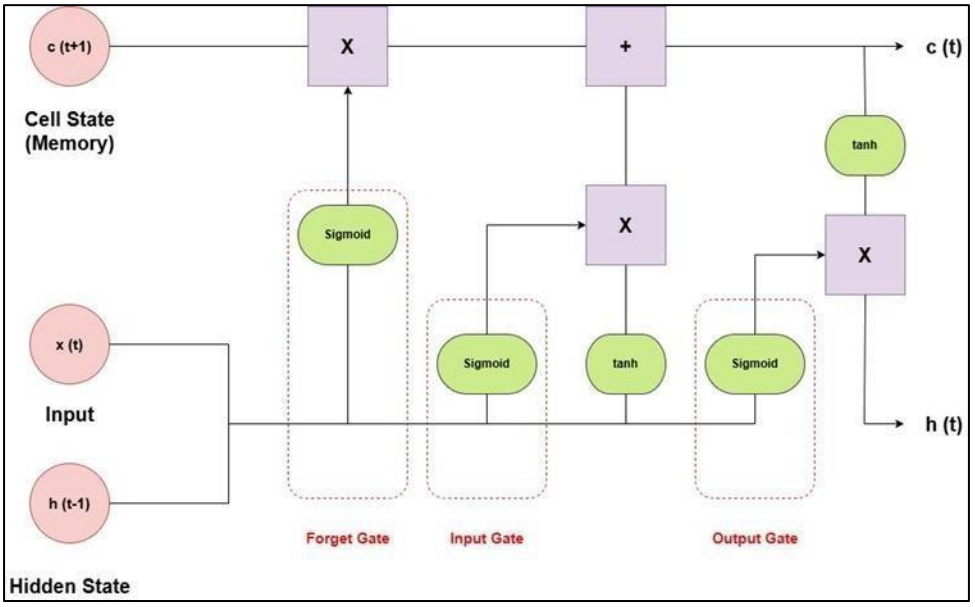
2.1.3. *Long Short-Term Memory (LSTM)*

Long Short Time Memory (LSTM) adalah salah satu jenis *neural network* yang dirancang khusus untuk mempodelkan data skuesnsial, terutama data time series. *LSTM* memiliki keunggulan dalam menangani dependensi jangka Panjang*(long-term dependencies)* pada data masukan, yang sering menjadi kendala dalam analisis data sekuensial. Dalam sebuah penelitian berjudul *"A New Method for Semantic Consistency Veryfication of Aviation Radiotelephony Communication Based on LSTM-RNN," LSTM* terbukti efektif diterapkan pada berbagai tugas sekuensial dan pemodelan Bahasa.

LSTM memiliki struktur yang disebut *cell* yang berfungsi sebagai penyimpanan informasi atau *cell state* untuk jangka Panjang maupun jangka waktu pendek. Keunggulan *LSTM* yang utama terletak pada *memory block* yang mampu memilih informasi yang tidak penting. Hal ini menjadikan *LSTM* sangat efektif dalam mempproses data sekuensial yang kompleks dengan mempertahankan informasi penting dari input sebelumnya.(Wiranda & Sadikin, 2020)

Metode LSTM dipilih dalam penelitian ini karena memiliki kemampuan untuk menangani data sekuensial yang kompleks, seperti teks jawaban esai siswa, dengan

mempertahankan informasi penting dari urutan data sebelumnya. Keunggulan LSTM dalam menangani dependensi jangka panjang menjadikannya sangat sesuai untuk memahami hubungan antar kata dan konteks kalimat dalam teks esai. Selain itu, struktur gate pada LSTM, seperti forget gate dan input gate, memungkinkan model untuk menyaring informasi yang relevan, sehingga menghasilkan prediksi yang lebih akurat dibandingkan dengan metode lain seperti RNN standar.



Gambar II.2 struktur Long Short-Term Memory

Pada gambar *II.2* terdapat jarngan *Long Short-Term Memory (LSTM)* yang merupakan jenis jaringan saraf berulang *(RNN)* seperti data deret waktu atau teks Bahasa alami. *LSTM* memiliki beberapa komponen utama seperti berikut:

1. ***Cell State*:** inti dari memori *LSTM.* Memori sepanjang masa.
2. ***Hidden State*:** mewakili keadaan saat ini dari *LSTM.* Memori masalalu.
3. ***Gates:*** mengontrol aliran informasi yang masuk dan kleuar dari *cell state.*
    - ***Forget Gate*:** menentukan informasi yang harus dihapus dari *cell state*
    - ***Input Gate* :** menentukan informasi yang harus ditambahkan ke *cell state*
    - ***Output Gate*:** mengontrol informasi dari *cell state* yang digunakan untuk memperbarui *hidden state.*

2.1.4. *Pre-Processing Text*

Dataset yang tidak terstruktur, seperti hasil *scarping* dari twitter, memerlukan proses pra-pemrosesan untuk menghilangkan redundansi data serta kata-kata yang tidak relevan. Proses ini bertujuan untuk menghapus duplikasi dan memperbaiki inkonsesistensi dalam data. Tahapan-tahapan yang biasanya dilakukan pra-premosesan datateks meliputi *case folding*, ppembesihan data, *tokensasi,* normalisasi, penghapusan *stopword*, dan *stemming. Case folding* adalah Langkah mengubah semua kata dalam dokumen menjadi huruf kecil. Pemebrsihan data bertujuan untuk menghilangkan tanda baca, symbol, angka, dan emoticon dari dokumen secara keseluruhan. (Dirjen et al., 2018). Tahap awal pemrosesan bertujuan untuk mempersiapkan data agar sesuai untuk diolah oleh algoritma *Bi-LSTM*. Proses ini mencakup beberapa Langkah, seperti mengonversi teks menjaaadi huruf kecil, menghapus tanda baca, menangani stopwords. (Siringoringo et al., 2023)

2.1.5. *Tokenisasi*

Tokenisasi adalah proses pemecah teks atau kalimat menjadi unit-unit yang lebihkecil yang disebut token. Token ini berupa kata, frasa, atau bahkan karakter. Pada Tingkat palingsederhana, tokenisasi dilakukan dengan memisahkan kata-kata dalam kalimat menggunakan spasi sebagai pemisah. Namun, proses ini juga mencakup langka-langkah lebih kpompleks, sperti misalkan tanda baca, menghapus karakter khusus, atau mengelompokkan kata-kata yang memiliki keterkaitan menjadi satu token. Salah satu parameter dalam *tokenizer* adalah *oov_token,* yang berfungsi mengganti kata-kata yang tidak dapat ditokenisasi dengan karakter tertentu. (Siringoringo et al., 2023)

2.1.6. *Filtering dan Stopword*

Dalam pengolahan bahasa alami (*Natural Language Processing/NLP), filtering* merupakan proses penting yang bertujuan untuk menyaring dan mengurangi data teks agar lebih relevan dan mudah dianalisis. Salah satu teknik filtering yang umum digunakan adalah penghapusan *stopwords. Stopwords* adalah kata-kata umum dalam suatu bahasa yang sering kali tidak memberikan makna signifikan dalam konteks analisis teks, seperti "dan", "atau", "adalah", dan "di".

Penghapusan stopwords membantu mengurangi dimensi data dan meningkatkan efisiensi pemrosesan, sehingga model pembelajaran mesin dapat fokus pada kata-kata yang lebih informatif dan relevan (Rachmawati & Pusponegoro, 2021)Menunjukkan bahwa penghilangan stopwords dapat meningkatkan akurasi model dalam tugas klasifikasi teks Menekankan pentingnya *preprocessing*, termasuk penghapusan *stopwords,* dalam meningkatkan performa sistem analisis sentiment. Dengan demikian, filtering dan penghapusan stopwords merupakan langkah krusial dalam *preprocessing* data teks yang mendukung pengembangan sistem *NLP* yang lebih efektif.

2.1.7 MaLSTM

Model MaLSTM (Manhattan LSTM) merupakan arsitektur yang dikembangkan untuk mengukur kemiripan semantik antara dua kalimat, terutama dalam tugas-tugas seperti semantic similarity dan sentence matching. Arsitektur ini memanfaatkan jaringan Long Short-Term Memory (LSTM) ganda yang memiliki bobot parameter yang sama atau disebut juga dengan shared weights. Dalam implementasinya, MaLSTM terdiri dari dua cabang jaringan LSTM yang masing-masing menerima satu input kalimat, yaitu kalimat A dan kalimat B. Kedua input ini terlebih dahulu dikonversi ke dalam bentuk vektor kata (word embeddings), yang kemudian diproses secara paralel oleh masing-masing LSTM.(Phreeraphattanakarn & Kijsirikul, 2021)

Output dari masing-masing LSTM adalah representasi vektor dari masing-masing kalimat, yang secara semantik mengandung makna keseluruhan dari kalimat tersebut. Untuk mengukur tingkat kemiripan antara kedua kalimat tersebut, digunakan fungsi jarak Manhattan (Manhattan Distance) yang menghitung seberapa jauh perbedaan antara dua vektor hasil keluaran dari LSTM A dan B. Nilai jarak ini kemudian diubah ke dalam bentuk skor kemiripan melalui fungsi aktivasi, biasanya sigmoid, sehingga dapat digunakan sebagai prediksi probabilistik terhadap tingkat kesamaan antara dua kalimat. Skor yang tinggi menunjukkan bahwa dua kalimat memiliki kemiripan semantik yang tinggi, dan sebaliknya.

Keunggulan utama dari MaLSTM adalah kemampuannya dalam memahami urutan kata dan konteks semantik yang kompleks, karena arsitektur LSTM sendiri dirancang untuk menangani data sekuensial dengan memperhatikan informasi dari masa lalu. Dengan menggunakan pendekatan Siamese Network (dua jaringan identik dengan bobot yang

sama), MaLSTM juga memastikan bahwa pembelajaran terhadap representasi dilakukan secara konsisten untuk kedua kalimat, yang sangat penting dalam penilaian kesamaan.

Dalam penelitian ini, MaLSTM digunakan sebagai bagian dari sistem automated essay scoring untuk mengevaluasi kemiripan antara jawaban siswa dan jawaban kunci (guru). Model ini dipilih karena relevansinya dalam menangkap kesamaan makna, bukan hanya persamaan kata secara eksplisit, sehingga sangat cocok untuk mengukur kualitas jawaban dalam bentuk esai.

2.3 Alasan pemilihan teori

Pemilihan metode Long Short-Term Memory (LSTM) sebagai dasar teori pada penelitian ini didasarkan pada beberapa pertimbangan berikut:

1. Kemampuan Memproses Data Berurutan

   LSTM dirancang secara khusus untuk menangani data sekuensial, di mana informasi masa lalu dapat memengaruhi hasil di masa depan. Hal ini relevan dengan konteks soal esai, karena jawaban siswa sering kali memiliki keterkaitan antar kata atau kalimat yang harus dipertimbangkan secara utuh.

2. Keunggulan dalam Mengatasi Masalah Long-Term Dependencies

   Salah satu tantangan utama dalam pemrosesan teks adalah masalah long-term dependencies, yaitu ketika konteks yang relevan berada jauh di dalam sekuens data. LSTM memiliki mekanisme gating yang memungkinkan model untuk "mengingat" atau "melupakan" informasi tertentu, sehingga dapat memproses jawaban siswa dengan lebih akurat meskipun terdapat variasi dalam penyampaiannya.

3. Cocok untuk Penilaian Soal Esai

   Jawaban soal esai memiliki keragaman bahasa yang tinggi, di mana siswa dapat memberikan jawaban dengan struktur kalimat yang berbeda meskipun maknanya sama. LSTM dapat menangkap pola-pola dalam teks ini dan menghasilkan representasi yang lebih baik untuk proses penilaian otomatis.

4. Penggunaan Fungsi Aktivasi yang Relevan

Fungsi aktivasi seperti sigmoid dan tanh yang digunakan dalam LSTM memungkinkan model untuk memetakan nilai-nilai penting dalam data, seperti tingkat kesesuaian jawaban siswa dengan label kebenaran. Mekanisme ini memberikan fleksibilitas dalam mempelajari pola-pola kompleks.

5. Dukungan oleh Penelitian Sebelumnya

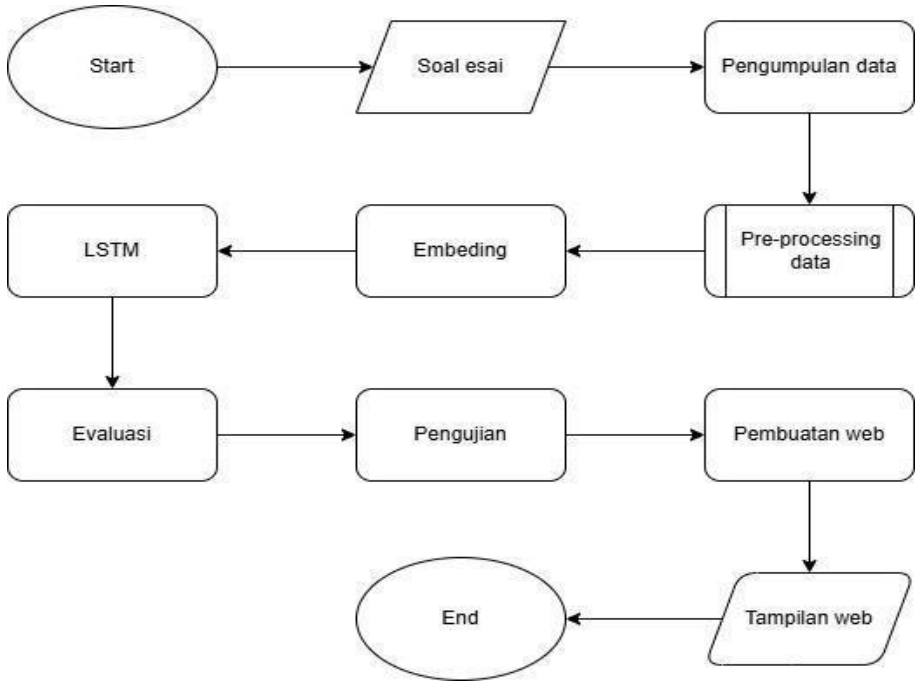
Banyak penelitian sebelumnya telah membuktikan keberhasilan LSTM dalam tugas pemrosesan bahasa alami (Natural Language Processing), seperti analisis sentimen, penerjemahan bahasa, dan penilaian teks. Oleh karena itu, LSTM diyakini memiliki potensi tinggi untuk diterapkan dalam sistem penilaian otomatis untuk soal esai.

6. Kebutuhan untuk Peningkatan Efisiensi Penilaian

Dalam dunia pendidikan, penilaian jawaban soal esai secara manual membutuhkan waktu dan usaha yang besar. Dengan kemampuan LSTM untuk belajar dari data latih dan memberikan evaluasi secara otomatis, proses ini dapat menjadi lebih efisien tanpa mengorbankan akurasi.

Dengan mempertimbangkan alasan-alasan tersebut, teori LSTM dipilih untuk mendasari pengembangan sistem penilaian otomatis pada penelitian ini. Metode ini diharapkan mampu mengatasi tantangan yang ada dalam penilaian soal esai dan memberikan hasil yang dapat membantu guru dalam proses pembelajaran.

BAB III
METODOLOGI PENELITIAN



Gambar III.1 Flowchart Alur Penelitian

Pada gambar III.1 terdapat flowchart untuk alur penelitian yang akan dilakukan yaitu, dengan mulai pengumpulan dataset seperti soal esai dan jawaban siswa, lalu tahap pre-processing data, lalu setelah itu tahap embeding, lalu tahap *LSTM,* lalu tahap evaluasi, lalu tahap pengujian rasio dan parameter *LSTM,* dan yang terakhir pembuatan web.

3.1 Pengumpulan dataset

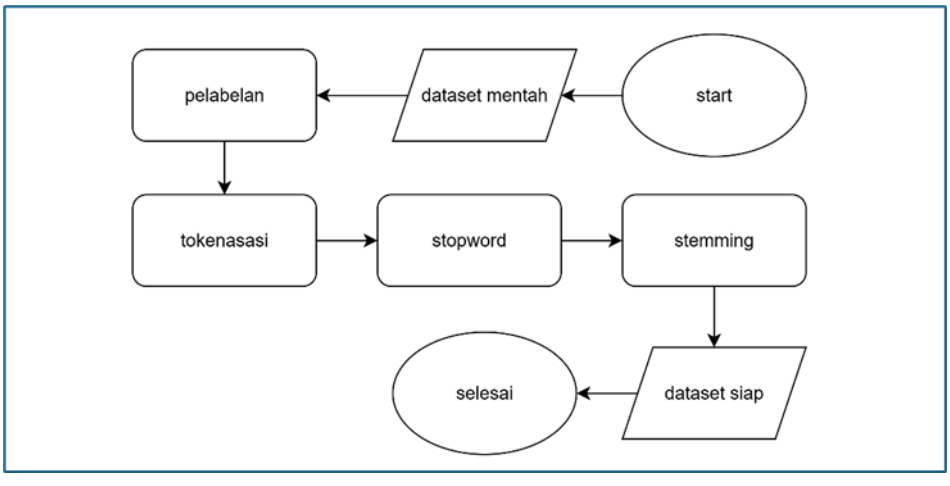
Pengumpulan dataset merupakan tahap awal yang sangat penting dalam penelitian ini, karena kualitas dataset akan sangat memengaruhi hasil dari sistem penilaian otomatis. Dataset yang digunakan terdiri dari , yaitu soal esai dari guru, jawaban siswa yang telah diberi label oleh guru, dan kunci jawaban guru. Soal esai dikumpulkan dari berbagai sumber, seperti buku pelajaran, soal ujian sekolah, serta bank soal daring yang relevan dengan mata pelajaran Ilmu Pengetahuan Sosial.

Dataset ini dimanfaatkan dalam dua bagian utama sistem:

- Pada model **LSTM Classifier**, dataset digunakan untuk melatih model dalam mengenali pola jawaban benar atau salah.
- Pada model **MaLSTM**, dataset digunakan untuk menghitung kemiripan semantik antara jawaban siswa dan jawaban guru.

Dengan membangun dataset sendiri, sistem menjadi lebih sesuai dengan konteks pelajaran IPS dan lebih adaptif terhadap variasi jawaban alami dari siswa di Indonesia.

3.2 Pre-processing dataset



Gambar III.2 Alur Pre-Processing

Pada gambar III.2 terdapat flowchart untuk preprocessing .Setelah dataset terkumpul, tahap *preprocessing* dilakukan untuk mempersiapkan data sebelum digunakan dalam model. Dataset yang digunakan dalam penelitian ini hanya terdiri dari soal esai dan jawaban siswa yang telah diberi label oleh guru IPS. Label ini berupa angka 0 untuk jawaban salah dan 1 untuk jawaban benar. Dataset tidak membutuhkan kunci jawaban guru karena penilaian jawaban siswa sudah merujuk pada kunci jawaban guru.

Contoh Dataset:

a. Soal: "Jelaskan mengapa Indonesia disebut negara maritim?"
b. Jawaban Siswa: "Karena Indonesia memiliki banyak pulau dan wilayah perairan yang luas."

c. Label: 1 (Benar)

Pada gambar III.2 merupakan alur dari *pre-processing* data yang terdiri dari input data, *Tokenisasi, Stopword, Stemming,* Pelabelan, dan output.

1. ***Input* Data**
   a. *Input*: Jawaban siswa berupa teks ("Karena Indonesia memiliki banyak pulau dan wilayah perairan yang luas") dimasukkan ke model.
   b. Tujuan: Memasukkan jawaban siswa ke tahap *preprocessing* untuk mempersiapkan data.
2. ***Preprocessing***

   Proses:

   a. *Tokenisasi*: Memecah teks menjadi unit kata:
      ["Karena", "Indonesia", "memiliki", "banyak", "pulau", "dan", "wilayah", "perairan", "yang", "luas"]
   b. *Stopword Removal*: Menghapus kata-kata yang tidak relevan seperti "dan", "yang" menjadi:
      ["Karena", "Indonesia", "memiliki", "banyak", "pulau", "wilayah", "perairan", "luas"]
   c. *Stemming*: Mengubah kata menjadi bentuk dasar, seperti
      ["Karena", "Indonesia", "miliki", "banyak", "pulau", "wilayah", "air", "luas"]
3. **Pelabelan**

   Pelabelan adalah proses memberikan label pada data jawaban siswa untuk mengidentifikasi tingkat kebenaran berdasarkan koreksi guru. Pada penelitian ini, jawaban siswa telah dikoreksi langsung oleh guru IPS dengan memberikan label "1" untuk jawaban yang benar dan "0" untuk jawaban yang salah. Proses pelabelan ini memastikan bahwa setiap jawaban siswa sudah memiliki acuan kebenaran yang jelas berdasarkan penilaian manual guru.

   Dengan pelabelan ini, dataset menjadi lebih terstruktur dan siap digunakan dalam tahap pelatihan model. Label "0" dan "1" juga

memungkinkan model untuk belajar membedakan pola-pola jawaban siswa yang sesuai atau tidak sesuai dengan kriteria jawaban yang benar. Hasil dari pelabelan ini menjadi dasar untuk mengukur performa model dalam menilai jawaban siswa secara otomatis.

Output: Teks yang telah dibersihkan untuk dimasukkan ke tahap embedding.

3.3 Embedding

Tahap *embedding* adalah proses mengubah data teks yang sudah melalui *preprocessing* menjadi representasi numerik yang dapat dipahami oleh model *LSTM*. Dalam penelitian ini, metode embedding yang digunakan adalah *Word2Vec* atau *GloVe*, yang mampu merepresentasikan kata-kata dalam bentuk vektor angka. Representasi ini tidak hanya mencerminkan arti kata secara individual, tetapi juga hubungan semantik antar kata.

Sebagai contoh, kata "Indonesia" dan "negara" mungkin memiliki representasi vektor yang dekat, karena memiliki makna yang berhubungan. *Embedding* ini sangat penting untuk memastikan model dapat memahami konteks dan makna dari data teks esai yang diberikan. Proses embedding dilakukan dengan melatih model embedding pada dataset yang sudah disiapkan atau menggunakan embedding pre-trained jika dataset terbatas.

Input: Teks hasil *preprocessing* (*["Karena", "Indonesia", "miliki", "banyak", "pulau", "wilayah", "air", "luas"]*).

Proses: Setiap kata diubah menjadi representasi vektor dengan embedding, misalnya:
"Indonesia" → [0.12, -0.45, 0.67, ...]
"pulau" → [0.34, 0.56, -0.21, ...]

Output: Representasi vektor untuk tiap kata yang membentuk jawaban siswa.

