SISTEM *CHATBOT* SEBAGAI LAYANAN TUGAS AKHIR PROGRAM STUDI TEKNIK INFORMATIKA UNISSULA MENGGUNAKAN METODE *Bidirectional Encoder Representations from Transformers* (BERT)

LAPORAN TUGAS AKHIR

Laporan ini Disusun untuk Memenuhi Salah Satu Syarat Memperoleh
Gelar Sarjana Strata 1 (S1) pada Program Studi Teknik Informatika
Fakultas Teknologi Industri Universitas Islam Sultan Agung Semarang

DISUSUN OLEH :

MOHAMMAD LUQI WIHARTO

NIM 32602000041

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM SULTAN AGUNG

SEMARANG

2024

FINAL PROJECT

CHATBOT SYSTEM AS A SERVICE FOR THE FINAL PROJECT OF THE UNISSULA INFORMATICS ENGINEERING STUDY PROGRAM USING THE METHOD OF *Bidirectional Encoder Representations from Transformers* (BERT)

Proposed to complete the requirement to obtain a bachelor's degree (S1)
at Informatics Engineering Departement of Industrial Technology Faculty
Sultan Agung Islamic University

Arranged By :

MOHAMMAD LUQI WIHARTO

NIM 32602000041

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM SULTAN AGUNG

SEMARANG

2024

LEMBAR PENGESAHAN PEMBIMBING

Laporan Tugas Akhir dengan judul "SISTEM CHATBOT SEBAGAI LAYANAN TUGAS AKHIR PROGRAM STUDI TEKNIK INFORMATIKA UNISSULA MENGGUNAKAN METODE Bidirectional Encoder Representations from Transformers (BERT)" ini disusun oleh :

Nama : Mohammad Luqi Wiharto

NIM : 32602000041

Program Studi : Teknik Informatika

Telah disahkan oleh dosen pembimbing pada :

Hari : Selasa

Tanggal : 27 Februari 2024

Mengesahkan,

Pembimbing I	Pembimbing II
Badie'ah, S.T., M.Kom	Imam Much Ibnu Subroto, S.T., M.Sc., Ph.D
NIDN. 0619018701	NIDN. 0613037301

Mengetahui,

Ketua Program Studi Teknik Informatika
Fakultas Teknologi Industri
Universitas Islam Sultan Agung

Ir. Sri Mulyono, M.Eng
NIDN. 0626066601

LEMBAR PENGESAHAN PENGUJI

Laporan tugas akhir dengan judul "SISTEM CHATBOT SEBAGAI LAYANAN TUGAS AKHIR PROGRAM STUDI TEKNIK INFORMATIKA UNISSULA MENGGUNAKAN METODE Bidirectional Encoder Representations from Transformers (BERT)" ini telah dipertahankan di depan dosen penguji Tugas Akhir pada :

Hari : Selasa

Tanggal : 27 Februari 2024

TIM PENGUJI

Anggota I	Anggota II
Bagus Satrio WP, S.Kom.,M.Cs	Ghufron, ST., M.Kom
NIDN.0626066601	NIDN.0602079005

Ketua Penguji

Bagus Satrio WP, S.Kom.,M.Cs

NIDN. 062606660

SURAT PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan dibawah ini :

Nama : Mohammad Luqi Wiharto

NIM : 32602000041

Judul Tugas Akhir : SISTEM CHATBOT SEBAGAI LAYANAN TUGAS AKHIR PROGRAM STUDI TEKNIK INFORMATIKA UNISSULA MENGGUNAKAN METODE Bidirectional Encoder Representations from Transformers (BERT)

Dengan bahwa ini saya menyatakan bahwa judul dan isi Tugas Akhir yang saya buat dalam rangka menyelesaikan Pendidikan Strata Satu (S1) Teknik Informatika tersebut adalah asli dan belum pernah diangkat, ditulis ataupun dipublikasikan oleh siapapun baik keseluruhan maupun sebagian, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka, dan apbila di kemudian hari ternyata terbukti bahwa judul Tugas Akhir tersebut pernah diangkat, ditulis ataupun dipublikasikan, maka saya bersedia dikenakan sanksi akademis. Demikian surat pernyataan ini saya buat dengan sadar dan penuh tanggung jawab.

Semarang, 29 Februari 2024

Yang Menyatakan,

Mohammad Luqi Wiharto

KATA PENGANTAR

Dengan mengucap syukur alhamdulillah atas kehadirat Allah SWT yang telah memberikan rahmat dan karunianya kepada penulis, sehingga dapat menyelesaikan Tugas Akhir dengan judul "Sistem *Chatbot* Sebagai Layanan Tugas Akhir Program Studi Teknik Informatika UNISSULA Menggunakan Metode *Bidirectional Encoder Representations from Transformers* (BERT)" ini untuk memenuhi salah satu syarat menyelesaikan studi serta dalam rangka memperoleh gelar sarjana (S-1) pada Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Islam Sultan Agung Semarang.

Tugas Akhir ini disusun dan dibuat dengan adanya bantuan dari berbagai pihak, materi maupun teknis, oleh karena itu saya selaku penulis mengucapkan terima kasih kepada:

1. Rektor UNISSULA Bapak Prof. Dr. H. Gunarto, S.H., M.H yang mengizinkan penulis menimba ilmu di kampus ini.
2. Dekan Fakultas Teknologi Industri Ibu Dr. Novi Marlyana, S.T., M.T.
3. Dosen pembimbing I penulis Badie'ah, S.T., M.Kom yang telah meluangkan waktu dan memberi ilmu.
4. Dosen pembimbing II penulis Imam Much Ibnu Subroto, S.T., M.Sc., Ph.D yang memberikan banyak nasehat dan saran.
5. Orang tua penulis yang telah mengizinkan untuk menyelesaikan laporan ini,
6. Dan kepada semua pihak yang tidak dapat saya sebutkan satu persatu.

Dengan rendah hati, penulis menyadari bahwa laporan masih memiliki banyak kekurangan dalam hal kuantitas, kualitas, dan ilmu pengetahuan. Oleh karena itu, penulis mengharapkan kritikan dan saran yang membangun untuk membantu laporan ini menjadi lebih baik di masa depan..

Semarang, 25 Desember 2023

Mohammad Luqi Wiharto

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ABSTRAK

Di era digital saat ini, akses informasi yang cepat dan efisien sangatlah penting. Pelayanan berbasis Teknologi