3.4 Proses Pembangunan Sistem

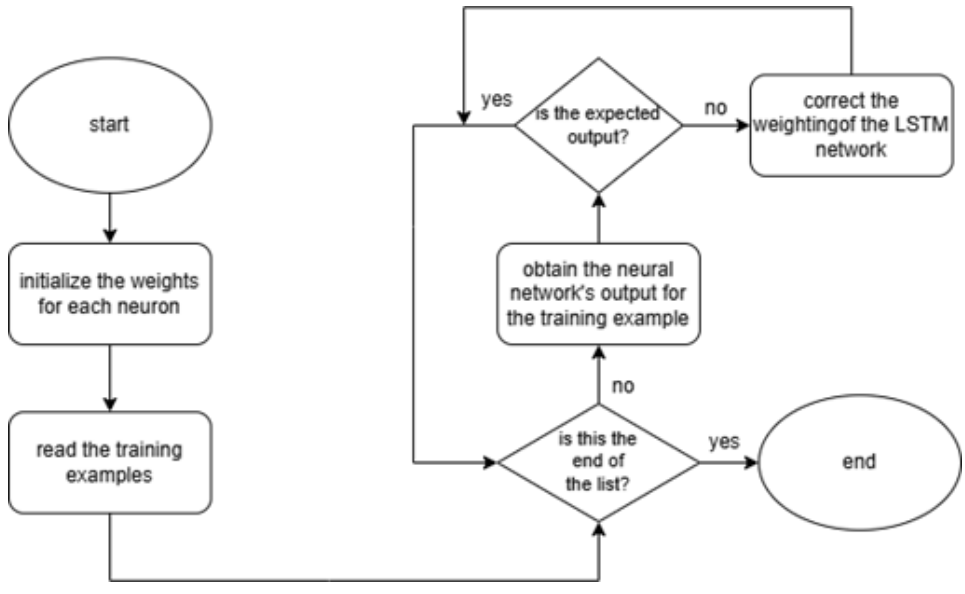
Proses pembangunan sistem dimulai dengan tahapan persiapan dan pengolahan data. Dataset yang digunakan terdiri atas soal, jawaban siswa, dan label penilaian (Benar/Salah) yang telah divalidasi oleh guru. Sebelum digunakan untuk pelatihan model, data terlebih dahulu melalui proses pra-pemrosesan (preprocessing) yang mencakup pembersihan teks, konversi huruf menjadi huruf kecil, penghapusan tanda baca, serta tokenisasi menggunakan tokenizer FastText.DONE

Setelah preprocessing selesai, data dibagi menjadi dua bagian utama, yaitu data latih dan data uji. Pembagian ini dilakukan menggunakan metode train-test split dengan proporsi tertentu (misalnya 80:20) agar model dapat diuji dengan data yang tidak dilibatkan dalam proses pelatihan. Pembagian ini bertujuan untuk mengevaluasi performa model secara objektif terhadap data baru yang belum pernah dilihat sebelumnya.

Untuk membangun dua model yang diusulkan, yaitu LSTM Classifier dan MaLSTM (Manhattan LSTM), masing-masing dilatih menggunakan data latih hasil pembagian tersebut. Model LSTM digunakan sebagai binary classifier untuk mengklasifikasikan jawaban siswa menjadi Benar atau Salah berdasarkan pola sekuensial dalam jawaban. Sedangkan model MaLSTM digunakan untuk menghitung tingkat kemiripan semantik antara jawaban siswa dan kunci jawaban guru berdasarkan representasi embedding dan perhitungan jarak Manhattan.

Setelah pelatihan selesai, kedua model disimpan dalam format .h5 dan diintegrasikan ke dalam sistem web menggunakan framework Flask. Sistem ini memungkinkan pengguna untuk memasukkan soal, kunci jawaban guru, dan jawaban siswa melalui antarmuka web. Setelah input diberikan, kedua model akan dijalankan secara bersamaan untuk menampilkan hasil prediksi klasifikasi serta skor kemiripan semantik secara otomatis dan real-time.

3.5 LSTM



Gambar III.3 Flowchart LSTM

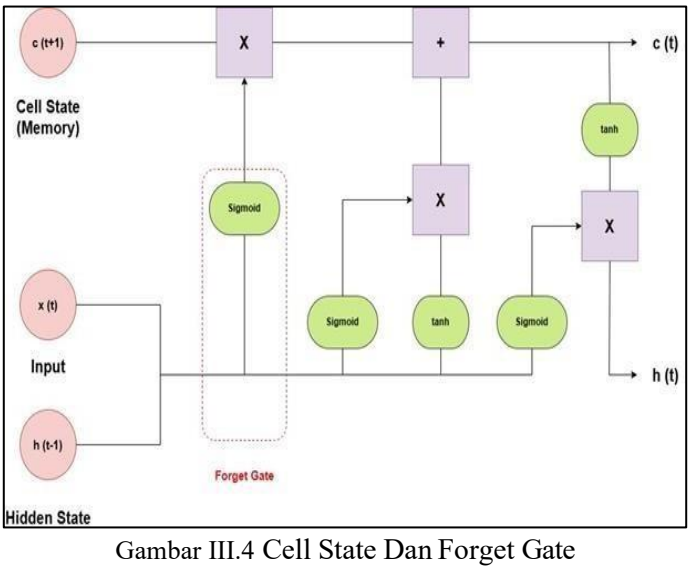
Pada Gambar III. 3 Gambar tersebut menunjukkan alur proses pelatihan model Long Short-Term Memory (LSTM) dalam menyelesaikan tugas klasifikasi. Proses diawali dengan tahap inisialisasi bobot (weight) pada setiap neuron dalam jaringan. Inisialisasi ini penting karena bobot awal akan memengaruhi proses pembelajaran dan konvergensi model. Setelah itu, sistem membaca data latih satu per satu. Data latih ini terdiri dari input (fitur) dan target (label) yang digunakan sebagai referensi selama proses pembelajaran berlangsung.

Selanjutnya, model LSTM akan memproses input dan menghasilkan output berupa prediksi. Hasil output ini kemudian dibandingkan dengan nilai label yang diharapkan. Jika prediksi yang dihasilkan oleh model belum sesuai atau berbeda dari label yang seharusnya, maka sistem akan melakukan proses koreksi bobot melalui algoritma backpropagation, khususnya Backpropagation Through Time (BPTT), yang dirancang untuk menangani data sekuensial seperti teks. Koreksi ini dilakukan untuk meminimalkan kesalahan (loss) antara prediksi model dan label sebenarnya.

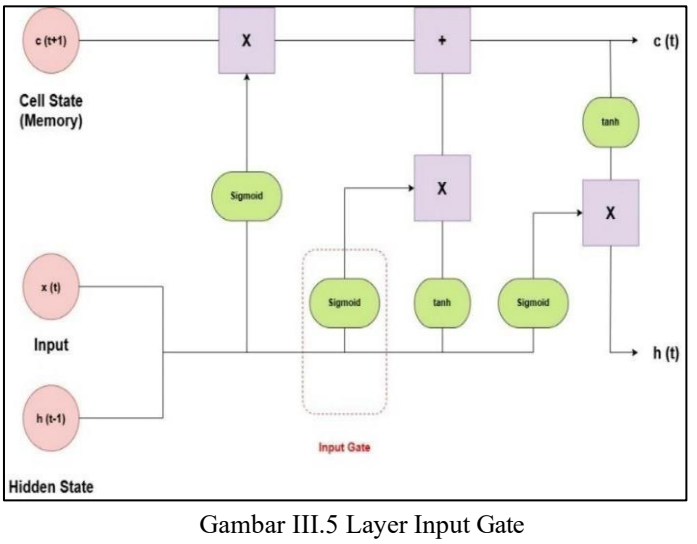
Setelah satu data selesai diproses, sistem akan mengecek apakah seluruh data dalam kumpulan data latih sudah digunakan. Jika belum, maka sistem akan melanjutkan ke data berikutnya dan mengulangi proses yang sama mulai dari menghasilkan output hingga melakukan koreksi bobot jika diperlukan. Siklus ini berlangsung terus hingga

seluruh data latih selesai diproses. Dengan kata lain, model terus diperbaiki secara bertahap menggunakan banyak iterasi (epoch) agar mampu mengenali pola dan hubungan temporal dalam data teks, terutama pada data esai siswa yang bersifat sekuensial.Proses pelatihan ini menjadi fondasi utama agar model LSTM dapat melakukan klasifikasi dengan akurasi tinggi ketika digunakan untuk data uji atau data baru yang belum pernah dilihat sebelumnya.

Dataset yang digunakan dalam penelitian ini hanya terdiri dari **soal esai** dan **jawaban siswa yang telah diberi label** oleh guru IPS. Label ini berupa angka 0 untuk jawaban salah dan 1 untuk jawaban benar. Dataset tidak membutuhkan kunci jawaban guru karena penilaian jawaban siswa sudah merujuk pada kunci jawaban guru. Pada *gambar III.4* menjelaskan bahwa data yang masuk akan menuju *cell state.*



Gambar III.4 Cell State Dan Forget Gate



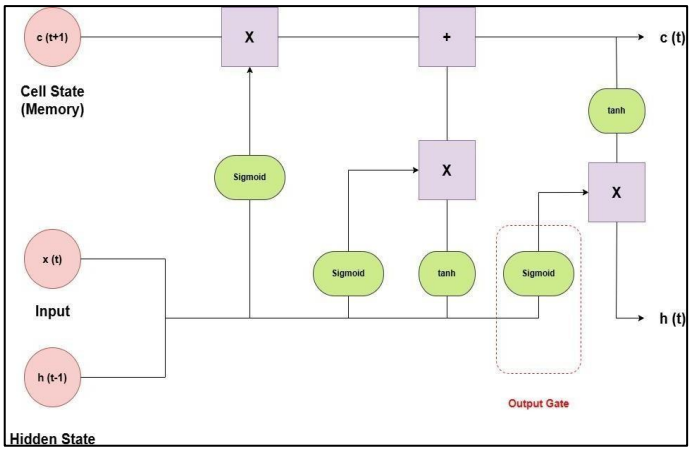
Gambar III.5 Layer Input Gate

Pada *gambar III.5* menunjukkan layer input gate, yang menjalaskan proses teradinya informasi atau dataset yang berada di *cell state* masuk ke *layer input gate.*

1. **Input**: Vektor dari embedding layer.
   a. **Forget Gate (Sigmoid)**: Memutuskan informasi mana yang akan dilupakan dari memori. Misalnya, informasi terkait kata *"banyak"* dianggap tidak relevan dan dihapus.
   b. **Input Gate (Sigmoid)**: Memutuskan informasi baru mana yang akan disimpan. Contohnya, hubungan antara kata *"pulau"* dan *"wilayah"* dianggap penting untuk konteks soal.
   c. **Tanh**: Membuat representasi baru dengan nilai antara -1 hingga 1 untuk menambahkan konteks yang lebih terfokus.
   d. **Output Gate (Sigmoid)**: Memutuskan bagian mana dari memori yang akan digunakan sebagai output. Misalnya, informasi tentang *"pulau" dan "wilayah air"* diteruskan karena penting untuk penilaian.
   e. **Output**: Representasi jawaban siswa yang mencerminkan hubungan temporal antar kata.
2. Fully Connected Layer

   Input: Output dari *LSTM* berupa representasi akhir dari teks.
   Proses:

   Output: Skor probabilitas antara 0 hingga 1.



Gambar III.6 Layer Output Gate

Pada *gambar III.6* terdapat *layer output gate*, yang terdapat proses terjadinya dataset masuk setelah diproses di *layer input gate.*

3. **Output Layer**

Input: Skor probabilitas dari fully connected layer.

Output Layer:

a. Terdiri dari dua node: `y1` (nilai kebenaran) dan `y2` (nilai kesalahan).
b. Menggunakan fungsi aktivasi **softmax** untuk menghasilkan dua skor probabilitas yang totalnya sama dengan 1.

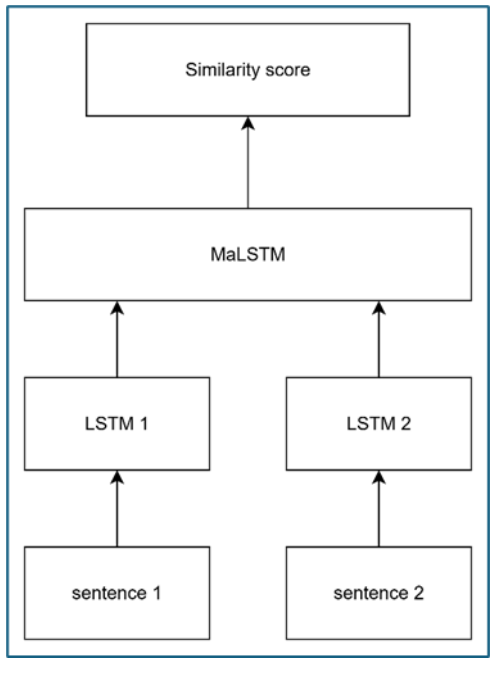
Interpretasi Output:

a. Jika `y1 > y2`, maka output adalah "Benar", dengan **confidence level** sebesar `y1 * 100%`.
b. Jika `y2 > y1`, maka output adalah "Salah", dengan **confidence level** sebesar `y2 * 100%`.

Contoh:

a. Jika model menghasilkan output `[0.8, 0.2]`, maka:
   - `y1 = 0.8`, `y2 = 0.2`.
   - Prediksi: "Benar" dengan confidence level 80%.
b. Jika model menghasilkan output `[0.3, 0.7]`, maka:
   - `y1 = 0.3`, `y2 = 0.7`.
   - Prediksi: "Salah" dengan confidence level 70%.

3.6 Manhattan LSTM



Gambar III.7 Flowchart MaLSTM

Pada gambar III.7 terdapat flowchart MaLSTM Arsitektur MaLSTM dalam sistem ini dirancang dengan pendekatan Siamese Network, di mana terdapat dua buah input utama yang disebut sebagai Sentence 1 dan Sentence 2. Kedua input ini merepresentasikan pasangan jawaban, yaitu satu jawaban dari siswa dan satu kunci jawaban guru sebagai acuan. Masing-masing kalimat ini terlebih dahulu dikonversi menjadi vektor embedding menggunakan pretrained FastText untuk menangkap informasi semantik kata secara lebih dalam.

Setelah itu, kedua vektor input ini dialirkan ke dalam dua buah layer LSTM yang identik dan berbagi bobot (shared LSTM). Artinya, LSTM 1 dan LSTM 2 memiliki parameter yang sama sehingga proses pembelajaran berlangsung secara konsisten untuk kedua sisi input. Output dari masing-masing LSTM adalah representasi kontekstual dari masing-masing kalimat.

Langkah selanjutnya adalah menghitung jarak antara dua output tersebut menggunakan Manhattan Distance, yang menjadi ciri khas dari metode MaLSTM. Perhitungan ini menghasilkan sebuah nilai Similarity Score, yaitu skor kemiripan antara dua kalimat. Skor tersebut kemudian diproses melalui fungsi aktivasi sigmoid untuk menghasilkan probabilitas antara 0 dan 1. Probabilitas ini diinterpretasikan sebagai seberapa besar kemungkinan kedua kalimat (jawaban siswa dan guru) memiliki makna yang serupa, dan digunakan untuk menentukan label akhir (benar atau salah).

Dengan desain ini, MaLSTM mampu menangkap kemiripan semantik antara dua jawaban meskipun secara permukaan (kata-kata yang digunakan) berbeda. Arsitektur ini sangat cocok diterapkan pada tugas automated essay scoring, karena mampu memahami konteks makna daripada sekadar mencocokkan kata secara literal.

3.7 Evaluasi Model

Tahap evaluasi dilakukan untuk mengukur seberapa baik model yang telah dilatih dalam memberikan penilaian otomatis. Evaluasi ini dilakukan menggunakan beberapa metrik, yaitu akurasi*, precision, dan recall*. Akurasi mengukur persentase prediksi model yang sesuai dengan jawaban sebenarnya, precision menghitung tingkat ketepatan prediksi model, sedangkan recall

mengukur sejauh mana model mampu mendeteksi jawaban benar dalam dataset. Hasil evaluasi ini akan digunakan untuk menentukan apakah model sudah cukup baik atau perlu perbaikan lebih lanjut.

Confidence level yang dihasilkan oleh model diuji untuk memastikan validitasnya. Uji ini dilakukan dengan menganalisis distribusi confidence level pada prediksi yang benar dan salah, serta membandingkan skor probabilitas dengan metrik

evaluasi lain, seperti akurasi, precision, dan recall. Confidence level yang konsisten dengan hasil prediksi akan meningkatkan interpretabilitas model.

3.8 Pengujian

Penelitian ini bertujuan untuk menganalisis pengaruh preprocessing terhadap performa model Long Short-Term Memory (LSTM) dengan dua skenario pengujian utama, yaitu:

1. **Skenario dengan Preprocessing**: Dalam skenario ini, data jawaban siswa terlebih dahulu diproses melalui tahapan preprocessing, yang meliputi tokenisasi, stopword removal, dan stemming. Tujuan dari preprocessing ini adalah untuk menghapus informasi yang tidak relevan dan memperbaiki struktur data agar lebih mudah dipahami oleh model.
2. **Skenario Tanpa Preprocessing**: Pada skenario ini, data jawaban siswa dimasukkan langsung ke model LSTM tanpa melalui proses preprocessing. Tujuan dari pengujian ini adalah untuk mengukur performa model dengan menggunakan data mentah, sehingga dapat dibandingkan dengan skenario sebelumnya untuk melihat seberapa besar pengaruh preprocessing terhadap akurasi model.

Selanjutnya, tahap pengujian dilakukan untuk menganalisis faktor-faktor yang memengaruhi performa model. Pengujian ini dibagi menjadi dua bagian utama:

1. **Pengaruh Rasio Data**: Pengujian ini bertujuan untuk menganalisis pengaruh rasio data latih dan data uji terhadap akurasi model *LSTM Classifier*. Beberapa rasio yang diuji adalah 80:20, 70:30, dan 60:40.

Setiap rasio diuji untuk menentukan komposisi terbaik yang memberikan akurasi optimal.

2. **Pengaruh Parameter LSTM dan MaLSTM**: Pengujian ini dilakukan untuk mengetahui bagaimana berbagai parameter LSTM memengaruhi akurasi model. Parameter yang diuji meliputi jumlah unit memori, jumlah lapisan (*layer), dan learning rate.* Hasil ini diharapkan dapat memberikan gambaran mengenai konfigurasi parameter terbaik yang dapat meningkatkan performa model.
3. **Uji similarity score MaLSTM**: Pengujian ini bertujuan untuk mengevaluasi seberapa baik model MaLSTM menghasilkan skor similarity yang sesuai dengan label kebenaran dari data. Untuk itu, dilakukan analisis terhadap distribusi nilai similarity berdasarkan label.

   Distribusi similarity ditampilkan dalam bentuk histogram untuk dua kelas: label 1 (jawaban benar/mirip) dan label 0 (jawaban salah/tidak mirip). Hasil dari distribusi ini digunakan untuk mengevaluasi apakah model mampu membedakan pasangan kalimat yang mirip dan tidak mirip secara efektif. Semakin besar jarak antara distribusi skor similarity untuk label 1 dan 0, semakin baik kinerja model dalam mengukur kesamaan makna antar kalimat.
4. **Uji thresholding**: Pengujian ini dilakukan untuk mengevaluasi kemampuan MaLSTM dalam membedakan jawaban benar dan salah apabila skor similarity diubah menjadi prediksi kelas biner menggunakan ambang batas (threshold) tertentu. Beberapa nilai threshold diuji, seperti 0.5, 0.6, 0.7, dan 0.8. Setelah dilakukan thresholding, hasil klasifikasi dievaluasi menggunakan metrik seperti accuracy, precision, recall, dan f1-score melalui fungsi `classification_report` dari Scikit-Learn.

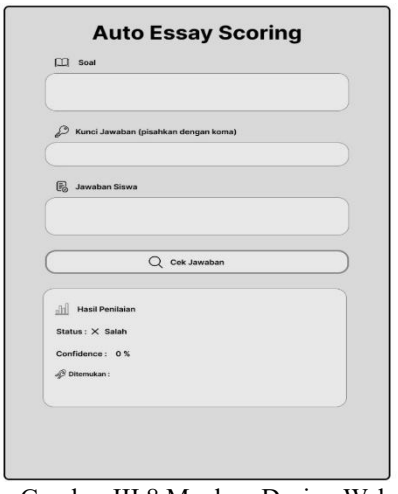
   Uji threshold ini dilakukan sebagai pendekatan tambahan untuk memahami sejauh mana skor similarity dari MaLSTM bisa digunakan sebagai dasar klasifikasi jawaban siswa.
5. **Uji Fungsionalitas sistem :** Uji fungsionalitas dilakukan untuk memastikan bahwa seluruh fitur dalam sistem bekerja sesuai dengan

fungsinya dan dapat digunakan oleh pengguna secara interaktif. Pengujian ini mencakup beberapa aspek utama, antara lain kemampuan sistem menerima input dari pengguna (soal, kunci jawaban guru, dan jawaban siswa), menjalankan proses klasifikasi menggunakan dua model (LSTM dan MaLSTM), serta menampilkan hasil keluaran berupa status klasifikasi, tingkat kepercayaan (confidence), dan skor kemiripan semantik. Setiap fungsi diuji menggunakan skenario penggunaan nyata, seperti yang akan dilakukan oleh guru dalam proses penilaian esai. Uji fungsionalitas juga mencakup validasi apakah setiap komponen dalam alur sistem dapat berjalan tanpa error, dan hasilnya sesuai dengan yang diharapkan berdasarkan data masukan. Hasil dari pengujian ini menunjukkan bahwa sistem berhasil merespons seluruh masukan pengguna dengan baik, dan menampilkan hasil prediksi dalam antarmuka web secara jelas, cepat, dan akurat.

3.9 Pembuatan web

Setelah model selesai dilatih dan diuji, langkah berikutnya adalah mengintegrasikan model ke dalam sebuah aplikasi web. Aplikasi ini dirancang agar guru dapat dengan mudah menggunakan sistem penilaian otomatis ini. Proses pembuatan web melibatkan pengembangan frontend dan backend. Frontend dirancang menggunakan framework seperti React untuk memberikan antarmuka yang user- friendly. Backend dikembangkan menggunakan Flask, yang bertugas menghubungkan model LSTM dengan aplikasi web. Data input berupa soal dan jawaban siswa dikirimkan melalui web untuk diproses oleh model, dan hasil penilaian ditampilkan kembali kepada pengguna.

3.10 Tampilan web



Gambar III.8 Mockup Design Web

Pada Gambar III.8 bagian terakhir dari sistem ini adalah tampilan web yang digunakan oleh pengguna. Tampilan web dirancang agar intuitif dan mudah dipahami, dengan halaman input untuk memasukkan soal esai dan jawaban siswa. Selain itu, terdapat halaman hasil penilaian yang menampilkan skor dan analisis jawaban. Desain web dibuat responsif dan dapat diakses di berbagai perangkat.

3.11 Jadwal Penelitian

Tabel III.1 Rencana Jadwal Penelitian

No.	Nama Kegiatan	Juli					Agustus					September					Oktober					November					Desember				
		1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	Pengajuan Judul					■	■																								
2	Studi Literatur		■	■			■	■	■	■	■	■	■	■	■	■						■	■	■							
3	Pengumpulan Data									■	■	■	■	■	■	■				■	■	■	■	■			■	■	■	■	■
4	Perancangan Model													■	■	■	■	■	■	■	■										
5	Penulisan Proposal								■	■	■	■			■	■	■	■	■	■	■										
6	Implementasi dan Pengujian														■	■				■	■	■	■	■	■	■					
7	Evaluasi dan Analisis																					■	■	■	■	■	■	■	■	■	■
8	Penyelesaian Tugas Akhir																			■	■	■	■	■	■	■	■	■	■	■	■

Tabel III.1 merupakan rencana jadwal penelitian yang dilakukan selama 6 bulan sejak bulan Juli hingga Desember. Terdapat 8 kegiatan, yaitu pengajuan judul, studi

literatur, pengumpulan data, perancangan model, penulisan proposal, implementasi dan pengujian, evaluasi dan analisis, serta penyelesaian tugas akhir.

BAB IV
HASIL DAN PEMBAHASAN

4.1. Implementasi

4.1.1. Pengumpulan *Dataset*

Pada tahap pertama adalah pengumpulan data yang dilakukan melalui cara mendatangi beberapa sekolah menengah pertama. Data yang dikumpulkan berupa teks soal yang dilakukan siswa saat ujian atau ulangan harian, jawaban siswa yang telah di koreksi yang telah dilabeli 0 untuk salah dan 1 untuk benar, dan kunci jawaban guru. Data didapatkan dengan cara berdiskusi langsung dengan guru mapel ips untuk dilakukan pengambilan soal dan jawaban siswa sebagai dataset, dengan jumlah dataset 1 soal dengan 30-50 jawaban siswa. Pada gambar IV.1 adallah contoh datset soal dan jawaban siswa yang telah dimiliki.

soal	jawaban_siswa	label
jelaskan yang dimaksud oil refinery	oil nefinary adalah pabrik atau fasillitas industri yang mengolah minyak mentahmenjadi produk petrolium	1
jelaskan yang dimaksud oil refinery	adalah pabrik atau fasilitas industri yang mengolah minyak mentah menjadi produk potrollium yang bisa llangsung digunakan maupun produk lain yang menjadi bahan baku bagi industri petrokimia	1
jelaskan yang dimaksud oil refinery	pabrik/fasillitas yang mengolah minyak mentah menjadi produk petrolium yg bisa langsung bisa digunakan maupun produk" lain yang menjadi bahan baku bagi industri petrokimia	1
jelaskan yang dimaksud oil refinery	engraving dan kartu pos, kalminir, poster reproduksi,percetakan, lukisan dan barang cetakan lainya serta rekaman mikro film	0
Apakah yang dimaksud dengan rumah tangga produsen?	rumah tangga produsen adalah istilah ekonomi yang merujuk pada ekonomi yang memprodusikan barang dan jasa dijual di pasar	1
Apakah yang dimaksud dengan rumah tangga produsen?	rumah tangga produsen adalah istilah ekonomi yang merujuk pada ekonomi yang memprodusikan barang dan jasa dijual di pasar	1
Apakah yang dimaksud dengan rumah tangga produsen?	rumah tangga produsen adalah rumah tangga yang sering dipakek untuk kebutuhan sehari hari contoh, panci, wajan, sotel, garpu, sendok, dll itu biasanya dibeli di pasar dibeli untuk dipokok sehari hari	0

Gambar IV.1 Contoh Dataset Soal Dan Jawaban Siswa

4.1.2. *Preprocessing* Data

Tahap *pre-processing* data terdiri dari beberapa tahapa yang akan dilakukan, yaitu pelabelan, tokenisasi, stopword,stemming

a. Proses Tokenisasi

Tabel IV.1 Kode Program Tokenisasi

	Kode program
	`# Tokenizer`
1	`tokenizer = Tokenizer()`
2	`tokenizer.fit_on_texts(texts)`
3	`sequences = tokenizer.texts_to_sequences(texts)`

pada tabel IV.I code program terdapat, proses memecah teks menjadi unit-unit kecil yang disebut *token*, biasanya berupa kata atau frasa, yang kemudian digunakan sebagai input dalam pemrosesan teks oleh model. tokenisasi diawali dengan membuat objek `Tokenizer()` yang berfungsi untuk mempersiapkan alat pemroses teks menjadi bentuk numerik. Selanjutnya, metode `fit_on_texts()` digunakan untuk membangun kamus (vocabulary) dari seluruh teks jawaban yang tersedia. Setelah kamus terbentuk, setiap kalimat dalam teks kemudian diubah menjadi deretan angka menggunakan `texts_to_sequences()`, di mana setiap kata diganti dengan indeks unik berdasarkan urutan frekuensinya. Tahapan ini penting agar teks bisa diproses oleh model berbasis deep learning seperti LSTM yang hanya menerima input berupa angka.

Tabel IV.2 Hasil Tokenisasi

Jawaban siswa asli	Hasil tokenisasi
oil nefinary adalah pabrik atau fasillitas industri yang mengolah minyak mentahmenjadi produk petrolium	['oil', 'nefinary', 'adalah', 'pabrik', 'atau', 'fasillitas', 'industri', 'yang', 'mengolah', 'minyak', 'mentahmenjadi', 'produk', 'petrolium']
engraving dan kartu pos, kalminir, poster reproduksi,percetakan, lukisan dan barang cetakan lainya serta rekaman mikro film	['engraving', 'dan', 'kartu', 'pos,', 'kalminir,', 'poster', 'reproduksi,percetakan,', 'lukisan', 'dan', 'barang', 'cetakan', 'lainya', 'serta', 'rekaman', 'mikro', 'film']
Menjaga stabilitas keuangan menjaga arus kos	['menjaga', 'stabilitas', 'keuangan', 'menjaga', 'arus', 'kos']

Pada tabel IV.2 terdapat tabel hasil dari jawaban siswa yang melalui tahap tokenisasi.

b. Stopword removal

proses **stopword removal**, yaitu menghapus kata-kata umum dalam Bahasa Indonesia yang dianggap tidak memiliki nilai penting dalam analisis, seperti “yang”, “dan”, “di”, dan sebagainya. Sebelum menghapus stopword, teks diubah menjadi huruf kecil agar sesuai dengan format dalam daftar stopword. Selain itu, karakter non-alfabet juga dihapus menggunakan regular expression.

Tabel IV.3 Hasill Stopword

tokenizing	Stopword
['oil', 'nefinary', 'adalah', 'pabrik', 'atau', 'fasillitas', 'industri', 'yang', 'mengolah', 'minyak', 'mentahmenjadi', 'produk', 'petrolium']	['oil', 'nefinary', 'pabrik', 'fasillitas', 'industri', 'mengolah', 'minyak', 'mentahmenjadi', 'produk', 'petrolium']
['engraving', 'dan', 'kartu', 'pos', 'kalminir', 'poster', 'reproduksipercetakan', 'lukisan', 'dan', 'barang', 'cetakan', 'lainya', 'serta', 'rekaman', 'mikro', 'film']	['engraving', 'kartu', 'pos', 'kalminir', 'poster', 'reproduksipercetakan', 'lukisan', 'barang', 'cetakan', 'lainya', 'rekaman', 'mikro', 'film']
['menjaga', 'stabilitas', 'keuangan', 'menjaga', 'arus', 'kos']	['menjaga', 'stabilitas', 'keuangan', 'menjaga', 'arus', 'kos']

Pada tabel IV.3 adalah contoh hasil tokenisasi yang kemudian masuk ke tahap stopword removal untuk kemudian masuk ke tahap embedding.

Tabel IV.4kode Program Stopword

	Kode program
1	`import nltk`
2	`nltk.download('stopwords')`
3	`from nltk.corpus import stopwords`
4	
5	`stop_words = set(stopwords.words('indonesian'))`
6	
7	`def clean_text(text):`
8	`    text = text.lower()`
9	`    text = re.sub(r'[^a-zA-Z\s]', '', text)  # hapus`
10	`karakter selain huruf & spasi`
11	`    words = text.split()`
12	`    words = [word for word in words if word not in`
13	`stop_words]  # stopword removal`
14	`    return ' '.join(words)`

Pada tabel VI.4 terdapat kode yangberfungsi untuk stopword pada tahap preprocessing setelah masuk tahap tokenizing, berikut adalah penjelasan dari kode diatas, Potongan kode ini berfungsi untuk membersihkan jawaban siswa sebelum diproses lebih lanjut. Modul `nltk` digunakan untuk mengambil daftar stopword dalam bahasa Indonesia, yang kemudian disimpan ke dalam variabel `stop_words`. Fungsi `clean_text` akan mengubah semua huruf dalam teks menjadi huruf kecil agar konsisten, lalu memecah teks menjadi kata-kata. Setelah itu, kata-kata yang termasuk dalam daftar stopword dihapus agar hanya menyisakan kata-kata yang relevan. Proses ini bertujuan untuk mengurangi noise dalam data sebelum masuk ke tahap pemodelan.

c. Stemming

Tabel IV.5 Kode Program Stemming

	Kode program
1	`=== Stemming sederhana`
2	`def simple_stem(word):`
3	`    word = word.lower()`
4	`    word =`
5	`re.sub(r'\b(meng|meny|mem|me|di|ter|ke|se|ber|per)', '',`
6	`word)`
7	`    word = re.sub(r'(kan|an|i)\b', '', word)`
8	`    return word`

Stemming adalah proses mengubah kata turunan menjadi bentuk dasarnya (*root word*). Tujuannya untuk menyamakan berbagai bentuk kata yang memiliki makna sama, agar lebih konsisten dalam pengolahan teks oleh model. Pada tabel IV.5 terdapat kode program dari tahap stemming yang memiliki beberapa tahap di dalam nya yaitu, Potongan kode ini berfungsi untuk melakukan stemming pada kata-kata dalam teks. Fungsi `simple_stem` menerima satu input berupa kata (word), yang pertama-tama diubah menjadi huruf kecil untuk menjaga konsistensi. Selanjutnya, kode ini menghapus awalan umum dalam bahasa Indonesia, seperti `meng`, `mem`, `ber`, dan lainnya, yang seringkali tidak berkontribusi signifikan pada makna kata, misalnya “mengolah” diubah menjadi “olah”. Kemudian, kode ini juga menghapus akhiran umum seperti `-kan`, `-an`, dan `-i`, contohnya melakukan diubah menjadi laku. Proses ini bertujuan untuk menyederhanakan kata dan menyimpan bentuk dasarnya, sehingga meningkatkan kualitas model dalam memproses teks.

Tabel IV.6 Hasil Stemming

stopword	stemming
['oil', 'nefinary', 'pabrik', 'fasillitas', 'industri', 'mengolah', 'minyak', 'mentahmenjadi', 'produk', 'petrolium']	['oil', 'nefinary', 'pabrik', 'fasillitas', 'industr', 'olah', 'minyak', 'ntahmenjad', 'produk', 'petrolium']
['engraving', 'kartu', 'pos,', 'kalminir,', 'poster', 'reproduksi,percetakan,', 'lukisan', 'barang', 'cetakan', 'lainya', 'rekaman', 'mikro', 'film']	['engraving', 'kartu', 'pos,', 'kalminir,', 'poster', 'reproduks,ceta,', 'lukis', 'barang', 'ceta', 'lainya', 'rekam', 'mikro', 'film']
['menjaga', 'stabilitas', 'keuangan', 'menjaga', 'arus', 'kos']	['jaga', 'stabilitas', 'uang', 'jaga', 'arus', 'kos']

Pada tabel IV.6 terdapat tabel yangberisi hasil stopword yangmelalui tahapan stemming sehingga siap untukk masuk ke tahap tekonisasi.

4.1.3. Embedding

Tabel IV.7 Kode Program Embedding

	Kode program
1	`from gensim.models import KeyedVectors`
2	
3	`embedding_path = 'cc.id.300.vec'`
4	`embedding_model =`
5	`KeyedVectors.load_word2vec_format(embedding_path,`
6	`limit=50000)  # hanya load 50 ribu kata`
7	`print(embedding_model['perdagangan'])  # coba cek vektor`
8	`kata`

Pada tabel IV.7 terdapat kode embdding yangberfungsi untuk merubh kata menjadi angka yang bisa di proses oleh model. Berikut adlaah penjelasan dari kode di atas, kode ini digunakan untuk memuat pre-trained word embedding berbahasa Indonesia menggunakan pustaka Gensim. Modul `KeyedVectors` diimpor untuk membaca file embedding eksternal seperti FastText atau Word2Vec. File embedding yang digunakan adalah `cc.id.300.vec`, yang berisi representasi vektor berdimensi 300 untuk kata-kata dalam bahasa Indonesia. Untuk efisiensi memori, hanya 50.000 kata pertama yang dimuat ke dalam model menggunakan fungsi `load_word2vec_format`, karena format `.vec` ini kompatibel dengan Word2Vec-style embeddings. Embedding ini akan digunakan sebagai representasi numerik dari kata-kata dalam model pembelajaran mesin.

4.1.4 LSTM

Tabel IV.8 Kode Build Lstm

	Kode program
1	`def build_lstm_classifier():`
2	`    input_1 = Input(shape=(max_len,))`
3	`    input_2 = Input(shape=(max_len,))`
4	
5	`    embedding_layer = Embedding(input_dim=vocab_size,`
6	`                                output_dim=embedding_dim,`
7	`                                weights=[embedding_matrix],`
8	`                                trainable=False)`
9	
10	`    x1 = embedding_layer(input_1)`
11	`    x2 = embedding_layer(input_2)`
12	
13	`    shared_lstm = LSTM(64)`
14	
15	`    out1 = shared_lstm(x1)`
16	`    out2 = shared_lstm(x2)`
17	
18	`    merged = Concatenate()([out1, out2])`
19	`    dense = Dense(64, activation='relu')(merged)`
20	`    output = Dense(2, activation='softmax')(dense)`
21	
22	`    model = Model(inputs=[input_1, input_2], outputs=output)`
23	`    model.compile(optimizer='adam',`
24	`loss='categorical_crossentropy', metrics=['accuracy'])`
25	`    return model`

Pada tabel IV.8 terdapat kode program model arsitektur LSTM Classifier, berikut adalah penjelasan dari kode tabel yang ada di atas, Fungsi ini digunakan untuk membangun arsitektur model LSTM classifier yang bekerja dengan dua input, yaitu pasangan teks dari jawaban guru dan siswa yang telah diproses dan dipadatkan hingga panjang tertentu (`max_len`). Kedua input ini kemudian diterjemahkan ke dalam bentuk vektor angka menggunakan layer embedding, di mana bobot embedding berasal dari model FastText yang telah dimuat sebelumnya. Embedding ini diatur agar tidak dilatih ulang (trainable=False). Selanjutnya, kedua input melewati LSTM layer yang sama (shared) untuk menangkap informasi urutan dan konteks. Hasil dari kedua LSTM ini kemudian digabung dan diproses melalui fully-connected layer untuk mempelajari pola yang lebih kompleks. Output akhir

dihasilkan dari layer dense dengan dua neuron dan fungsi aktivasi softmax untuk menghasilkan probabilitas dari dua kelas: benar (1) dan salah (0). Model kemudian dikompilasi menggunakan optimizer Adam dan metrik akurasi sebagai acuan performa.

Tabel IV.9 Training Lstm

	code
1	`tokenize_pad = lambda texts:`
2	`pad_sequences(tokenizer.texts_to_sequences(texts),`
3	`maxlen=max_len)`
4	`pairs, labels = generate_pairs(df)`
5	
6	`X1 = tokenize_pad([x[0] for x in pairs])`
7	`X2 = tokenize_pad([x[1] for x in pairs])`
8	
9	`from tensorflow.keras.utils import to_categorical`
10	`y = to_categorical(labels, num_classes=2)`
11	
12	`from sklearn.model_selection import train_test_split`
13	`X1_train, X1_test, X2_train, X2_test, y_train, y_test =`
14	`train_test_split(`
15	`    X1, X2, y, test_size=0.2, random_state=42`
16	`)`
17	
18	`from sklearn.utils.class_weight import compute_class_weight`
19	`classes = np.array([0, 1])`
20	`weights = compute_class_weight(class_weight='balanced',`
21	`classes=classes, y=np.argmax(y_train, axis=1))`
22	`class_weights = dict(zip(classes, weights))`
23	
24	`model = build_lstm_classifier()`
25	`model.fit([X1_train, X2_train], y_train, batch_size=8,`
26	`epochs=10, class_weight=class_weights, verbose=1)`
27	
28	`model.save("lstm_classifier.h5")`
29	
30	
31	
32	
33	
34	

Pada tabel IV.9 terdapat kode program untuk training model lstm classifier, berikut adalah penjelasannya, Tahapan ini mencakup proses konversi teks menjadi urutan angka menggunakan tokenizer dan padding agar seluruh input memiliki panjang

yang seragam (`max_len`). Dataset kemudian disusun dalam bentuk pasangan input (jawaban siswa dan jawaban guru) beserta label kemiripannya (1 untuk mirip, 0 untuk tidak). Setelah di-tokenisasi dan dipad, data dibagi menjadi data latih dan data uji dengan rasio 80:20, untuk menguji kemampuan generalisasi model. Karena distribusi label bisa tidak seimbang, bobot kelas dihitung otomatis agar model tidak bias terhadap kelas tertentu. Model klasifikasi kemudian dibangun menggunakan fungsi arsitektur LSTM yang telah disiapkan. Proses pelatihan dilakukan selama 10 epoch dengan batch size 8 dan mempertimbangkan bobot kelas. Setelah selesai dilatih, model disimpan ke dalam file `.h5` untuk digunakan pada tahap prediksi selanjutnya.

4.1.5 MaLSTM

Tabel IV.10 Arsitektur Lstm

	Kode program

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
	`from tensorflow.keras.layers import Input, Embedding, LSTM, Lambda, Dense`
	`from tensorflow.keras.models import Model`
	`from tensorflow.keras import backend as K`
	
	`def exponent_neg_manhattan_distance(vects):`
	`  x, y = vects`
	`  return K.exp(-K.sum(K.abs(x - y), axis=1, keepdims=True))`
	
	`def build_malstm_model():`
	`  input_1 = Input(shape=(max_len,))`
	`  input_2 = Input(shape=(max_len,))`
	
	`  embedding_layer = Embedding(input_dim=vocab_size,`
	`  output_dim=embedding_dim,`
	`  weights=[embedding_matrix],`
	`  trainable=False)`
	
	`  encoded_1 = embedding_layer(input_1)`
	`  encoded_2 = embedding_layer(input_2)`
	
	`  shared_lstm = LSTM(50)`
	
	`  output_1 = shared_lstm(encoded_1)`
	`  output_2 = shared_lstm(encoded_2)`
	
	`  # Output similarity pakai sigmoid`
	`  malstm_distance = Lambda(exponent_neg_manhattan_distance,`

```
                                    output_shape=lambda x: (x[0][0],
1))([output_1, output_2])

    output = Dense(1, activation='sigmoid')(malstm_distance)

    model = Model(inputs=[input_1, input_2], outputs=output)
    model.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['accuracy'])
    return model
```

Pada tabel IV.10 terdapat kode program arsitektur MaLSTM, yang digunakan untuk mengukur tingkat kemiripan jawaban siswa dan kunci jawaban guru. Model MaLSTM dibangun menggunakan arsitektur Siamese Network dengan dua input berupa jawaban siswa dan guru. Setiap input diproses oleh LSTM berbagi bobot, lalu dihitung jarak Manhattan eksponensial negatif untuk menilai kemiripan antar teks. Hasil akhir berupa probabilitas kemiripan melalui sigmoid classifier. Berikut adalah penjelasan nya, Pada tahap ini dibangun arsitektur model MaLSTM (Manhattan LSTM), yang digunakan untuk mengukur kemiripan semantik antara dua input teks, yaitu jawaban siswa dan jawaban guru. Model dimulai dengan dua input, masing-masing berisi urutan token dari jawaban siswa dan guru yang telah diproses. Setiap token kemudian diterjemahkan menjadi vektor menggunakan layer embedding yang memanfaatkan bobot pre-trained dari FastText. Kedua input tersebut diproses oleh LSTM yang berbagi bobot (shared LSTM) agar arsitekturnya menjadi Siamese dan hasil representasi antar kalimat dapat dibandingkan secara langsung. Perbandingan dilakukan melalui fungsi Lambda yang menghitung jarak Manhattan, lalu dikonversi menjadi skor kemiripan menggunakan fungsi eksponensial negatif. Akhirnya, output dari model berupa probabilitas kemiripan antara dua jawaban dihasilkan melalui layer Dense dengan aktivasi sigmoid. Model ini dikompilasi dengan fungsi loss binary_crossentropy dan dioptimasi menggunakan algoritma Adam.

Tabel IV.11 Training Malstm

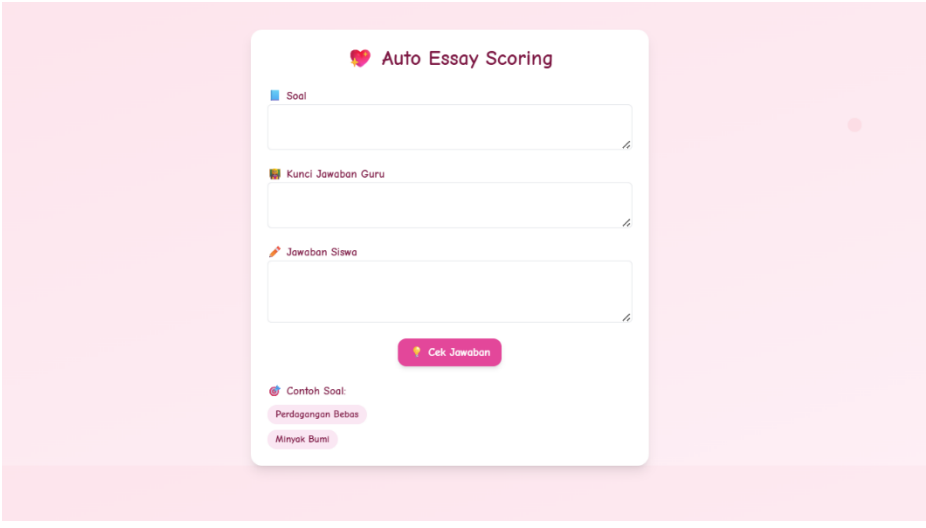
	Kode program

1
2
3
4
5
6
7
8
9
10
	`pairs, labels = generate_pairs(df)`
	`X1 = pad_sequences(tokenizer.texts_to_sequences([x[0] for x in pairs]), maxlen=max_len)`
	`X2 = pad_sequences(tokenizer.texts_to_sequences([x[1] for x in pairs]), maxlen=max_len)`
	`y = np.array(labels)`
	
	`model = build_malstm_model()`
	`model.fit([X1, X2], y, batch_size=8, epochs=10, verbose=1)`
	`model.save("malstm_model.h5")`

pada tabel IV.11 terdapat kode program training MaLSTM, tahap training MaLSTM dilakukan dengan membentuk pasangan data jawaban siswa dan jawaban guru, kemudian ditokenisasi dan diproses menjadi input berukuran seragam. Model dilatih untuk membedakan kemiripan teks menggunakan jarak Manhattan eksponensial dan menghasilkan probabilitas kemiripan. Model disimpan dalam format `.h5` untuk digunakan pada tahap prediksi. Berikut adalah penjelasan kode diatas, langkah awal pada bagian ini adalah membentuk pasangan data dari DataFrame berupa [jawaban_siswa, jawaban_guru] beserta label klasifikasinya. Kedua input teks tersebut kemudian diproses melalui tahap tokenisasi dan padding untuk menghasilkan representasi numerik yang memiliki panjang seragam, yaitu X1 untuk jawaban siswa dan X2 untuk jawaban guru. Label yang telah disiapkan diubah menjadi array NumPy agar dapat dibaca oleh model. Selanjutnya, model MaLSTM dibangun melalui fungsi `build_malstm_model()` dan dilatih menggunakan data input tersebut. Proses pelatihan dilakukan dengan batch size 8 dan jumlah epoch sebanyak 10 kali. Setelah pelatihan selesai, model disimpan dalam format `.h5` agar dapat digunakan kembali untuk keperluan inferensi atau integrasi dengan antarmuka web.

4.1.6 implementasi web

Sistem klasifikasi multi-label yang telah dibangun dengan menggunakan model final kemudian diterapkan ke dalam sebuah sistem antarmuka berbasis website. Website ini dirancang untuk mempermudah pengguna dalam melakukan klasifikasi terhadap data kritik dan saran mahasiswa.



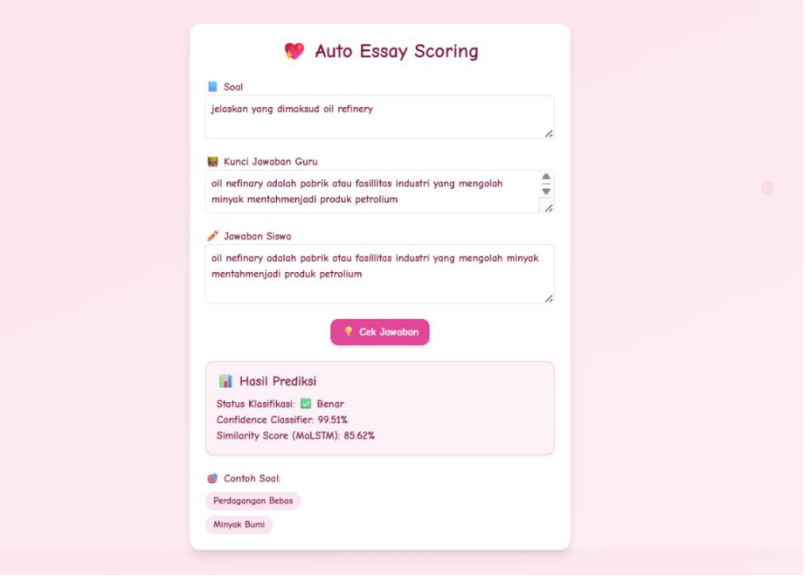
Gambar IV.2 Halaman Web

Pada Gambar IV.2 Gambar di atas menunjukkan tampilan antarmuka web dari sistem Automated Essay Scoring yang telah dikembangkan. Antarmuka ini dirancang dengan pendekatan sederhana dan user-friendly, agar mudah digunakan oleh guru maupun pengguna umum. Dalam tampilan utama, pengguna dapat mengisi tiga komponen input utama, yaitu Soal, Kunci Jawaban Guru, dan Jawaban Siswa. Ketiga input ini merupakan elemen yang dibutuhkan sistem untuk melakukan penilaian otomatis terhadap jawaban siswa.

Setelah seluruh kolom diisi, pengguna dapat menekan tombol “Cek Jawaban” untuk memulai proses penilaian. Sistem akan melakukan evaluasi berbasis model deep learning (MaLSTM dan LSTM Classifier), dan hasil klasifikasi—berupa prediksi benar atau salah, serta tingkat kepercayaan model—akan ditampilkan setelah proses selesai.

Untuk menjaga konsistensi input terhadap model deep learning yang telah dilatih, sistem menetapkan batas maksimal panjang input teks sebanyak **100 token**. Token DONE di sini mengacu pada representasi kata (setelah proses tokenisasi) yang digunakan sebagai input untuk model LSTM dan MaLSTM. Apabila pengguna (guru atau siswa) menginputkan teks yang lebih panjang dari 100 token, sistem secara otomatis memangkas bagian akhir teks tersebut. Sebaliknya, jika input lebih pendek, sistem akan melakukan *padding* untuk menyamakan panjang input. Batasan ini ditetapkan berdasarkan panjang input saat pelatihan model.

Dengan antarmuka yang intuitif dan responsif, sistem ini bertujuan mempermudah proses penilaian esai secara cepat dan objektif, serta mendukung pengajaran yang efisien di lingkungan pendidikan.



Gambar IV.3 Halaman Web Sistem

Gambar di atas merupakan tampilan halaman sistem setelah proses klasifikasi dilakukan. Pada bagian ini, pengguna telah menginput Soal, Kunci Jawaban Guru, dan Jawaban Siswa, kemudian menekan tombol “Cek Jawaban”. Sistem kemudian menampilkan hasil evaluasi secara otomatis pada panel Hasil Prediksi.

Informasi yang ditampilkan meliputi:

- Status Klasifikasi: Memberikan hasil prediksi apakah jawaban siswa dikategorikan sebagai Benar atau Salah.
- Confidence Classifier: Menunjukkan tingkat kepercayaan model klasifikasi terhadap prediksi yang diberikan, dalam hal ini ditampilkan dalam bentuk persentase.
- Similarity Score (MaLSTM): Menampilkan nilai kesamaan semantik antara jawaban siswa dan jawaban guru yang dihitung menggunakan arsitektur MaLSTM berbasis jarak Manhattan.

Tampilan ini tidak hanya memberikan hasil klasifikasi, namun juga menyertakan skor pendukung yang bersifat interpretatif agar pengguna (dalam hal ini guru atau evaluator) dapat memahami alasan di balik prediksi model.

Fitur seperti ini sangat membantu dalam menjembatani kecanggihan model deep learning dengan kejelasan hasil yang dibutuhkan oleh pengguna akhir, sehingga tetap transparan dan informatif.

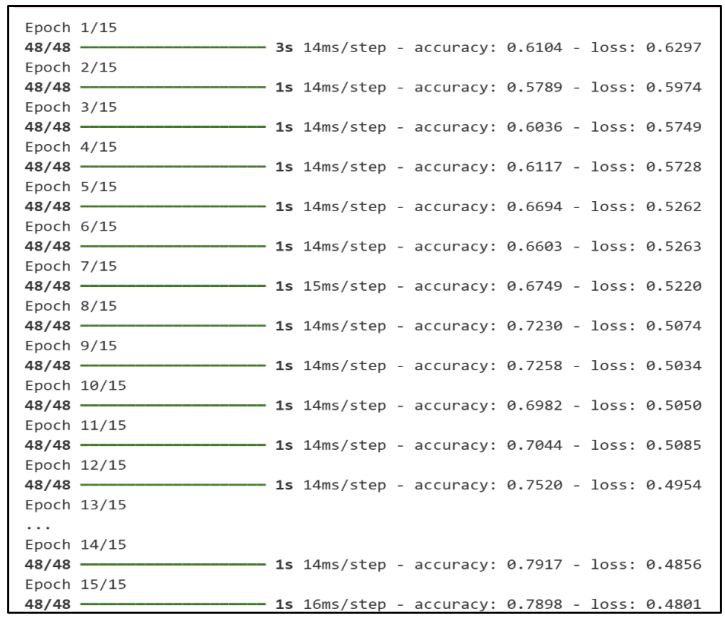
4.2. Hasil Pengujian

Penelitian ini bertujuan untuk menganalisis pengaruh preprocessing terhadap performa model Long Short-Term Memory (LSTM) dengan dua skenario pengujian utama, yaitu:

4.2.1 Skenario dengan Preprocessing

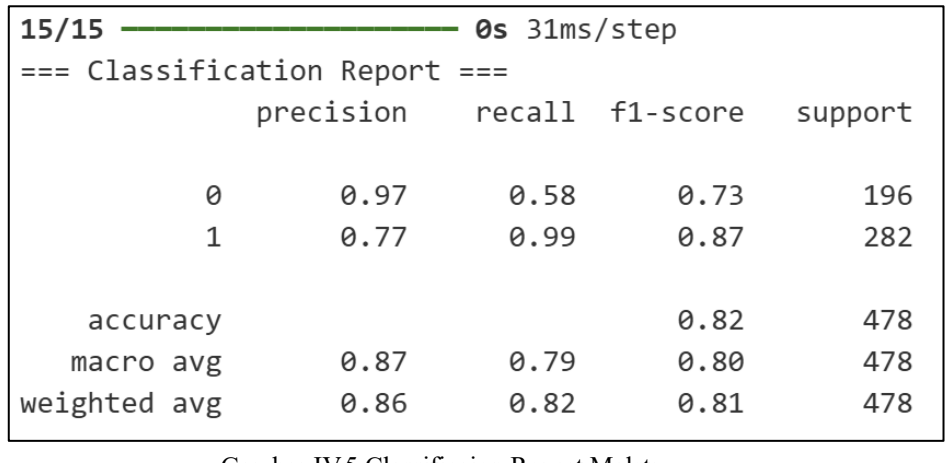
Pada skenario ini, proses pengujian dilakukan dengan menerapkan tahapan preprocessing terlebih dahulu terhadap data jawaban siswa. Tahapan preprocessing yang digunakan meliputi tokenisasi, penghapusan stopword, serta stemming. Tujuannya adalah untuk menghilangkan kata-kata tidak penting dan menyederhanakan struktur kata agar model dapat lebih mudah mengenali pola semantik antar kalimat.

Model yang digunakan adalah MaLSTM (Manhattan LSTM), yang dirancang untuk menghitung kemiripan antara dua kalimat, yaitu antara jawaban siswa dan jawaban guru. Setelah data diproses, pasangan kalimat diubah menjadi vektor, kemudian dilatih pada model MaLSTM dengan parameter tertentu seperti batch size, jumlah epoch, dan pembobotan kelas (class weight). Hasil pelatihan selanjutnya dievaluasi menggunakan metrik seperti akurasi dan classification report.



Gambar IV.4 Hasil Per Epoch Malstm

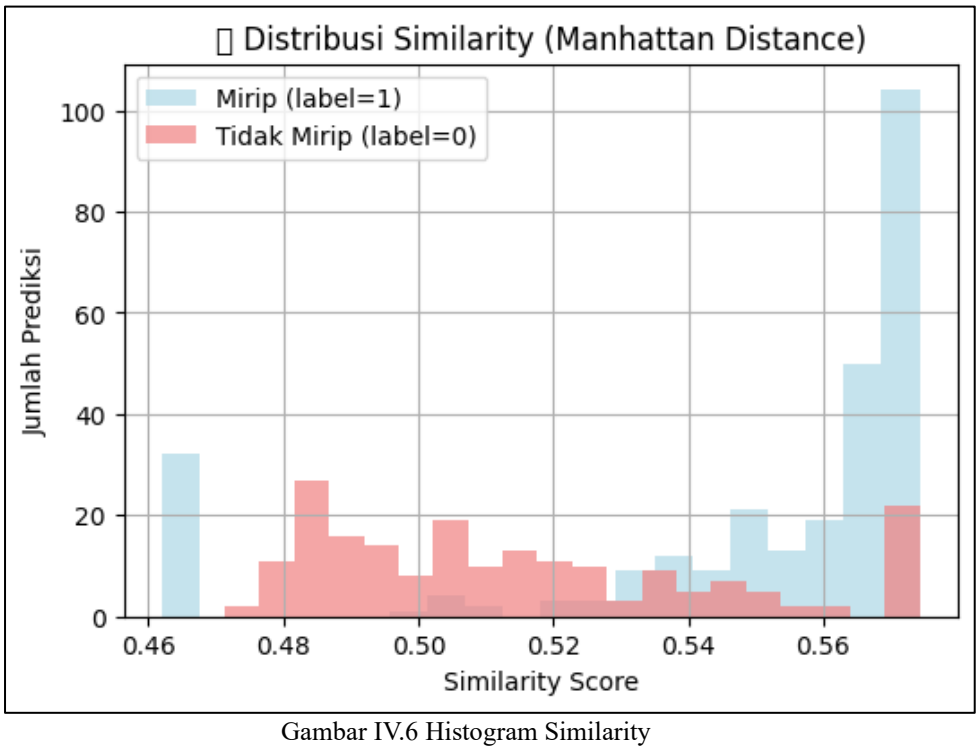
Pada Gambar IV.4 adalah proses pelatihan model MaLSTM dilakukan selama 15 epoch menggunakan dataset pasangan kalimat antara jawaban siswa dan kunci jawaban guru. Model menunjukkan peningkatan performa yang cukup stabil dari awal pelatihan hingga akhir. Pada epoch pertama, akurasi model masih berada di angka 61% dengan nilai loss sekitar 0.63. Seiring bertambahnya epoch, akurasi terus meningkat hingga mencapai sekitar 79% pada epoch ke-13 sampai ke-15, dengan penurunan nilai loss hingga 0.48. Hal ini mengindikasikan bahwa model semakin mampu membedakan pasangan jawaban yang mirip dan tidak mirip berdasarkan representasi teks yang dipelajari. Setelah proses pelatihan selesai, model disimpan dalam format HDF5 (.h5).



Gambar IV.5 Classificaion Report Malstm

Pada gambar IV.5 terdapat hasil evaluasi setelah model MaLSTM selesai dilatih, dilakukan evaluasi terhadap performa klasifikasi menggunakan confusion matrix dan classification report. Hasil evaluasi menunjukkan bahwa model memiliki **akurasi sebesar 82%**, yang berarti mayoritas prediksi terhadap pasangan jawaban

(mirip/tidak mirip) berhasil diklasifikasikan dengan tepat. Untuk kelas **0 (tidak mirip)**, precision-nya sangat tinggi yaitu 0.97, tetapi recall-nya masih tergolong rendah (0.58), yang menunjukkan bahwa model cukup konservatif dalam memutuskan bahwa dua jawaban tidak mirip. Sebaliknya, untuk kelas **1 (mirip)**, model menunjukkan recall sangat tinggi (0.99), namun precision-nya lebih rendah (0.77), menandakan bahwa model lebih sering mengklasifikasikan pasangan sebagai mirip, walaupun tidak semuanya benar. Nilai rata-rata (macro average) dari precision, recall, dan f1-score berada di angka **sekitar 0.80**, yang menunjukkan keseimbangan performa antar kelas. Secara keseluruhan, model cukup andal dalam mengenali jawaban siswa yang mirip dengan kunci, meskipun masih ada ruang untuk peningkatan pada deteksi jawaban yang tidak mirip.



Gambar IV.6 Histogram Similarity

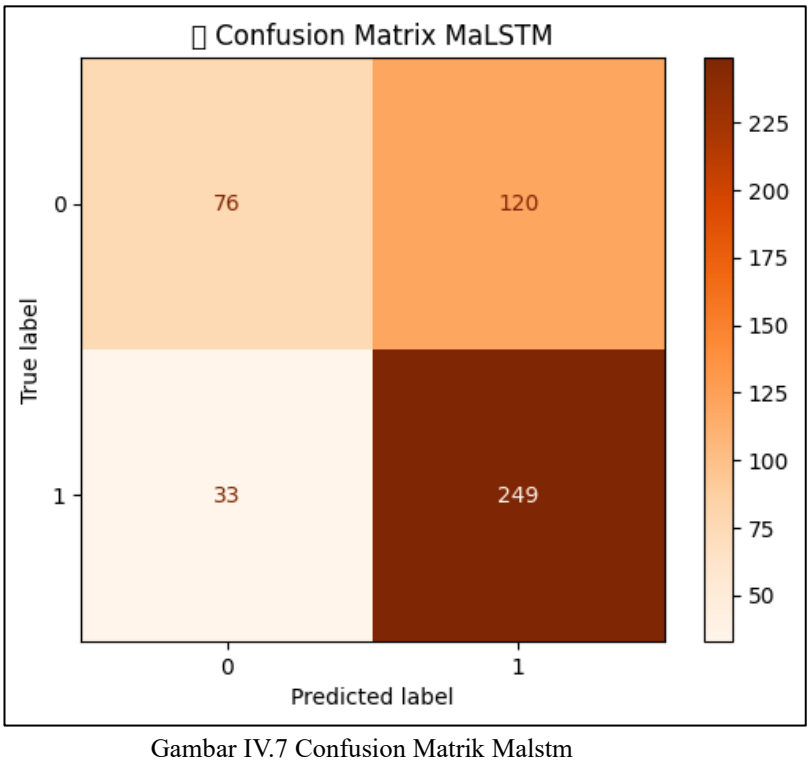
Pada gambar IV.6 Gambar di atas menunjukkan visualisasi distribusi skor similarity yang dihasilkan oleh model **MaLSTM** menggunakan **Manhattan Distance**. Histogram ini memberikan insight penting mengenai bagaimana model menilai kemiripan antara jawaban siswa dengan jawaban kunci berdasarkan representasi vektor kalimat.

Dalam grafik tersebut, dua warna digunakan untuk membedakan label asli dari data:

- Warna **biru muda** merepresentasikan prediksi yang dikategorikan sebagai *mirip* (label 1),
- Warna **merah muda** menandakan prediksi sebagai *tidak mirip* (label 0).

Dari pola distribusi yang muncul, dapat dilihat bahwa skor similarity untuk label 1 cenderung terkonsentrasi di angka **di atas 0.54**. Hal ini mengindikasikan bahwa model memiliki kecenderungan yang konsisten untuk memberikan nilai similarity tinggi terhadap pasangan jawaban yang memang semantik-nya mirip. Dengan kata lain, MaLSTM cukup yakin terhadap jawaban-jawaban yang benar.

Sebaliknya, skor similarity untuk label 0 justru lebih tersebar dan bervariasi, terutama di rentang **0.47 hingga 0.53**. Area ini bisa dianggap sebagai zona abu-abu atau *ambiguous zone*, di mana nilai similarity tidak terlalu rendah namun juga tidak cukup tinggi untuk disebut *mirip*. Di sinilah model kerap kali merasa "ragu" dan cenderung berpotensi melakukan kesalahan klasifikasi, khususnya false positive—di mana jawaban yang tidak mirip malah dianggap benar oleh model.



Gambar IV.7 Confusion Matrik Malstm

Pada gambar IV.7 Gambar di atas menunjukkan confusion matrix dari hasil prediksi model **MaLSTM** dalam mengklasifikasikan jawaban siswa sebagai *mirip* (label 1) atau *tidak mirip* (label 0) dengan jawaban guru. Matrix ini memberikan gambaran performa model berdasarkan empat kategori evaluasi:

- **True Positive (TP) = 249**
  Model berhasil memprediksi jawaban siswa sebagai *mirip*, dan faktanya memang mirip. Ini adalah prediksi yang tepat dan menjadi indikator keberhasilan model dalam mengidentifikasi kesesuaian jawaban.
- **True Negative (TN) = 76**
  Model memprediksi jawaban sebagai *tidak mirip* dan kenyataannya memang tidak mirip. Ini juga termasuk prediksi yang benar dan menunjukkan kemampuan model untuk mengenali jawaban yang tidak sesuai dengan kunci.
- **False Positive (FP) = 120**
  Model memprediksi jawaban siswa sebagai *mirip*, padahal seharusnya tidak mirip. Ini menunjukkan bahwa model terlalu optimis atau "terlalu memaklumi" kesamaan yang tidak substansial. Jumlah FP yang cukup besar ini bisa menjadi tanda bahwa model mengalami *overprediction* terhadap label 1.
- **False Negative (FN) = 33**
  Model memprediksi jawaban sebagai *tidak mirip*, padahal sebenarnya mirip. Hal ini menunjukkan bahwa model terkadang gagal mengenali kemiripan yang valid.

Secara umum, model MaLSTM menunjukkan performa yang kuat dalam mengenali jawaban siswa yang benar atau mirip dengan kunci jawaban (terlihat dari TP yang cukup tinggi, yaitu 249). Namun, model masih mengalami kesulitan dalam mendeteksi jawaban yang salah atau tidak mirip (dengan FP yang mencapai 120). Artinya, model cenderung menganggap terlalu banyak jawaban sebagai mirip meskipun sebenarnya tidak, yang bisa berdampak pada tingkat akurasi keseluruhan. Kondisi ini dapat terjadi karena representasi vektor antar kalimat yang mirip secara sintaksis tapi berbeda secara semantik masih dianggap relevan oleh model. Untuk

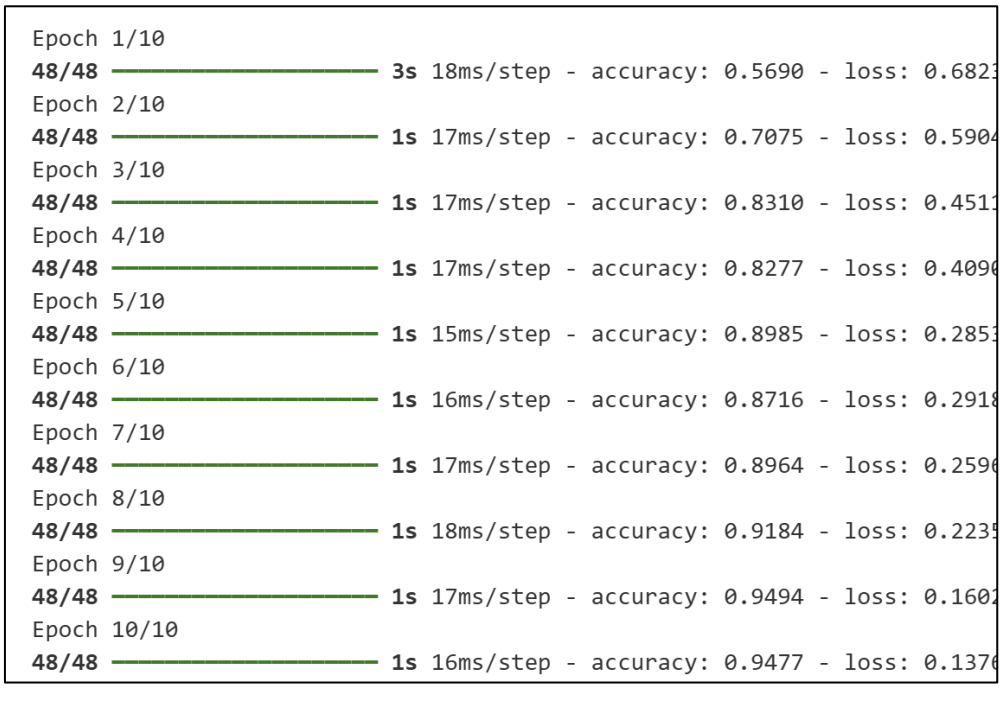
mengatasi hal ini, pengaturan threshold, peningkatan kualitas embedding, atau penyesuaian arsitektur model bisa menjadi alternatif pengembangan.

2.uji LSTM Classifier

Pada tahap ini, dilakukan pengujian terhadap model LSTM Classifier menggunakan data yang telah melalui proses preprocessing. Tahapan preprocessing meliputi tokenisasi, penghapusan stopword, dan stemming, yang bertujuan untuk menyederhanakan bentuk teks sehingga lebih mudah dipahami oleh model.

Model dilatih menggunakan data hasil preprocessing untuk mengklasifikasikan jawaban siswa ke dalam dua kategori, yaitu benar dan salah. Penggunaan preprocessing diharapkan dapat meningkatkan kualitas input teks yang diterima model, sehingga proses pembelajaran menjadi lebih efektif.

Hasil pengujian awal menunjukkan bahwa LSTM Classifier mampu belajar dengan baik dari data yang telah dibersihkan. Model menunjukkan potensi akurasi yang tinggi dan kemampuan klasifikasi yang stabil. Pengaruh preprocessing terlihat dari kualitas prediksi yang lebih akurat dibandingkan data mentah tanpa pembersihan.



Gambar IV.8 Epoch LSTM Classifier

Pada gambar IV.8 terdapat proses pelatihan model **LSTM Classifier** dilakukan selama 10 epoch dengan ukuran batch sebesar 8. Dari hasil training yang

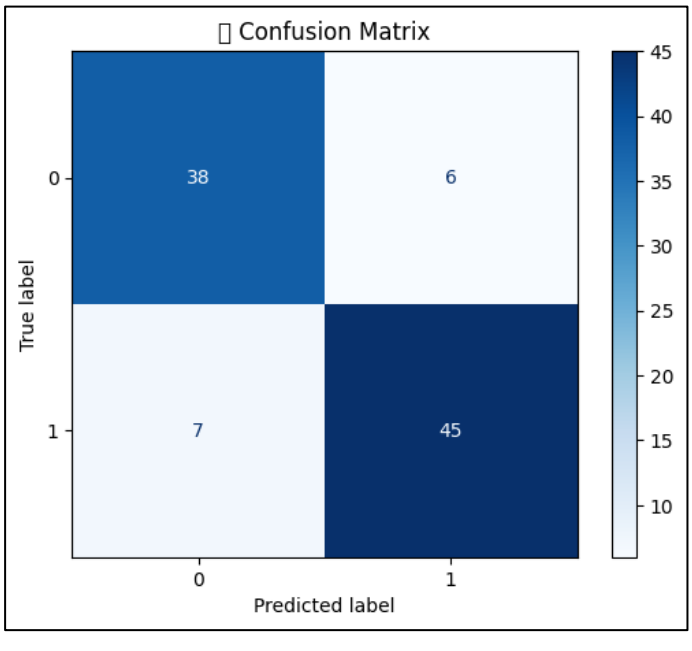
ditampilkan, terlihat adanya peningkatan performa model yang cukup signifikan seiring bertambahnya epoch. Pada awal pelatihan (epoch ke-1), akurasi model masih cukup rendah di angka **56.9%**, dengan nilai loss sebesar **0.6823**, yang menunjukkan bahwa model belum cukup mampu memahami pola data secara optimal.

Namun, seiring berjalannya epoch, performa model terus menunjukkan peningkatan stabil. Misalnya, pada epoch ke-3, akurasi melonjak menjadi **83.1%**, dengan penurunan loss ke angka **0.4511**. Ini menandakan bahwa model mulai berhasil mempelajari representasi jawaban dengan baik. Peningkatan ini terus berlanjut hingga mencapai **94.7%** akurasi di epoch ke-10, dan nilai loss juga terus menurun hingga mencapai **0.1376**, yang merupakan indikator bahwa model semakin yakin terhadap prediksi yang dihasilkannya.

Performa tinggi di akhir training ini menunjukkan bahwa model telah berhasil melakukan generalisasi terhadap pola data pelatihan dengan cukup baik. Namun demikian, perlu diingat bahwa akurasi tinggi selama pelatihan belum tentu menjamin performa serupa pada data uji.

Preprocessing terbukti sangat berpengaruh terhadap performa model, karena data yang lebih bersih memudahkan LSTM dalam mengenali struktur dan konteks kalimat dengan lebih akurat.

.



Gambar IV.9 Confusion Matrix Lstm Classifier

Pada gambar IV.9 terdapat Gambar di atas menampilkan **confusion matrix** dari hasil evaluasi model **LSTM Classifier** setelah pelatihan selama 10 epoch. Dari visual ini, kita bisa ngelihat seberapa baik model bisa membedakan jawaban siswa yang benar dan yang salah berdasarkan hasil klasifikasi dua kelas (0 = salah, 1 = benar). Berikut penjelasan komponen-komponennya:

- **True Positive (TP) = 45** → Model berhasil memprediksi jawaban siswa yang benar sebagai benar. Ini menunjukkan bahwa model mampu mengenali pola-pola yang sesuai dengan jawaban kunci atau rubric penilaian.
- **True Negative (TN) = 38** → Model juga cukup kuat dalam mendeteksi jawaban yang memang salah dan mengklasifikasikannya dengan tepat.
- **False Positive (FP) = 6** → Ada 6 kasus di mana model mengira jawaban itu benar, padahal seharusnya salah. Kesalahan jenis ini berisiko memberikan nilai pada jawaban yang tidak sesuai.
- **False Negative (FN) = 7** → Ada juga 7 kasus di mana model mengira jawaban itu salah, padahal sebenarnya benar. Ini berarti beberapa jawaban yang seharusnya mendapat nilai malah dianggap tidak layak.

Secara keseluruhan, model menunjukkan performa **yang cukup stabil dan seimbang** dalam mengenali kedua jenis jawaban (benar dan salah). Jumlah kesalahan prediksi (FP dan FN) juga terbilang **rendah**, yang menandakan bahwa model sudah mampu **belajar representasi fitur yang relevan** dari data pelatihan meskipun dengan jumlah data yang terbatas (hanya 96 pairs total).

Hal ini makin memperkuat bahwa pendekatan LSTM cukup efektif digunakan dalam sistem automated essay scoring berbasis klasifikasi. Namun, tetap ada ruang untuk perbaikan—misalnya dengan menambah variasi data atau tuning parameter lebih lanjut.

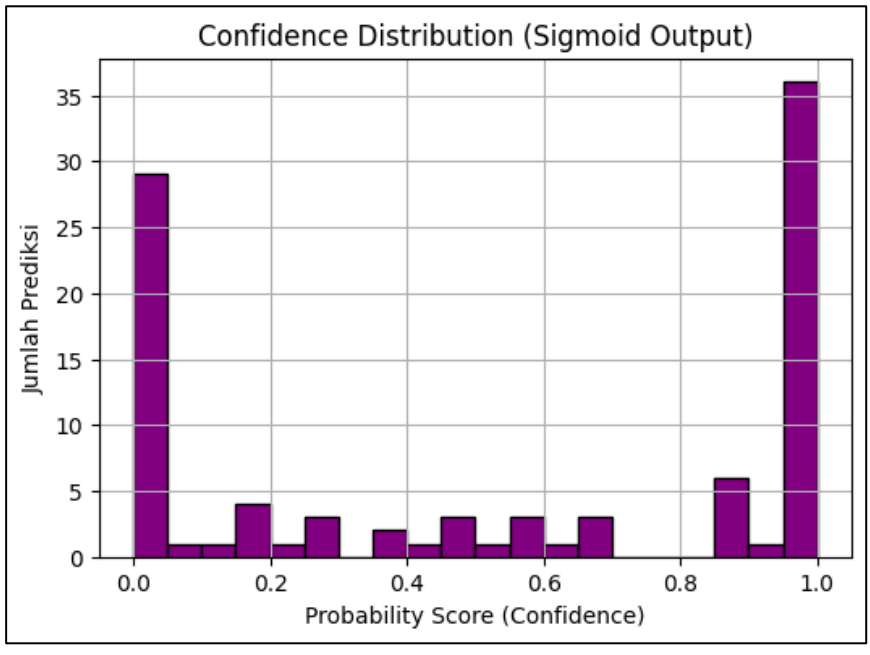
```
=== Classification Report ===
              precision    recall  f1-score   support

           0       0.84      0.86      0.85        44
           1       0.88      0.87      0.87        52

    accuracy                           0.86        96
   macro avg       0.86      0.86      0.86        96
weighted avg       0.86      0.86      0.86        96
```

Gambar IV.10 Classification Report Lstm Classifier

Pada gambar IV.10 Berdasarkan hasil evaluasi menggunakan classification report, model **LSTM Classifier** menunjukkan performa yang cukup baik dengan **akurasi keseluruhan sebesar 86%**. Pada label 0 (jawaban salah), model memperoleh **precision sebesar 0.84**, **recall sebesar 0.86**, dan **f1-score sebesar 0.85**. Artinya, model cukup konsisten dalam mengenali jawaban yang tidak sesuai dengan kunci atau rubrik penilaian. Sementara itu, untuk label 1 (jawaban benar), performanya juga sebanding dengan **precision sebesar 0.88**, **recall sebesar 0.87**, dan **f1-score sebesar 0.87**. Ini menunjukkan bahwa model mampu mengidentifikasi jawaban yang benar dengan tingkat kepercayaan yang tinggi.

Lebih lanjut, nilai **macro average dan weighted average untuk ketiga metrik utama (precision, recall, dan f1-score)** semuanya berada di angka **0.86**, yang mengindikasikan keseimbangan kinerja model di kedua kelas tanpa adanya bias yang signifikan terhadap salah satu label. Karena jumlah data relatif seimbang (44 data untuk label 0 dan 52 data untuk label 1), nilai rata-rata tertimbang (weighted) dan rata-rata makro (macro) pun tidak jauh berbeda.

Dengan performa ini, dapat disimpulkan bahwa LSTM Classifier memiliki potensi yang solid dalam membedakan antara jawaban benar dan salah dalam sistem automated essay scoring, meskipun model ini dilatih dengan jumlah data yang terbatas.



Gambar IV.11 Confidence Histogram Lstm Classifier

Pada Gambar IV.11 Gambar ini memperlihatkan distribusi confidence atau tingkat keyakinan dari prediksi model LSTM Classifier terhadap setiap pasangan jawaban yang diuji. Nilai confidence diambil dari output sigmoid model, yaitu rentang antara 0 dan 1, yang menggambarkan seberapa yakin model bahwa suatu jawaban tergolong sebagai “benar” (label 1).

Dari grafik histogram, tampak bahwa sebagian besar prediksi model berada di dua kutub ekstrem: mendekati 0.0 dan mendekati 1.0. Ini menandakan bahwa model memberikan prediksi dengan tingkat keyakinan tinggi — baik dalam menyatakan suatu jawaban itu salah (skor mendekati 0), maupun benar (skor mendekati 1). Sebaliknya, hanya sedikit prediksi yang berada di zona tengah (sekitar 0.4–0.6), yang menunjukkan bahwa model jarang berada dalam kondisi "ragu" atau tidak yakin terhadap keputusannya.

Fenomena ini cukup umum dalam model klasifikasi biner yang terlatih dengan baik, di mana model belajar secara jelas perbedaan pola dari masing-masing kelas. Hal ini bisa menjadi indikasi positif bahwa model mampu membedakan dengan tajam antara jawaban yang benar dan yang salah berdasarkan fitur-fitur dari embedding dan arsitektur LSTM.

Distribusi ini menyiratkan bahwa model sangat percaya diri dalam membuat keputusan. Dalam konteks automated essay scoring, hal ini bisa sangat berguna

untuk memastikan konsistensi penilaian dan mengurangi ambiguitas. Namun, tingginya tingkat confidence juga perlu diwaspadai karena bisa menjadi gejala **overconfidence** — yaitu model terlalu yakin terhadap prediksinya, bahkan saat menghadapi data yang sebenarnya ambigu atau tidak jelas secara semantik.

4.2.2 Skenario Tanpa Preprocessing

1.uji MaLSTM

```
Epoch 1/15
48/48 ━━━━━━━━━━━━━━━━━━━━ 3s 17ms/step - accuracy: 0.6141 - loss: 0.6062
Epoch 2/15
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - accuracy: 0.6198 - loss: 0.5858
Epoch 3/15
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - accuracy: 0.6309 - loss: 0.5449
Epoch 4/15
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.6456 - loss: 0.5512
Epoch 5/15
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.6787 - loss: 0.5328
Epoch 6/15
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.6659 - loss: 0.5170
Epoch 7/15
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.6962 - loss: 0.5041
Epoch 8/15
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.7096 - loss: 0.4922
Epoch 9/15
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.6870 - loss: 0.4883
Epoch 10/15
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.7304 - loss: 0.4748
Epoch 11/15
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.7093 - loss: 0.4939
Epoch 12/15
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.7321 - loss: 0.4745
Epoch 13/15
...
Epoch 14/15
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.7950 - loss: 0.4438
Epoch 15/15
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.8256 - loss: 0.4431
```

Gambar IV.12 Epoch Malstm Tanpa Preprocessing

Pada gambar IV.12 Model MaLSTM tahap pengujian model MaLSTM tanpa preprocessing, model dilatih selama 15 epoch dengan seluruh data pasangan jawaban yang tersedia. Hasil pelatihan menunjukkan adanya peningkatan akurasi secara bertahap seiring dengan bertambahnya epoch. Di awal pelatihan, model mencatat akurasi sebesar 61,41% dengan nilai loss 0.6062. Seiring proses pelatihan berlangsung, akurasi model terus meningkat dan mencapai titik tertinggi sebesar 82,56% pada epoch ke-15, disertai penurunan nilai loss menjadi 0.4431. Hal ini menunjukkan bahwa meskipun data tidak melalui proses pembersihan seperti penghapusan stopword atau stemming, model masih mampu mempelajari pola-pola

jawaban siswa yang relevan untuk membedakan jawaban benar dan salah secara cukup efektif.

Secara keseluruhan, tren akurasi dan penurunan nilai loss yang stabil menunjukkan bahwa model MaLSTM tetap dapat bekerja dengan baik meskipun tanpa preprocessing. Namun, hasil ini juga mengindikasikan bahwa ada ruang untuk perbaikan lebih lanjut, seperti pengujian threshold prediksi atau evaluasi lanjutan dengan visualisasi confusion matrix dan distribusi similarity untuk memastikan apakah ketidakhadiran preprocessing berdampak pada overfitting atau kesalahan klasifikasi tertentu. Hal ini penting aga

```
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
=== Classification Report (Threshold = 0.5) ===
              precision    recall  f1-score   support

           0       0.84      0.56      0.67       196
           1       0.75      0.93      0.83       282

    accuracy                           0.78       478
   macro avg       0.80      0.74      0.75       478
weighted avg       0.79      0.78      0.76       478
```

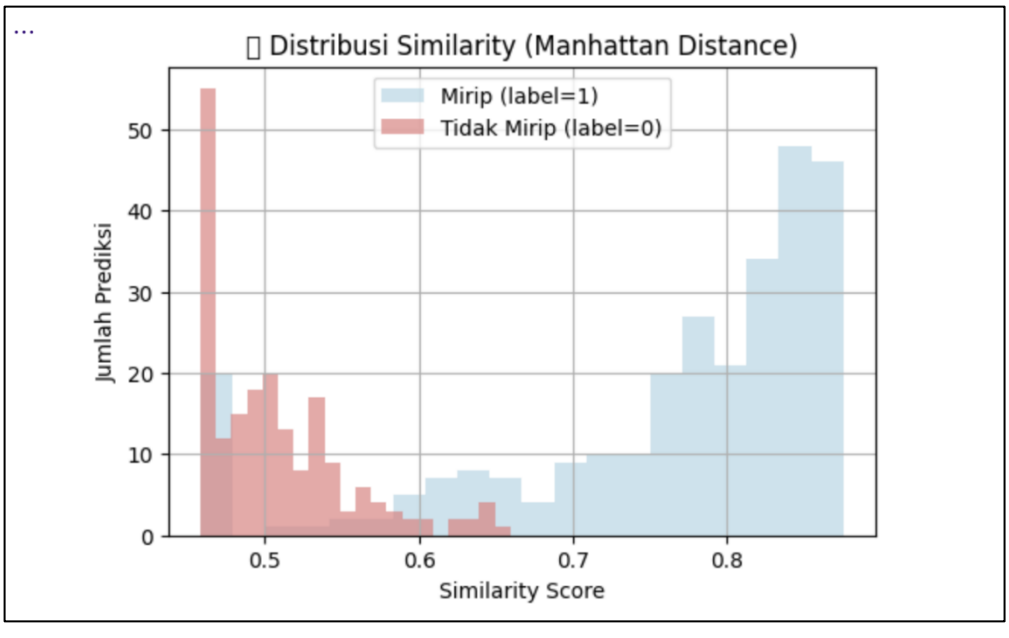
Gambar IV.13 Classification Report Malstm Tanpa Preprocessing

Pada gambar IV.13 Hasil evaluasi model MaLSTM tanpa preprocessing menggunakan classification report menunjukkan performa yang cukup baik meskipun tidak melalui tahapan pembersihan teks seperti stopword removal maupun stemming. Dengan threshold default sebesar 0.5, model mencapai akurasi keseluruhan sebesar **78%**. Pada kelas label 1 (jawaban benar/mirip), model mencatat nilai **precision 0.75** dan **recall 0.93**, menghasilkan **f1-score sebesar 0.83**. Ini menunjukkan bahwa model sangat baik dalam mengenali jawaban yang semestinya diklasifikasikan sebagai benar.

Namun, untuk kelas label 0 (jawaban salah/tidak mirip), performa model terlihat menurun, khususnya pada aspek recall yang hanya mencapai **0.56**, artinya masih banyak jawaban salah yang justru diklasifikasikan sebagai benar. Nilai **f1-score kelas 0 sebesar 0.67** memperlihatkan bahwa ketidakseimbangan ini cukup

signifikan. Distribusi ini berdampak pada **macro average f1-score sebesar 0.75** dan **weighted average f1-score sebesar 0.76**, yang menunjukkan kecenderungan model untuk lebih “percaya” terhadap jawaban yang dianggap benar, mirip seperti kecenderungan overpredict di label 1.

Secara keseluruhan, meskipun tidak menggunakan preprocessing, model tetap menunjukkan kemampuan generalisasi yang cukup baik. Akan tetapi, performa terhadap jawaban yang seharusnya tidak benar masih menjadi tantangan yang terlihat dari banyaknya false positive. Hal ini mengindikasikan perlunya strategi evaluasi tambahan seperti penyesuaian threshold, regularisasi, atau penguatan data label 0 agar performa bisa lebih seimbang di kedua kelas.

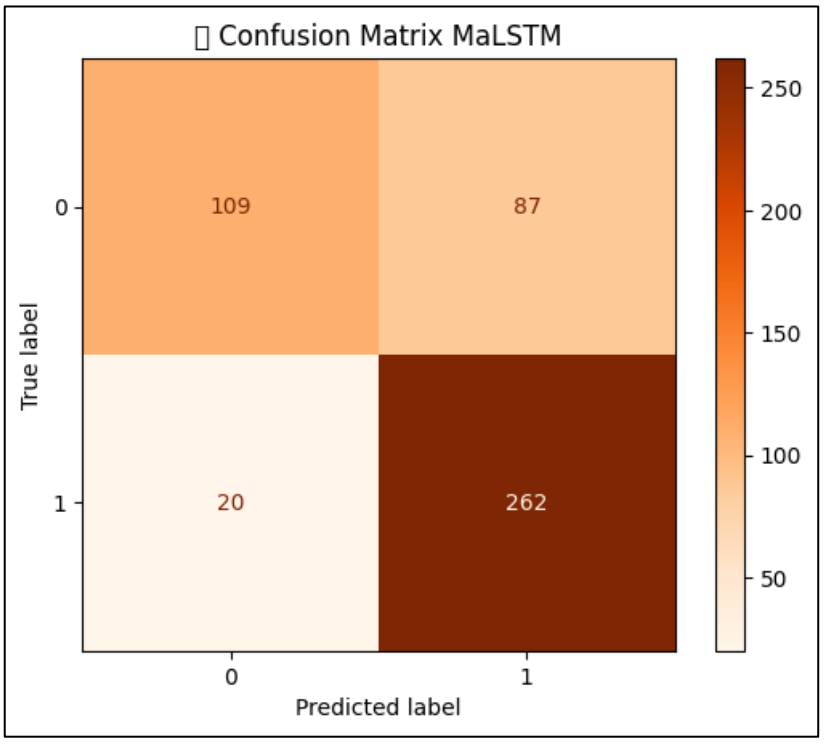


Gambar IV.14 Histogram Malstm Tanpa Preprocessing

Pada gambar IV.14 tahap evaluasi model MaLSTM tanpa preprocessing, Gambar ini menyajikan histogram yang menggambarkan distribusi skor kemiripan antara dua kategori, yaitu "Mirip" (label 1) dan "Tidak Mirip" (label 0), yang diukur menggunakan jarak Manhattan. Pada sumbu Y, jumlah prediksi untuk setiap kategori kemiripan ditampilkan, sementara sumbu X menunjukkan skor kemiripan yang berkisar antara 0.5 hingga 0.8. Histogram ini menunjukkan bahwa terdapat lebih banyak prediksi untuk kategori "Mirip" di kisaran skor kemiripan yang lebih tinggi,

sedangkan kategori "Tidak Mirip" memiliki distribusi yang cenderung lebih tersebar dengan beberapa puncak pada setiap skor.

- **Mirip (label=1)**: Ditunjukkan dengan warna biru, memiliki distribusi yang lebih tersebar pada kisaran skor kemiripan yang lebih tinggi, menunjukkan bahwa banyak objek atau data dianggap mirip satu sama lain.
- **Tidak Mirip (label=0)**: Ditunjukkan dengan warna merah muda, lebih terkonsentrasi pada nilai yang lebih rendah, menandakan bahwa objek dalam kategori ini cenderung memiliki kesamaan yang lebih sedikit.



Gambar IV.15 Confusion Mtriks Malstm

Confusion matrix berikut menyajikan evaluasi performa model MaLSTM dalam mengklasifikasikan jawaban siswa sebagai "Benar" (label 1) atau "Salah" (label 0) tanpa melalui proses preprocessing data. Matriks ini menggambarkan hubungan antara label asli (true label) dan prediksi model (predicted label).

Isi Matriks:

True Positive (TP): 262 — Jawaban benar yang berhasil diprediksi sebagai benar.

True Negative (TN): 109 — Jawaban salah yang berhasil diprediksi sebagai salah.

False Positive (FP): 87 — Jawaban salah yang salah diklasifikasikan sebagai benar.

False Negative (FN): 20 — Jawaban benar yang salah diklasifikasikan sebagai salah.

Model MaLSTM menunjukkan performa yang cukup baik secara keseluruhan, dengan jumlah klasifikasi benar (TP + TN) sebesar 371 dari 478 data uji, menghasilkan akurasi sekitar 77.6%.

Nilai precision untuk kelas Benar cukup tinggi, menandakan bahwa sebagian besar prediksi positif memang benar adanya. Begitu pula recall model dalam mendeteksi jawaban yang benar juga tergolong baik, walaupun masih terdapat sejumlah false negative dan false positive.

Model berhasil mengidentifikasi kemiripan semantik antar teks meskipun data yang digunakan belum dibersihkan (tanpa stopword removal, stemming, dll). Namun, keberadaan 87 data salah yang diprediksi benar dan 20 data benar yang diprediksi salah menunjukkan masih ada ruang perbaikan.

2. Uji LSTM

	precision	recall	f1-score	support
0	0.84	0.84	0.84	44
1	0.87	0.87	0.87	52
accuracy			0.85	96
macro avg	0.85	0.85	0.85	96
weighted avg	0.85	0.85	0.85	96

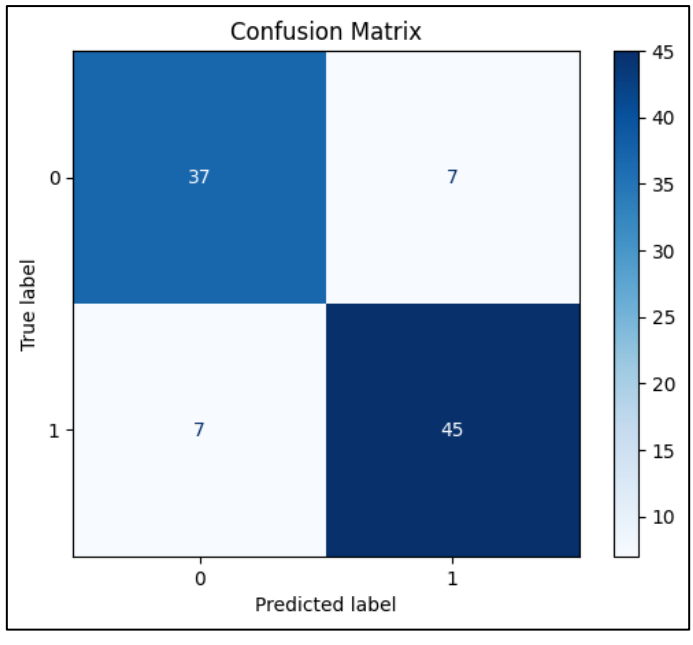
Gambar IV.16 Clasification Report Lstm

Berdasarkan hasil evaluasi menggunakan classification report pada gambar IV.16 terhadap model LSTM Classifier tanpa preprocessing, diperoleh performa yang cukup baik dengan akurasi keseluruhan sebesar 85%. Model menunjukkan kinerja yang seimbang antara kedua kelas, baik pada label 0 (jawaban salah) maupun label 1 (jawaban benar). Untuk label 0, nilai precision, recall, dan f1-score masing-masing sebesar 0.84. Ini mengindikasikan bahwa model mampu mengenali jawaban yang tidak sesuai dengan rubric penilaian secara konsisten.

Sementara itu, untuk label 1, model mencatat precision dan recall sebesar 0.87, dengan f1-score yang juga mencapai 0.87. Hasil ini menunjukkan bahwa model

sangat efektif dalam mengidentifikasi jawaban siswa yang sesuai dengan kunci jawaban. Nilai rata-rata makro (macro average) dan rata-rata tertimbang (weighted average) untuk ketiga metrik utama juga berada di angka 0.85, yang mencerminkan keseimbangan performa tanpa bias dominan terhadap salah satu kelas.

Secara keseluruhan, hasil ini memperlihatkan bahwa meskipun tanpa tahap preprocessing seperti stopword removal dan stemming, model LSTM Classifier tetap mampu melakukan klasifikasi dengan cukup akurat dan stabil. Ini menunjukkan bahwa arsitektur model cukup robust terhadap variasi bentuk teks dalam jawaban siswa.



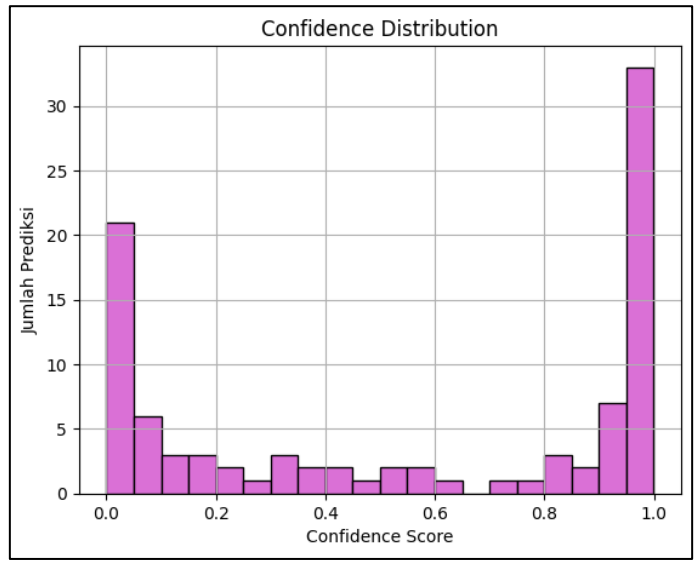
Gambar IV.17 Confusion Matrix Lstm

Hasil evaluasi model LSTM Classifier tanpa preprocessing ditampilkan dalam bentuk confusion matrix seperti pada Gambar IV.17 Dari total 96 data uji, model berhasil mengklasifikasikan 37 jawaban salah (label 0) dan 45 jawaban benar (label 1) dengan tepat. Namun, terdapat 7 jawaban salah yang salah diklasifikasikan sebagai benar (false positive), dan 7 jawaban benar yang salah diklasifikasikan sebagai salah (false negative).

Secara keseluruhan, model mencatat akurasi sebesar 85%, dengan precision dan recall yang seimbang pada kedua kelas. Nilai f1-score untuk label 0 adalah 0.84 dan untuk label 1 adalah 0.87, yang menunjukkan kemampuan model dalam mengenali

kedua kategori dengan baik. Distribusi kesalahan yang relatif kecil antara false positive dan false negative juga mencerminkan kestabilan performa model, meskipun tidak dilakukan preprocessing pada data teks.

Visualisasi ini menunjukkan bahwa model LSTM tetap mampu melakukan klasifikasi yang akurat dan andal dalam mendeteksi jawaban benar dan salah, bahkan tanpa tahap pembersihan data seperti stopword removal atau stemming. Dengan demikian, arsitektur LSTM menunjukkan robust-ness yang cukup baik terhadap variasi jawaban siswa dalam bentuk mentah.



Gambar IV.18 Matrix Confidence

Gambar IV.xx menunjukkan distribusi nilai confidence score yang dihasilkan oleh model LSTM Classifier tanpa preprocessing. Histogram ini menggambarkan seberapa yakin model dalam memberikan prediksi terhadap masing-masing jawaban siswa.

Distribusi menunjukkan bahwa sebagian besar prediksi berada pada dua kutub ekstrem, yaitu di kisaran 0.0–0.1 dan 0.9–1.0. Hal ini menandakan bahwa model cenderung memberikan prediksi dengan tingkat keyakinan yang sangat tinggi, baik saat memprediksi jawaban sebagai salah (label 0) maupun benar (label 1). Jumlah prediksi dengan confidence mendekati 1.0 cukup dominan, mengindikasikan bahwa model sangat yakin dalam mengklasifikasikan sebagian besar jawaban benar. Sebaliknya, prediksi dengan confidence rendah (dekat 0.0) juga cukup banyak, merepresentasikan keyakinan tinggi dalam mengklasifikasikan jawaban salah.

Distribusi ini memperlihatkan bahwa model memiliki kecenderungan confidence yang kuat dan jarang ragu dalam prediksinya, yang bisa jadi positif karena mengindikasikan ketegasan model. Namun, hal ini juga perlu diwaspadai karena prediksi yang terlalu yakin bisa mengabaikan kemungkinan error, terutama ketika berhadapan dengan jawaban yang ambigu atau berada di batas klasifikasi.

Dengan confidence level yang cenderung ekstrem ini, model menunjukkan kepercayaan tinggi terhadap output-nya, sekaligus membuka ruang untuk evaluasi tambahan seperti threshold adjustment guna mengontrol prediksi yang terlalu "percaya diri".

4.2.3 Pengaruh Rasio Data

Pengaruh Rasio Data: Pengujian ini bertujuan untuk menganalisis pengaruh rasio data latih dan data uji terhadap akurasi model. Beberapa rasio yang diuji adalah 80:20, 70:30, dan 60:40. Setiap rasio diuji untuk menentukan komposisi terbaik yang memberikan akurasi optimal.

1. LSTM

Tabel IV.12 Kode Program Uji Rasio LSTM

	Kode program

1
2
3
4
5
6
	`# Split manual`
	`from sklearn.model_selection import train_test_split`
	`X1_train, X1_test, X2_train, X2_test, y_train, y_test =`
	`train_test_split(`
	`    X1, X2, y, test_size=0.4, random_state=42`
	`)`

Pada tabel IV.14 di atas digunakan untuk membagi data menjadi dua bagian, yaitu data latih dan data uji, dengan memanfaatkan fungsi `train_test_split` dari library `scikit-learn`. Dalam implementasinya, input model yang terdiri dari X1 dan X2, serta label y, dipisahkan menjadi X1_train, X1_test, X2_train, X2_test, y_train, dan y_test. Pada contoh ini, parameter `test_size=0.4` digunakan untuk menentukan bahwa 40% data akan digunakan sebagai data uji, sementara sisanya 60% dipakai untuk pelatihan. Parameter random_state=42 berfungsi untuk menjaga agar pembagian data dilakukan secara konsisten setiap kali kode dijalankan, sehingga hasil eksperimen dapat direproduksi.

Meskipun pada contoh ini digunakan rasio 60:40, nilai `test_size` dapat diubah sesuai kebutuhan untuk pengujian lain seperti 0.2 (80:20) atau 0.3 (70:30), tanpa perlu menulis ulang seluruh potongan kode. Hal ini menjadikan satu blok kode ini cukup fleksibel untuk berbagai skenario pengujian dan analisis performa model.

Tabel IV.13 Hasil Uji Data LSTM

Rasio Train:Test	Precision (Kelas 0)	Recall (Kelas 0)	F1-Score (Kelas 0)	Precision (Kelas 1)	Recall (Kelas 1)	F1-Score (Kelas 1)	Accuracy
80 : 20	0.87	0.48	0.62	0.69	0.94	0.79	0.73
70 : 30	0.83	0.66	0.74	0.78	0.90	0.84	0.80
60 : 40	0.78	0.65	0.71	0.82	0.90	0.86	0.81

Pada tabel IV.15 terdapat pengaruh rasio pembagian data terhadap performa model LSTM Classifier, dilakukan eksperimen dengan tiga rasio berbeda: **80:20**, **70:30**, dan **60:40**. Hasil evaluasi menunjukkan bahwa model secara umum mampu memberikan hasil klasifikasi yang cukup baik di semua rasio, namun ada perbedaan kinerja yang menarik untuk dianalisis.

Pada rasio **80:20**, model memiliki **precision tinggi untuk kelas 0 (0.87)** namun nilai **recall-nya rendah (0.48)**, yang berarti model cukup hati-hati dalam memprediksi jawaban salah. Sebaliknya, **kelas 1 (jawaban benar)** memiliki recall yang sangat tinggi (0.94), namun precision-nya sedikit lebih rendah (0.69). Hal ini membuat model cenderung lebih **percaya diri dalam memprediksi jawaban benar**, meskipun beberapa prediksi salah ikut dianggap benar.

Ketika proporsi data uji diperbesar ke **70:30**, model menjadi lebih stabil. Nilai precision dan recall untuk **kelas 0** meningkat dan menjadi lebih seimbang (precision 0.83, recall 0.66), sementara **kelas 1** juga tetap tinggi (precision 0.78, recall 0.90). Akurasi keseluruhan juga meningkat menjadi **80%**, menunjukkan bahwa model makin adaptif terhadap variasi data.

Pada rasio **60:40**, performa model tetap konsisten dan bahkan sedikit meningkat dalam hal akurasi (**81%**). Precision dan recall pada kedua kelas cukup stabil, dengan F1-score tertinggi di kelas 1 (0.86). Ini menunjukkan bahwa meskipun data latih lebih sedikit, model tetap bisa menjaga performa berkat generalisasi yang cukup baik.

Secara keseluruhan, model LSTM Classifier menunjukkan performa yang stabil dan andal di berbagai rasio data, dengan kecenderungan lebih baik saat proporsi data uji lebih besar (70:30 dan 60:40). Hal ini menandakan bahwa model cukup fleksibel dalam mempelajari pola-pola dari data, dan mampu beradaptasi terhadap variasi ukuran data training tanpa penurunan performa signifikan.

4.2.4 Pengaruh Parameter LSTM

1. MaLSTM

Tabel IV.14 Pengaruh Parameeter

	Kode program

```
param_combinations = [
    (32, 1, 0.001),
    (64, 1, 0.001),
    (128, 1, 0.001),
    (64, 2, 0.001),
    (64, 1, 0.0001),
    (128, 2, 0.001),
    (64, 2, 0.0001),
    (32, 2, 0.001),
    (128, 1, 0.0005),
    (64, 3, 0.001),
]
results = []
for units, layers, lr in param_combinations:
    print(f"\n Testing LSTM with units={units},
layers={layers}, lr={lr}")
    model = build_lstm_classifier(units=units,
layers=layers, learning_rate=lr)
    model.fit([X1_train, X2_train], y_train, batch_size=8,
epochs=5, verbose=0)

    pred = model.predict([X1_test, X2_test])
    pred_binary = (pred[:, 1] >= 0.5).astype(int)

    report = classification_report(np.argmax(y_test,
axis=1), pred_binary, output_dict=False)
```

```
    print(report)
```

Pada tabel IV. ini, dilakukan pengujian terhadap berbagai kombinasi parameter pada model MaLSTM untuk melihat pengaruhnya terhadap performa model. Parameter yang diuji mencakup jumlah unit memori pada LSTM, jumlah lapisan (layers), dan nilai learning rate. Semua kombinasi parameter tersebut dimasukkan ke dalam list param_combinations, yang berisi tuple dengan format (units, layers, learning_rate). Setelah itu, dilakukan iterasi menggunakan loop for untuk setiap kombinasi parameter. Di dalam setiap iterasi, model MaLSTM dibangun ulang dengan menggunakan fungsi build_lstm_classifier() yang menerima input parameter konfigurasi tersebut. Kemudian model dilatih (fit) dengan data training yang sudah dipisahkan sebelumnya selama 5 epoch dengan batch_size sebesar 8, dan verbose=0 agar training berjalan diam-diam tanpa output berlebihan.

Setelah model selesai dilatih, proses prediksi dilakukan terhadap data uji (X1_test, X2_test). Hasil prediksi berbentuk probabilitas dari output sigmoid, sehingga dilakukan thresholding dengan mengambil nilai kolom ke-1 (kelas “1”) dan membandingkannya terhadap nilai 0.5 untuk menghasilkan output biner (0 atau 1). Terakhir, hasil prediksi dievaluasi menggunakan classification_report() dari sklearn untuk menampilkan metrik evaluasi seperti precision, recall, f1-score, dan akurasi. Hasil dari tiap kombinasi parameter akan langsung dicetak ke terminal agar bisa dilihat dan dibandingkan performanya secara langsung.

Tabel IV.15hasil Uji Parameter Lewat Jupyter

Unit Memori	Jumlah Layer	Learning Rate	Accuracy	Precision (0/1)	Recall (0/1)	F1-Score (0/1)
32	1	0.001	0.68	0.79 / 0.65	0.41 / 0.91	0.54 / 0.76
64	1	0.001	0.70	0.80 / 0.67	0.44 / 0.91	0.57 / 0.77
128	1	0.001	0.72	0.81 / 0.68	0.48 / 0.91	0.60 / 0.78
64	2	0.001	0.68	0.79 / 0.65	0.41 / 0.91	0.54 / 0.76
64	1	0.0001	0.62	1.00 / 0.59	0.15 / 1.00	0.26 / 0.74

Unit Memori	Jumlah Layer	Learning Rate	Accuracy	Precision (0/1)	Recall (0/1)	F1-Score (0/1)
128	2	0.001	0.68	0.83 / 0.65	0.37 / 0.94	0.51 / 0.77
64	2	0.0001	0.62	1.00 / 0.59	0.15 / 1.00	0.26 / 0.74
32	2	0.001	0.63	0.62 / 0.64	0.48 / 0.76	0.54 / 0.69
128	1	0.0005	0.67	0.77 / 0.64	0.37 / 0.91	0.50 / 0.75
64	3	0.001	0.68	0.79 / 0.65	0.41 / 0.91	0.54 / 0.76

Pada tabel IV.17 pengujian parameter terhadap model MaLSTM dilakukan untuk mengevaluasi bagaimana pengaruh variasi konfigurasi terhadap performa klasifikasi. Parameter yang diuji meliputi jumlah unit memori (32, 64, dan 128), jumlah layer (1, 2, dan 3), serta nilai learning rate (0.001, 0.0005, dan 0.0001). Pengujian dilakukan pada data yang telah dibagi dengan rasio 60:40 menggunakan 10 kombinasi parameter berbeda.

Secara umum, performa terbaik dicapai oleh konfigurasi dengan 128 unit memori, 1 layer, dan learning rate 0.001, dengan akurasi sebesar 72%. Model dengan konfigurasi tersebut menunjukkan keseimbangan antara recall dan precision pada kedua kelas, menghasilkan F1-score sebesar 0.60 untuk kelas 0 dan 0.78 untuk kelas 1. Ini menunjukkan bahwa model tidak hanya mampu mengenali jawaban benar (kelas 1) dengan baik, tetapi juga memiliki performa yang relatif stabil dalam mengklasifikasikan jawaban salah (kelas 0).

Beberapa konfigurasi yang menggunakan learning rate lebih kecil (0.0001) menunjukkan kecenderungan overfitting terhadap kelas positif. Hal ini tercermin dari recall yang sangat tinggi (1.00) pada kelas 1, namun precision pada kelas 0 menurun drastis menjadi 0.00 atau sangat kecil, mengindikasikan ketidakseimbangan dalam prediksi.

Secara keseluruhan, hasil pengujian menunjukkan bahwa peningkatan jumlah unit memori cenderung berdampak positif terhadap performa, namun peningkatan

jumlah layer secara berlebihan tidak memberikan peningkatan signifikan dan justru menambah kompleksitas model. Nilai learning rate 0.001 memberikan hasil yang paling stabil dibanding nilai lainnya. Maka dari itu, konfigurasi optimal yang disarankan untuk model MaLSTM pada penelitian ini adalah menggunakan 128 unit memori, 1 atau 2 layer, dan learning rate 0.001.

2. LSTM

Tabel IV.16 Kode Program Parameter LSTM

	Kode program

```
param_combinations = [
    (32, 1, 0.001),
    (64, 1, 0.001),
    (128, 1, 0.001),
    (64, 2, 0.001),
    (64, 1, 0.0001),
    (128, 2, 0.001),
    (64, 2, 0.0001),
    (32, 2, 0.001),
    (128, 1, 0.0005),
    (64, 3, 0.001),
]
results = []
for units, layers, lr in param_combinations:
    print(f"\n🔥 Testing LSTM with units={units},
layers={layers}, lr={lr}")
    model = build_lstm_classifier(units=units,
layers=layers, learning_rate=lr)
    model.fit([X1_train, X2_train], y_train, batch_size=8,
epochs=5, verbose=0)

    pred = model.predict([X1_test, X2_test])
    pred_binary = (pred[:, 1] >= 0.5).astype(int)

    report = classification_report(np.argmax(y_test,
axis=1), pred_binary, output_dict=False)
    print(report)
```

Pada tabel IV.18 terdapat kode program yang berfungsi Untuk menguji pengaruh parameter pada model LSTM classifier, dilakukan pengulangan proses training dan evaluasi menggunakan berbagai kombinasi parameter. Proses ini dilakukan dalam

satu loop menggunakan list param_combinations yang berisi kombinasi dari tiga parameter penting, yaitu jumlah unit memori (units), jumlah lapisan LSTM (layers), dan learning rate (lr). Nilai-nilai yang digunakan dalam kombinasi tersebut mencakup skenario dari model yang ringan hingga model yang kompleks, termasuk pengaturan learning rate yang lebih rendah untuk melihat efeknya terhadap stabilitas pembelajaran.

Pada setiap iterasi dalam loop, model LSTM dibangun ulang menggunakan fungsi build_lstm_classifier() dengan parameter yang sesuai. Setelah model dibentuk, data latih digunakan untuk melatih model selama 5 epoch secara silent (verbose=0) agar proses training tidak terlalu memakan waktu. Selanjutnya, prediksi dilakukan terhadap data uji (X1_test, X2_test) yang telah dipersiapkan sebelumnya. Hasil prediksi berupa probabilitas, sehingga perlu dikonversi menjadi prediksi biner dengan cara membandingkan probabilitas kelas positif terhadap threshold 0.5.

Akhirnya, evaluasi dilakukan menggunakan fungsi classification_report() dari library sklearn untuk mengukur performa model berdasarkan metrik seperti precision, recall, dan f1-score. Hasil dari setiap kombinasi dicetak dalam bentuk laporan klasifikasi. Dengan pendekatan ini, setiap konfigurasi model dapat dibandingkan secara langsung, sehingga dapat diidentifikasi kombinasi parameter yang paling optimal untuk digunakan dalam sistem penilaian otomatis berbasis LSTM.

Tabel IV.17 Hasil Uji Parameter

Units	Layers	Learning Rate	Accuracy	Precision	Recall	F1-Score
32	1	0.001	0.68	0.70	0.66	0.66
64	1	0.001	0.70	0.75	0.67	0.66
128	1	0.001	0.67	0.73	0.64	0.61
64	2	0.001	0.68	0.72	0.66	0.65
64	1	0.0001	0.62	0.79	0.57	0.50

Units	Layers	Learning Rate	Accuracy	Precision	Recall	F1-Score
128	2	0.001	0.70	0.75	0.67	0.66
64	2	0.0001	0.62	0.79	0.57	0.50
32	2	0.001	0.65	0.69	0.62	0.60
128	1	0.0005	0.63	0.69	0.60	0.56
64	3	0.001	0.67	0.67	0.65	0.65

Pada tabel IV.19 pengujian dilakukan untuk mengevaluasi pengaruh kombinasi parameter jumlah unit memori, jumlah layer, dan learning rate terhadap performa model LSTM Classifier. Hasil menunjukkan bahwa kombinasi 64 unit, 1 layer, dan learning rate 0.001 memberikan akurasi tertinggi yaitu 70%, dengan F1-score sebesar 0.66.

Menariknya, model dengan konfigurasi 128 unit dan 2 layer juga mampu mencapai akurasi serupa, namun dengan waktu pelatihan yang lebih tinggi karena kompleksitas arsitektur bertambah. Di sisi lain, learning rate yang terlalu kecil seperti 0.0001 justru menyebabkan penurunan akurasi, meskipun precision tampak tinggi—karena model terlalu yakin pada satu kelas namun gagal mengenali variasi data dengan baik (recall-nya rendah).

Secara keseluruhan, konfigurasi yang seimbang antara ukuran unit, jumlah layer, dan learning rate terbukti penting untuk mencapai hasil klasifikasi yang optimal. Kombinasi dengan learning rate 0.001 tampaknya lebih stabil dan andal dalam konteks eksperimen ini.

4.2.5 Uji Similarity MaLSTM

Tabel IV.18 Kode Program Uji Similarity MaLSTM

	`import matplotlib.pyplot as plt`
	
	`pairs, labels = generate_pairs(df)`

```
X1 = pad_sequences(tokenizer.texts_to_sequences([x[0] for x in
pairs]), maxlen=max_len)
X2 = pad_sequences(tokenizer.texts_to_sequences([x[1] for x in
pairs]), maxlen=max_len)
y = np.array(labels)
from sklearn.model_selection import train_test_split
X1_train, X1_test, X2_train, X2_test, y_train, y_test =
train_test_split(
    X1, X2, y, test_size=0.2, random_state=42)

model = build_malstm_model()

model.fit([X1_train, X2_train], y_train, batch_size=8,
epochs=15, verbose=1)
y_pred_prob = model.predict([X1_test, X2_test])

plt.figure(figsize=(10,6))
plt.hist(y_pred_prob[y_test == 1], bins=20, alpha=0.7,
label='Mirip (label=1)', color='skyblue')
plt.hist(y_pred_prob[y_test == 0], bins=20, alpha=0.7,
label='Tidak Mirip (label=0)', color='salmon')
plt.xlabel('Similarity Score')
plt.ylabel('Jumlah Pasangan')
plt.title('Distribusi Skor Similarity MaLSTM')
plt.legend()
plt.grid(True)
plt.show()
```

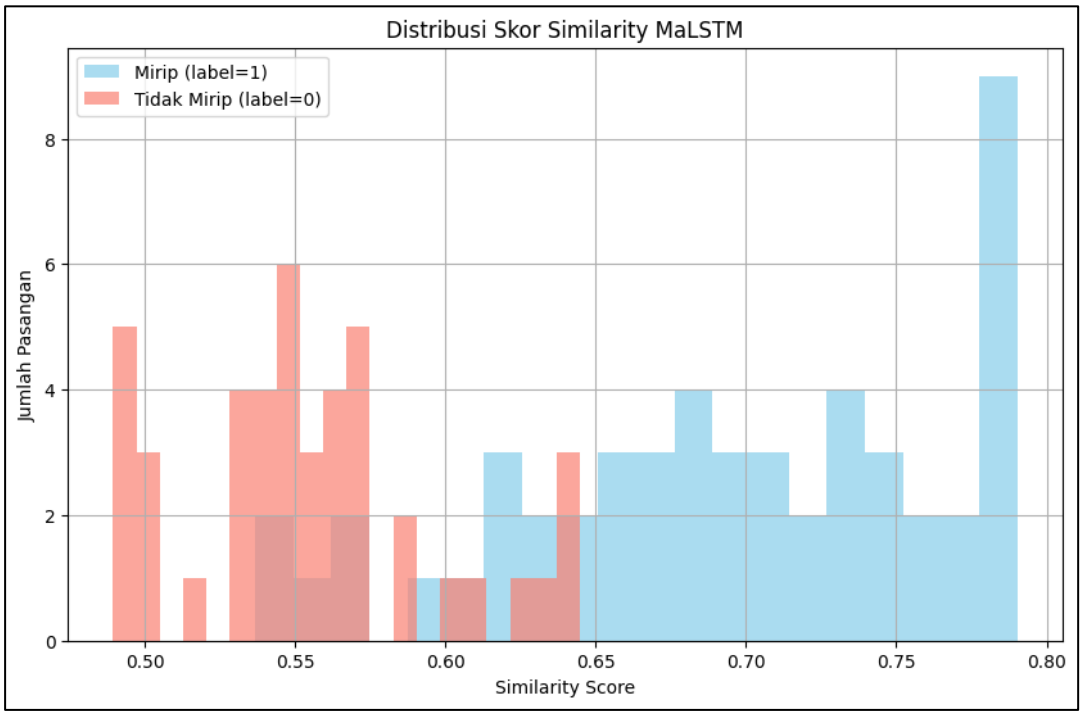
Pada Tabel IV.18 tahap pengujian model MaLSTM, data yang digunakan terdiri dari pasangan kalimat yang dibentuk dari gabungan antara jawaban guru (sebagai kunci) dan jawaban siswa. Masing-masing pasangan tersebut diberi label biner (0 atau 1), yang menunjukkan apakah pasangan tersebut dianggap mirip (label 1) atau tidak mirip (label 0) berdasarkan anotasi dari guru.

Sebelum model dilatih, seluruh data jawaban terlebih dahulu melalui proses tokenisasi, yaitu diubah menjadi urutan angka berdasarkan indeks kata pada kamus (vocab). Hasil tokenisasi ini kemudian dilakukan padding menggunakan pad_sequences agar seluruh input memiliki panjang yang seragam (sesuai dengan max_len). Hasil akhir dari tahap ini adalah dua buah array: X1 yang menyimpan

representasi jawaban guru, dan X2 yang menyimpan representasi jawaban siswa. Label dari pasangan jawaban disimpan dalam array y.

Selanjutnya, data tersebut dibagi menjadi dua bagian menggunakan fungsi train_test_split, dengan rasio 80% untuk pelatihan (X1_train, X2_train, y_train) dan 20% untuk pengujian (X1_test, X2_test, y_test). Model MaLSTM kemudian dibangun dan dilatih dengan input pasangan kalimat ([X1_train, X2_train]) serta labelnya (y_train) selama 15 epoch.

Gambar IV.19 Histogram MaLSTM



Pada Gambar IV.16 Distribusi skor similarity yang dihasilkan oleh model MaLSTM ditampilkan dalam bentuk histogram untuk dua kategori, yaitu pasangan kalimat yang mirip (label 1) dan tidak mirip (label 0). Terlihat bahwa distribusi skor similarity untuk label 1 cenderung berada pada rentang nilai yang lebih tinggi, terutama di atas 0.65 hingga 0.80. Sementara itu, label 0 lebih banyak tersebar di rentang nilai yang lebih rendah, yakni sekitar 0.48 hingga 0.60. Hal ini menunjukkan bahwa model mampu memberikan perbedaan skor similarity yang cukup jelas antara pasangan kalimat yang dianggap mirip dan tidak mirip berdasarkan anotasi guru.

Semakin besar jarak antara dua distribusi ini, maka semakin baik kemampuan model dalam membedakan tingkat kemiripan antar jawaban siswa dan kunci jawaban.

Hasil prediksi berdasarkan label aslinya (y_test), lalu menggambarkan distribusi skor similarity dari pasangan yang mirip (label=1) dan tidak mirip (label=0). Tujuan utama dari visualisasi ini adalah untuk melihat sejauh mana model mampu memisahkan dua kelas tersebut secara eksplisit. Jika histogram memperlihatkan bahwa skor similarity untuk label 1 terpusat pada nilai yang tinggi, dan skor untuk label 0 berada di nilai yang rendah, maka dapat disimpulkan bahwa model memiliki performa yang baik dalam mengenali kemiripan antar kalimat.

Visualisasi ini menjadi dasar evaluasi kualitas skor similarity MaLSTM dan sangat berguna untuk menganalisis apakah model cukup sensitif terhadap perbedaan makna antar jawaban siswa.

4.2.6 Uji Threshold

Tabel IV.19 Kode Program Uji Thersholds

	Kode program

```
import matplotlib.pyplot as plt

pairs, labels = generate_pairs(df)
X1 = pad_sequences(tokenizer.texts_to_sequences([x[0] for x in
pairs]), maxlen=max_len)
X2 = pad_sequences(tokenizer.texts_to_sequences([x[1] for x in
pairs]), maxlen=max_len)
y = np.array(labels)
from sklearn.model_selection import train_test_split
X1_train, X1_test, X2_train, X2_test, y_train, y_test =
train_test_split(
    X1, X2, y, test_size=0.2, random_state=42)

model = build_malstm_model()

model.fit([X1_train, X2_train], y_train, batch_size=8,
epochs=15, verbose=1)
y_pred_prob = model.predict([X1_test, X2_test])
thresholds = [0.5, 0.6, 0.7]

for threshold in thresholds:
    print(f"=== Classification Report (threshold =
{threshold}) ===")
    y_pred_class = (y_pred_prob >= threshold).astype(int)
    print(classification_report(y_test, y_pred_class))
```

```
    print("="*60)
```

Pada Tabel IV.19 Kode di atas digunakan untuk melatih dan mengevaluasi model MaLSTM dalam mengukur kemiripan antara jawaban siswa dan jawaban kunci. Pertama, data diproses dengan membentuk pasangan kalimat menggunakan fungsi `generate_pairs`, yang menghasilkan dua list: `pairs` (berisi pasangan kalimat) dan `labels` (berisi label 1 untuk mirip dan 0 untuk tidak mirip, berdasarkan anotasi guru). Setelah itu, setiap kalimat pada pasangan tersebut ditokenisasi dan diubah menjadi urutan angka menggunakan tokenizer, lalu dilakukan padding agar semua input memiliki panjang yang sama (`max_len`). Data tersebut kemudian dipecah menjadi data latih dan data uji dengan rasio 80:20.

Setelah data siap, model MaLSTM dibangun menggunakan fungsi `build_malstm_model()` dan dilatih dengan data latih selama 15 epoch dan batch size 8. Setelah pelatihan selesai, model digunakan untuk memprediksi skor kemiripan (`y_pred_prob`) pada data uji, dengan nilai keluaran berupa probabilitas antara 0 hingga 1, yang merepresentasikan tingkat kemiripan dua kalimat.

Langkah selanjutnya adalah melakukan evaluasi terhadap skor prediksi tersebut dengan menerapkan teknik thresholding. Nilai ambang batas (threshold) yang digunakan adalah 0.5, 0.6, dan 0.7. Untuk masing-masing threshold, prediksi probabilitas dikonversi menjadi label kelas biner (0 atau 1) berdasarkan apakah nilainya lebih besar atau sama dengan threshold tersebut. Evaluasi dilakukan menggunakan `classification_report` dari scikit-learn, yang menampilkan metrik precision, recall, f1-score, dan akurasi untuk masing-masing kelas. Proses ini bertujuan untuk melihat seberapa besar pengaruh perubahan threshold terhadap performa klasifikasi model MaLSTM.

Tabel IV.20 Hasil Uji Thershold

Threshold	Kelas	Accuracy	Precision	Recall	F1-Score	Macro Avg (F1)	Weighted Avg (F1)
0.5	0	0.60	1.00	0.14	0.24	0.49	0.51
	1	0.60	0.58	1.00	0.73		
0.6	0	0.88	0.82	0.93	0.87	0.87	0.88
	1	0.88	0.93	0.83	0.88		

Threshold	Kelas	Accuracy	Precision	Recall	F1-Score	Macro Avg (F1)	Weighted Avg (F1)
0.7	0	0.67	0.58	1.00	0.73	0.64	0.64
	1	0.67	1.00	0.38	0.56		

Pada Tabel IV.19 terdapat tahapan uji Setelah model MaLSTM dilatih, dilakukan pengujian terhadap hasil prediksi skor similarity dengan menggunakan pendekatan thresholding. Thresholding ini bertujuan untuk mengubah skor probabilitas (0–1) menjadi label klasifikasi biner, yaitu 1 (mirip) atau 0 (tidak mirip), berdasarkan ambang batas tertentu. Pengujian dilakukan terhadap tiga nilai threshold berbeda, yaitu 0.5, 0.6, dan 0.7. Hasil evaluasi dari masing-masing threshold ditampilkan dalam bentuk tabel yang mencakup metrik precision, recall, dan f1-score untuk masing-masing kelas, serta nilai akurasi total, macro average, dan weighted average.

Berdasarkan hasil tersebut, threshold 0.6 menghasilkan performa terbaik dengan akurasi sebesar 88%, f1-score kelas 0 dan 1 yang cukup seimbang (0.87 dan 0.88), serta nilai macro dan weighted average f1-score sebesar 0.87 dan 0.88. Sementara itu, threshold 0.5 menunjukkan kecenderungan model untuk lebih banyak mengklasifikasikan data sebagai kelas mirip (1), ditandai dengan recall tinggi pada kelas 1 namun sangat rendah pada kelas 0. Sebaliknya, threshold 0.7 justru lebih mendominasi kelas 0 dan menurunkan kinerja pada kelas 1. Dengan demikian, threshold 0.6 dapat dianggap sebagai nilai ambang paling optimal dalam mengonversi skor similarity menjadi klasifikasi biner.

4.2.7 Uji Fungsionalitas

Pengujian implementasi sistem menggunakan blackbox ditunjukkan pada Tabel IV.21 uji fungsionalitas

Tabel IV.21 Uji Fungsionalitas

No	Test Case ID	Deskripsi Pengujian	Input	Keluaran yang Diharapkan	Hasil Uji
1	TC-01	Pengujian input lengkap dan valid	Soal, kunci jawaban guru, dan jawaban siswa valid	Sistem menampilkan hasil klasifikasi (Benar/Salah), confidence score, dan similarity score (MaLSTM)	Berhasil

No	Test Case ID	Deskripsi Pengujian	Input	Keluaran yang Diharapkan	Hasil Uji
2	TC-02	Pengujian input kosong pada salah satu kolom	Kosong pada kolom jawaban siswa	Sistem menolak input dan menampilkan pesan kesalahan atau tidak memproses jawaban	Muncul pesan
3	TC-03	Pengujian input teks sangat panjang	Jawaban siswa dengan panjang lebih dari 1000 karakter	Sistem tetap memproses jawaban atau menampilkan peringatan apabila melebihi panjang maksimum input	Diproses
4	TC-04	Pengujian input berupa angka atau simbol	Jawaban siswa berisi angka atau simbol acak	Sistem tetap memproses jika tokenizer dapat mengenali input, atau menolak jika input dianggap tidak valid	Sesuai
5	TC-05	Pengisian hanya pada kolom jawaban siswa	Kolom soal dan kunci jawaban kosong	Sistem menolak input atau tidak menampilkan hasil prediksi	Sesuai
6	TC-06	Pengujian eksekusi dua model secara simultan	Input lengkap (soal, kunci jawaban, jawaban siswa)	Sistem menjalankan proses klasifikasi dengan dua model: LSTM Classifier dan MaLSTM	Berhasil
7	TC-07	Pengujian tampilan hasil ke pengguna	Klik tombol “Cek Jawaban”	Sistem menampilkan hasil klasifikasi, confidence score, dan similarity score ke layar pengguna	Muncul hasil
8	TC-08	Pengujian kemudahan antarmuka pengguna (UI)	Tampilan web saat digunakan oleh pengguna	Pengguna dapat dengan mudah memahami letak input, tombol aksi, dan hasil prediksi yang ditampilkan	Sesuai

4.3. Pembahasan

4.3.1 Interpretasi Hasill Pengujian Model

Hasil pengujian terhadap model LSTM Classifier dan MaLSTM menunjukkan bahwa kedua model memiliki performa yang cukup baik dalam menilai jawaban esai siswa. LSTM Classifier menunjukkan akurasi yang konsisten di atas 70% dalam berbagai rasio pembagian data (80:20, 70:30, dan 60:40). Ini mengindikasikan bahwa model dapat bekerja stabil meskipun jumlah data pelatihan dikurangi. Selain

itu, eksperimen terhadap parameter seperti jumlah unit memori, jumlah layer, dan learning rate berhasil menunjukkan pengaruhnya terhadap performa model. Sementara itu, MaLSTM juga menunjukkan hasil positif terutama saat preprocessing diterapkan, dengan distribusi similarity score yang cenderung tinggi dan mampu membedakan kemiripan jawaban dengan cukup baik.

4.3.2 Hubungan Hasil Dengan Tujuan Penelitian

Tujuan utama dari penelitian ini adalah mengembangkan sistem automated essay scoring yang mampu mengklasifikasikan jawaban siswa sebagai benar atau salah, sekaligus mengukur kemiripan jawaban secara semantik. Berdasarkan hasil pengujian, sistem yang dikembangkan telah mampu memenuhi tujuan tersebut. Model LSTM Classifier memberikan output klasifikasi lengkap dengan probabilitas confidence, sedangkan MaLSTM mampu mengidentifikasi kemiripan semantik antar teks dengan cukup akurat. Sistem ini juga telah terintegrasi ke dalam aplikasi berbasis web untuk digunakan secara lokal oleh guru atau pihak sekolah.

4.3.3 Kelebihan Sistem

Salah satu kelebihan dari sistem ini adalah penggunaan dua pendekatan model sekaligus yang saling melengkapi, yaitu klasifikasi dan semantik. Hasil prediksi disertai dengan confidence score menjadikan sistem ini lebih transparan dan informatif. Selain itu, sistem ini fleksibel karena dapat diuji dengan berbagai variasi parameter dan skenario data. Implementasinya pun cukup ringan dan dapat dijalankan secara lokal.

4.3.4 Keterbatasan Sistem

Keterbatasan utama sistem terletak pada ketergantungan terhadap proses preprocessing, terutama pada MaLSTM. Tanpa preprocessing yang memadai, performa model mengalami penurunan. Selain itu, karena dataset yang digunakan masih terbatas pada domain IPS dan hanya berasal dari satu sumber, model belum dapat dikatakan general jika digunakan di konteks pelajaran atau sekolah lain. Oleh karena itu, pengujian lanjutan dengan data yang lebih luas sangat diperlukan untuk validasi eksternal.

BAB V
KESIMPULAN DAN SARAN

5.2. Kesimpulan

Berdasarkan hasil analisis, dan implementasi sistem yang telah dilakukan, diperoleh beberapa simpulan utama sebagai berikut:

1. Sistem penilaian otomatis untuk soal esai mata pelajaran Ilmu Pengetahuan Sosial (IPS) berhasil diimplementasikan dengan menggunakan dua pendekatan model, yaitu Long Short-Term Memory (LSTM) dan Manhattan LSTM (MaLSTM). Model LSTM digunakan sebagai classifier untuk memprediksi apakah jawaban siswa benar atau salah, sedangkan MaLSTM digunakan untuk mengukur tingkat kemiripan semantik antara jawaban siswa dan kunci jawaban guru. Sistem ini juga berhasil diterapkan dalam bentuk antarmuka web yang memungkinkan pengguna melakukan penilaian secara interaktif dan real-time.
2. Hasil pengujian performa menunjukkan bahwa model LSTM menghasilkan akurasi yang lebih tinggi dibandingkan MaLSTM dalam mengklasifikasikan jawaban siswa. LSTM mencapai akurasi tertinggi sebesar 81% pada rasio data 60:40, sedangkan MaLSTM menunjukkan peningkatan performa saat parameter model dioptimasi, namun tetap sedikit lebih rendah dibanding LSTM. Hal ini menunjukkan bahwa model LSTM lebih optimal dalam tugas klasifikasi biner berbasis esai, sedangkan MaLSTM lebih cocok digunakan sebagai indikator pendukung dalam mengevaluasi kemiripan semantik.

5.3. Saran

Berdasarkan hasil pengembangan sistem dan eksperimen yang telah dilakukan, berikut merupakan beberapa saran untuk penelitian dan pengembangan sistem lebih lanjut agar penilaian otomatis terhadap soal esai dapat semakin optimal:

1. Identifikasi masalah yang lebih mendalam:

Disarankan untuk melakukan proses identifikasi permasalahan secara lebih menyeluruh, terutama dalam memahami konteks makna jawaban esai siswa. Hal ini

penting agar model dapat menangkap nuansa semantik dari jawaban yang secara makna benar, meskipun memiliki perbedaan dalam struktur kalimat.

2. Penyesuaian dengan perkembangan teknologi terbaru:

Pengembangan sistem dapat ditingkatkan dengan memanfaatkan model berbasis transformer atau pre-trained language model Bahasa Indonesia seperti IndoBERT atau IndoGPT. Teknologi ini dapat meningkatkan performa model dalam memahami bahasa alami dengan kompleksitas yang lebih tinggi.

3. Evaluasi terhadap kelemahan sistem:

Sistem perlu dievaluasi lebih lanjut terutama dalam menangani variasi gaya bahasa siswa yang tidak selalu sesuai dengan struktur formal. Dengan menambahkan data pelatihan yang lebih beragam dan proses pelabelan yang lebih akurat, sistem diharapkan mampu meningkatkan kemampuan generalisasi dalam menilai jawaban esai.

4. Pengembangan berkelanjutan secara adaptif dan kolaboratif:

Untuk hasil yang lebih maksimal, pengembangan sistem sebaiknya dilakukan secara bertahap dan terbuka terhadap pendekatan-pendekatan baru. Kolaborasi dengan ahli bahasa, guru, atau pakar pendidikan juga dapat menjadi strategi penting dalam penyempurnaan sistem ke depan.

DAFTAR PUSTAKA

Cahyadi, R., Damayanti, A., Aryadani, D., Rekayasa Multimedia Poltek Negeri Media Kreatif Jakarta Jl Srengseng Sawah, T., Selatan, J., Informatika STMIK AKAKOM Jl Raya Janti, T., & Yogyakarta, K. (2020). RECURRENT NEURAL NETWORK (RNN) DENGAN LONG SHORT TERM MEMORY (LSTM) UNTUK ANALISIS SENTIMEN DATA INSTAGRAM. In *Jurnal Informatika dan Komputer* (Vol. 5, Issue 1).

Esther Irawati Setiawan, & Ika Lestari. (2022). *Stance Classification Pada Berita Berbahasa Indonesia Berbasis Bidirectional LSTM*.

Handayani, R., Destania, Y., & Muhammadiyah Bengkulu, U. (2021). *Indiktika : Jurnal Inovasi Pendidikan Matematika SOAL ESSAY MATERI ARITMATIKA SOSIAL UNTUK KEMAMPUAN BERPIKIR KRITIS MATEMATIS SISWA KELAS VII*. *4*(1).

Iqbal, M., Hasmawati, & Romadhony, A. (2023). Implementation of Recurrent Neural Network (RNN) for Question Similarity Identification in Indonesian Language. *Jurnal Online Informatika*, *8*(2), 213–221. https://doi.org/10.15575/join.v8i2.1138

Ivanedra, K., & Mustikasari, M. (2019). *IMPLEMENTASI METODE RECURRENT NEURAL NETWORK PADA TEXT SUMMARIZATION DENGAN TEKNIK ABSTRAKTIF THE IMPLEMENTATION OF TEXT SUMMARIZATION WITH ABSTRACTIVE TECHNIQUES USING RECURRENT NEURAL NETWORK METHOD*. *6*(4), 377–382. https://doi.org/10.25126/jtiik.201961067

Lana Semba, D. (2024). *IMPLEMENTASI LONG-SHORT TERM MEMORY (LSTM) UNTUK GENERASI FEEDBACK BERBAHASA INDONESIA PADA SISTEM PENILAIAN ESAI*.

Mansoor, M., Ur Rehman, Z., Shaheen, M., Khan, M. A., & Habib, M. (2020). Deep learning based semantic similarity detection using text data. *Information Technology and Control*, *49*(4), 495–510. https://doi.org/10.5755/j01.itc.49.4.27118

Phreeraphattanakarn, T., & Kijsirikul, B. (2021). Text data-augmentation using Text Similarity with Manhattan Siamese long short-term memory for Thai language. *Journal of Physics: Conference Series*, *1780*(1). https://doi.org/10.1088/1742-6596/1780/1/012018

Rachmawati, R. N., & Pusponegoro, N. H. (2021). Spatial Bayes Analysis on Cases of Malnutrition in East Nusa Tenggara, Indonesia. *Procedia Computer Science*, *179*, 337–343. https://doi.org/10.1016/J.PROCS.2021.01.014

Salim, A., Rijal, K., & Hendrik, B. (2023). Studi Literatur Sistem Penilaian Esai Otomatis Pada E-Learning Dengan Algoritma Winnowing. *Jurnal Sistem Informasi Dan Ilmu Komputer*, *1*(3), 163–172. https://doi.org/10.59581/jusiik-widyakarya.v1i3.1227

Siahaan, P. A. (2021). *PENILAIAN OTOMATIS JAWABAN TES URAIAN LISAN MENGGUNAKAN METODE RECURRENT NEURAL NETWORK*.

Siringoringo, R., Perangin-angin, R., Julia Harianja, E. G., Lumbantoruan, G., & Novita Purba, E. (2023). *METHOMIKA: Jurnal Manajemen Informatika & Komputerisasi Akuntansi MODEL BIDIRECTIONAL LSTM UNTUK PEMROSESAN SEKUENSIAL DATA TEKS SPAM*. *7*(2). https://doi.org/10.46880/jmika.Vol7No2.pp265-271

Viji, D., & Revathy, S. (2022). A hybrid approach of Weighted Fine-Tuned BERT extraction with deep Siamese Bi – LSTM model for semantic text similarity identification. *Multimedia Tools and Applications*, *81*(5), 6131–6157. https://doi.org/10.1007/s11042-021-11771-6

Wiranda, L., & Sadikin, M. (2020). *PENERAPAN LONG SHORT TERM MEMORY PADA DATA TIME SERIES UNTUK MEMPREDIKSI PENJUALAN PRODUK PT. METISKA FARMA* (Vol. 8).

Yosia Wibowo, L., Annisa, N., Ananda Khairunnisa Viktor Handrianus Pranatawijaya, P., & Priskila, R. (2024). IMPLEMENTASI LONG SHORT-TERM MEMORY DALAM ANALISIS SENTIMEN PENGGUNA APLIKASI TWITTER YANG MENGANDUNG UJARAN KEBENCIAN. In *Jurnal Mahasiswa Teknik Informatika* (Vol. 8, Issue 3).

LAMPIRAN

Lampiran 1 Kode Program LSTM

```
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.utils.class_weight import compute_class_weight
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, LSTM, Lambda,
Dense, Concatenate
from sklearn.metrics import classification_report, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from gensim.models import FastText
import gensim
import os

import pandas as pd

df = pd.read_csv('datafix.csv', sep=';', quotechar='"',
engine='python')
df.columns = df.columns.str.strip()
df = df.dropna()
print(df.head())
import pandas as pd
import re
import nltk
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from nltk.corpus import stopwords

# Unduh stopwords indo (hanya perlu dijalankan sekali)
nltk.download('stopwords')
stop_words = set(stopwords.words('indonesian'))

# Sastrawi stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()

# Function untuk preprocessing
def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z\s]', '', text)
```

```
    words = text.split()
    tokens_awal = words.copy()
    words = [word for word in words if word not in stop_words]
    return tokens_awal, words

# Function stemming pakai Sastrawi
def apply_stemming(words):
    return [stemmer.stem(word) for word in words]

# Terapkan ke dataframe
df[['token_awal', 'token_stp']] = df['jawaban_siswa'].apply(lambda
x: pd.Series(clean_text(x)))
df['token_stem'] = df['token_stp'].apply(apply_stemming)

# Cek 20 data pertama
for i, row in df[['jawaban_siswa', 'token_awal', 'token_stp',
'token_stem']].head(20).iterrows():
    print(f'📄 Asli         : {row["jawaban_siswa"]}')
    print(f'🔤 Tokenisasi   : {row["token_awal"]}')
    print(f'🧹 Stopword     : {row["token_stp"]}')
    print(f'🌱 Stemming     : {row["token_stem"]}')
    print('-'*80)
from gensim.models import KeyedVectors

# Load hanya 100K kata biar gak over RAM
embedding_model =
KeyedVectors.load_word2vec_format("cc.id.300.vec", binary=False,
limit=100000)
print("Embedding siap dipakai!")
# Contoh ambil vektor kata
print(embedding_model['perdagangan'])
import pickle

# Ambil semua teks dari jawaban siswa (buat bikin vocab-nya)
texts = df['jawaban_siswa'].tolist()

# Tokenizer
tokenizer = Tokenizer()
tokenizer.fit_on_texts(texts)

# Simpan tokenizer (buat inference nanti)
with open('tokenizer.pkl', 'wb') as f:
    pickle.dump(tokenizer, f)

# Setup pad max length
max_len = 100
```

```
# Bikin fungsi bantu biar gampang padding-nya nanti
tokenize_pad = lambda texts:
pad_sequences(tokenizer.texts_to_sequences(texts), maxlen=max_len)

# Setelah tokenizer.fit_on_texts(texts)
word_index = tokenizer.word_index
vocab_size = len(word_index) + 1  # <--- INI dulu, baru lanjut ke
bawah

# Lalu baru bisa bikin matrix embedding
embedding_dim = embedding_model.vector_size
vocab_size = len(tokenizer.word_index) + 1
embedding_matrix = np.zeros((vocab_size, embedding_dim))

for word, i in tokenizer.word_index.items():
    if word in embedding_model:
        embedding_matrix[i] = embedding_model[word]
def build_lstm_classifier():
    input_layer = Input(shape=(max_len,))
    embedding_layer = Embedding(input_dim=vocab_size,
                                output_dim=embedding_dim,
                                weights=[embedding_matrix],
                                trainable=False)(input_layer)

    lstm = LSTM(64)(embedding_layer)
    dense = Dense(64, activation='relu')(lstm)
    output = Dense(1, activation='sigmoid')(dense)  # binary output

    model = Model(inputs=input_layer, outputs=output)
    model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
    return model

from sklearn.model_selection import train_test_split
from sklearn.utils.class_weight import compute_class_weight
from keras.models import Model
from keras.layers import Input, Embedding, LSTM, Dense
from keras.preprocessing.sequence import pad_sequences
import numpy as np

# --- 1. Tokenisasi & Padding ---
tokenize_pad = lambda texts:
pad_sequences(tokenizer.texts_to_sequences(texts), maxlen=max_len)
X = tokenize_pad(df['jawaban_siswa'])
y = df['label'].values

# --- 2. Split Data ---
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# --- 3. Class Weights (Biar Balance) ---
classes = np.array([0, 1])  # ☑ HARUS numpy array
weights = compute_class_weight(class_weight='balanced',
classes=classes, y=y_train)
class_weights = dict(zip(classes, weights))

# --- 5. Train & Save ---
model = build_lstm_classifier()  # ☑ pakai yang udah kamu
definisiin sebelumnya
model.fit(X_train, y_train, batch_size=8, epochs=10,
class_weight=class_weights, verbose=1)
model.save("lstm_classifier.h5")

from sklearn.metrics import classification_report,
confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Prediksi probabilitas
y_pred_prob = model.predict(X_test)

# Threshold ke label
y_pred = (y_pred_prob >= 0.5).astype(int).flatten()

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay(cm).plot(cmap="Blues")
plt.title("Confusion Matrix")
plt.grid(False)
plt.show()

# Classification report
print(classification_report(y_test, y_pred))

# Confidence histogram
plt.hist(y_pred_prob, bins=20, color="orchid", edgecolor="black")
plt.title("Confidence Distribution")
plt.xlabel("Confidence Score")
plt.ylabel("Jumlah Prediksi")
plt.grid(True)
plt.show()
```

Lampiran 2 Kode Program MaLSTM

```
import pickle
# Ambil semua teks dari jawaban siswa (buat bikin vocab-nya)
texts = df['jawaban_siswa'].tolist()
# Tokenizer
tokenizer = Tokenizer()
tokenizer.fit_on_texts(texts)
# Simpan tokenizer (buat inference nanti)
with open('tokenizer.pkl', 'wb') as f:
    pickle.dump(tokenizer, f)
# Setup pad max length
max_len = 100
# Bikin fungsi bantu biar gampang padding-nya nanti
tokenize_pad = lambda texts:
pad_sequences(tokenizer.texts_to_sequences(texts), maxlen=max_len)
# Setelah tokenizer.fit_on_texts(texts)
word_index = tokenizer.word_index
vocab_size = len(word_index) + 1  # <--- INI dulu, baru lanjut ke
bawah
# Lalu baru bisa bikin matrix embedding
embedding_dim = embedding_model.vector_size
vocab_size = len(tokenizer.word_index) + 1
embedding_matrix = np.zeros((vocab_size, embedding_dim))

for word, i in tokenizer.word_index.items():
    if word in embedding_model:
        embedding_matrix[i] = embedding_model[word]

def generate_pairs(df):
    pairs = []
    labels = []

    for soal in df['soal'].unique():
        subset = df[df['soal'] == soal]

        # Ambil satu jawaban yang labelnya 1 sebagai 'kunci jawaban
guru'
        kunci_list = subset[subset['label'] ==
1]['jawaban_siswa'].tolist()
        if not kunci_list:
            continue  # lewati kalau nggak ada jawaban benar

        kunci = kunci_list[0]  # ambil satu kunci aja
```

```
        for _, row in subset.iterrows():
            pairs.append((kunci, row['jawaban_siswa']))
            labels.append(row['label'])

    return pairs, labels
from tqdm import tqdm

def rolling_leave_one_out(df):
    for soal in tqdm(df['soal'].unique()):
        subset = df[df['soal'] == soal].reset_index(drop=True)

        for i in range(len(subset)):
            test = subset.iloc[i]  # 1 jawaban siswa utk test
            train = subset.drop(i)

            # ambil 1 jawaban benar dari train sebagai kunci
            kunci_list = train[train['label'] ==
1]['jawaban_siswa'].tolist()
            if not kunci_list:
                continue  # skip kalau di train ga ada jawaban
benar
            kunci = kunci_list[0]

            # bikin pair (kunci, jawaban siswa) utk training
            train_pairs, train_labels = generate_pairs(train)

            # bikin pair test → (kunci, jawaban_siswa_test)
            test_pairs = [(kunci, test['jawaban_siswa'])]
            test_labels = [test['label']]

            yield train_pairs, train_labels, test_pairs,
test_labels
from tensorflow.keras.layers import Input, Embedding, LSTM, Lambda,
Dense
from tensorflow.keras.models import Model
from tensorflow.keras import backend as K

def exponent_neg_manhattan_distance(vects):
    x, y = vects
    return K.exp(-K.sum(K.abs(x - y), axis=1, keepdims=True))

def build_malstm_model():
    input_1 = Input(shape=(max_len,))
    input_2 = Input(shape=(max_len,))

    embedding_layer = Embedding(input_dim=vocab_size,
                                output_dim=embedding_dim,
```

```
                                weights=[embedding_matrix],
                                trainable=False)

    encoded_1 = embedding_layer(input_1)
    encoded_2 = embedding_layer(input_2)

    shared_lstm = LSTM(50)

    output_1 = shared_lstm(encoded_1)
    output_2 = shared_lstm(encoded_2)

    # Output similarity pakai sigmoid
    malstm_distance = Lambda(exponent_neg_manhattan_distance,
                             output_shape=lambda x: (x[0][0],
1))([output_1, output_2])

    output = Dense(1, activation='sigmoid')(malstm_distance)

    model = Model(inputs=[input_1, input_2], outputs=output)
    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    return model
# 1. Generate pasangan kalimat (kunci, jawaban siswa)
# Label: 1 = mirip, 0 = tidak mirip (berdasarkan anotasi guru)
pairs, labels = generate_pairs(df)

# 2. Tokenisasi dan padding
X1 = pad_sequences(tokenizer.texts_to_sequences([x[0] for x in
pairs]), maxlen=max_len)
X2 = pad_sequences(tokenizer.texts_to_sequences([x[1] for x in
pairs]), maxlen=max_len)
y = np.array(labels)

# 3. Split data menjadi training dan testing (80:20)
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

X1_train, X1_test, X2_train, X2_test, y_train, y_test =
train_test_split(
    X1, X2, y, test_size=0.2, random_state=42
)

# 4. Bangun dan latih model MaLSTM
model = build_malstm_model()
model.fit([X1_train, X2_train], y_train, batch_size=8, epochs=15,
verbose=1)
```

```
# 5. Simpan model
model.save("malstm_model.h5")

# 6. Prediksi skor similarity untuk data uji
y_pred_prob = model.predict([X1_test, X2_test])  # output dalam
rentang 0–1

# === Evaluasi 1: Visualisasi distribusi similarity ===
import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))
plt.hist(y_pred_prob[y_test == 1], bins=20, alpha=0.7, label='Mirip
(label=1)', color='skyblue')
plt.hist(y_pred_prob[y_test == 0], bins=20, alpha=0.7, label='Tidak
Mirip (label=0)', color='salmon')
plt.xlabel('Similarity Score')
plt.ylabel('Jumlah Pasangan')
plt.title('Distribusi Skor Similarity MaLSTM')
plt.legend()
plt.grid(True)
plt.show()

# === Evaluasi 2: Thresholding klasifikasi berdasarkan similarity
score ===
thresholds = [0.5, 0.6, 0.7]

for threshold in thresholds:
    print(f"=== Classification Report (threshold = {threshold})
===")
    y_pred_class = (y_pred_prob >= threshold).astype(int)
    print(classification_report(y_test, y_pred_class))
    print("="*60)
from sklearn.metrics import classification_report,
confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Prediksi skor similarity
pred = model.predict([X1, X2])

# Konversi skor similarity ke kelas biner berdasarkan threshold
threshold = 0.5
pred_binary = (pred >= threshold).astype(int)

# Evaluasi klasifikasi
print("=== Classification Report (Threshold = 0.5) ===")
print(classification_report(y, pred_binary))
```

```
# Confusion Matrix
cm = confusion_matrix(y, pred_binary)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap='Oranges')
plt.title(" Confusion Matrix MaLSTM")
plt.show()

# Distribusi Skor Similarity
plt.figure(figsize=(6, 4))
plt.hist(pred[y == 1], bins=20, alpha=0.7, label="Mirip (label=1)",
color='lightblue')
plt.hist(pred[y == 0], bins=20, alpha=0.7, label="Tidak Mirip
(label=0)", color='lightcoral')
plt.title(" Distribusi Similarity (Manhattan Distance)")
plt.xlabel("Similarity Score")
plt.ylabel("Jumlah Prediksi")
plt.legend()
plt.grid(True)
plt.show()
```

Lampiran 3 Kode Program app.py

```
from flask import Flask, render_template, request
import pickle
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras import backend as K

# === Load Tokenizer
with open('tokenizer.pkl', 'rb') as f:
    tokenizer = pickle.load(f)

max_len = 100

# === Load MaLSTM Model
def exponent_neg_manhattan_distance(vects):
    x, y = vects
    return K.exp(-K.sum(K.abs(x - y), axis=1, keepdims=True))

model_sim = load_model('malstm_model.h5', custom_objects={
    'exponent_neg_manhattan_distance':
exponent_neg_manhattan_distance
})

# === Load LSTM Classifier Model
```

```
model_cls = load_model('lstm_classifier.h5')

# === Predict Similarity Score (MaLSTM)
def predict_ma_lstm(jawaban_guru, jawaban_siswa):
    seq_1 = tokenizer.texts_to_sequences([jawaban_guru])
    seq_2 = tokenizer.texts_to_sequences([jawaban_siswa])

    padded_1 = pad_sequences(seq_1, maxlen=max_len)
    padded_2 = pad_sequences(seq_2, maxlen=max_len)

    pred = model_sim.predict([padded_1, padded_2])
    score = float(pred[0][0])  # output sigmoid
    return round(score * 100, 2)  # persentase

# === Predict Classification (Benar/Salah)
def predict_lstm_classifier(jawaban_siswa):
    seq = tokenizer.texts_to_sequences([jawaban_siswa])
    padded = pad_sequences(seq, maxlen=max_len)

    pred = model_cls.predict(padded)
    confidence = float(pred[0][0])
    prediction = int(round(confidence))  # sigmoid → 0 or 1
    return prediction, round(confidence * 100, 2)

# === Flask App
app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        soal = request.form['soal']
        jawaban_guru = request.form['jawaban_guru']
        jawaban_siswa = request.form['jawaban_siswa']

        similarity_score = predict_ma_lstm(jawaban_guru,
jawaban_siswa)
        cls_pred, cls_conf =
predict_lstm_classifier(jawaban_siswa)  # ☑ hanya jawaban siswa!

        return render_template("frontend.html",
            similarity_score=similarity_score,
            cls_pred=cls_pred,
            cls_conf=cls_conf,
            soal=soal,
            jawaban_guru=jawaban_guru,
            jawaban_siswa=jawaban_siswa
        )
```

```
    # GET method default values
    return render_template("frontend.html",
        similarity_score=None,
        cls_pred=None,
        cls_conf=None,
        soal='',
        jawaban_guru='',
        jawaban_siswa=''
    )

if __name__ == '__main__':
    app.run(debug=True)
```

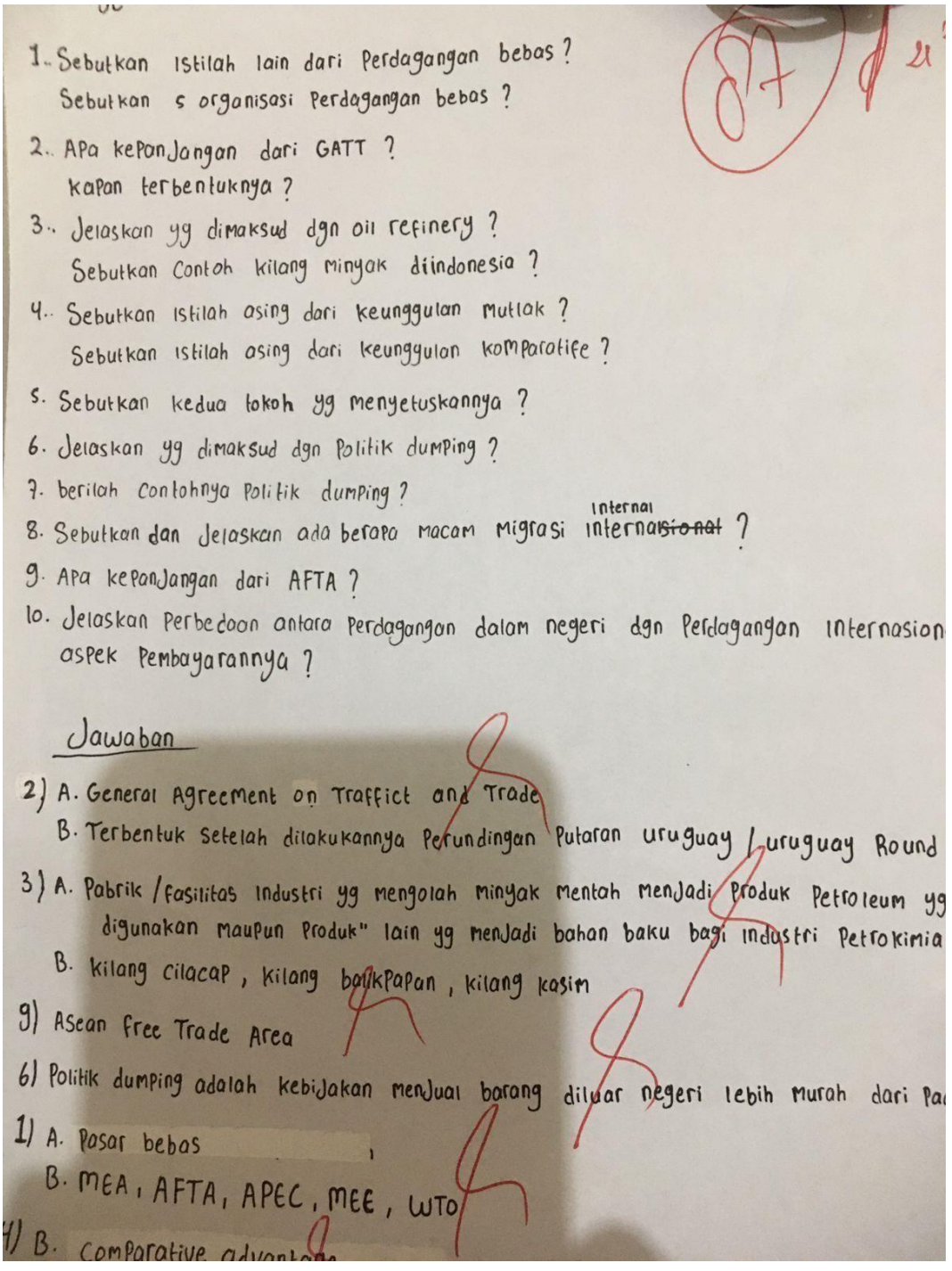
Lampiran 4 Kode Program frontend.html

```
<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Auto Essay Scoring student 💔</title>
  <script src="https://cdn.tailwindcss.com"></script>
  <link
href="https://fonts.googleapis.com/css2?family=Comic+Neue:wght@700&
display=swap" rel="stylesheet">
  <style>
    body {
      font-family: 'Comic Neue', cursive;
      background: linear-gradient(to bottom right, #ffe4ec,
#fff0f6);
      position: relative;
    }
    .bubble {
      position: absolute;
      width: 20px;
      height: 20px;
      background: rgba(255, 192, 203, 0.3);
      border-radius: 50%;
      animation: float 6s infinite ease-in-out;
    }
    @keyframes float {
      0% { transform: translateY(0); }
      50% { transform: translateY(-100px); }
      100% { transform: translateY(0); }
```

```
    }
  </style>
</head>
<body class="text-pink-900">
  <div class="max-w-xl mx-auto mt-10 p-6 bg-white rounded-2xl
shadow-lg relative z-10">
    <h1 class="text-3xl font-bold text-center mb-6">💘 Auto Essay
Scoring</h1>

    <form action="/" method="POST" class="space-y-4">
      <div>
        <label for="soal" class="block font-semibold">▧
Soal</label>
        <textarea id="soal" name="soal" rows="2" class="w-full p-2
border rounded-md" required>{{ soal }}</textarea>
      </div>

      <div>
        <label for="jawaban_guru" class="block font-semibold">👩‍🏫
Kunci Jawaban Guru</label>
        <textarea id="jawaban_guru" name="jawaban_guru" rows="2"
class="w-full p-2 border rounded-md" required>{{ jawaban_guru
}}</textarea>
      </div>

      <div>
        <label for="jawaban_siswa" class="block font-semibold">✏️
Jawaban Siswa</label>
        <textarea id="jawaban_siswa" name="jawaban_siswa" rows="3"
class="w-full p-2 border rounded-md" required>{{ jawaban_siswa
}}</textarea>
      </div>

      <div class="text-center">
        <button type="submit" class="bg-pink-500 hover:bg-pink-600
text-white font-bold py-2 px-4 rounded-xl shadow-md">
          💡 Cek Jawaban
        </button>
      </div>
    </form>

    {% if cls_pred is not none %}
    <div class="mt-6 p-4 bg-pink-50 border border-pink-200 rounded-
xl shadow">
      <h2 class="text-xl font-semibold mb-2">📊 Hasil Prediksi</h2>
      <p><strong>Status Klasifikasi:</strong> {{ '☑ Benar' if
cls_pred == 1 else '❌ Salah' }}</p>
```

```
      <p><strong>Confidence Classifier:</strong> {{ cls_conf
}}%</p>
      <p><strong>Similarity Score (MaLSTM):</strong> {{
similarity_score }}%</p>
    </div>
    {% endif %}
    <div class="mt-6">
      <h3 class="font-bold mb-2">🎯 Contoh Soal:</h3>
      <ul class="space-y-2">
        <li>
          <button type="button" onclick="isiContoh('Apa dampak
perdagangan bebas?', 'Perdagangan bebas dapat meningkatkan ekspor
dan impor.')"
            class="text-sm bg-pink-100 px-3 py-1 rounded-full
hover:bg-pink-200 transition">
            Perdagangan Bebas
          </button>
        </li>
        <li>
          <button type="button" onclick="isiContoh('Apa fungsi
minyak bumi?', 'Minyak bumi digunakan sebagai bahan bakar dan
sumber energi.')"
            class="text-sm bg-pink-100 px-3 py-1 rounded-full
hover:bg-pink-200 transition">
            Minyak Bumi
          </button>
        </li>
      </ul>
    </div>
  </div>

  <!-- Bubble Effect -->
  <div class="bubble" style="top:20%; left:10%; animation-delay:
0s;"></div>
  <div class="bubble" style="top:30%; left:80%; animation-delay:
2s;"></div>
  <div class="bubble" style="top:70%; left:50%; animation-delay:
4s;"></div>

  <script>
    function isiContoh(soal, jawaban) {
      document.getElementById('soal').value = soal;
      document.getElementById('jawaban_guru').value = jawaban;
    }
  </script>
</body>
</html>
```

Lampiran 5 Scan Soal dan Jawaban Siswa



1. Sebutkan istilah lain dari perdagangan bebas ?
   Sebutkan 5 organisasi perdagangan bebas ?
2. Apa kepanjangan dari GATT ?
   kapan terbentuknya ?
3. Jelaskan yg dimaksud dgn oil refinery ?
   Sebutkan contoh kilang minyak diindonesia ?
4. Sebutkan istilah asing dari keunggulan mutlak ?
   Sebutkan istilah asing dari keunggulan komparatife ?
5. Sebutkan kedua tokoh yg menyetuskannya ?
6. Jelaskan yg dimaksud dgn Politik dumping ?
7. berilah contohnya Politik dumping ?
8. Sebutkan dan Jelaskan ada berapa macam migrasi internasional ~~sional~~ ?
9. Apa kepanjangan dari AFTA ?
10. Jelaskan perbedaan antara perdagangan dalam negeri dgn perdagangan internasion aspek pembayarannya ?

Jawaban

2) A. General Agreement on Traffict and Trade
   B. Terbentuk setelah dilakukannya perundingan putaran uruguay / uruguay Round
3) A. Pabrik / fasilitas industri yg mengolah minyak mentah menjadi produk petroleum yg digunakan maupun produk" lain yg menjadi bahan baku bagi industri petrokimia
   B. kilang cilacap, kilang balikpapan, kilang kasim
9) Asean free Trade Area
6) Politik dumping adalah kebijakan menjual barang diluar negeri lebih murah dari pa
1) A. pasar bebas
   B. MEA, AFTA, APEC, MEE, WTO
4) B. comparative advanta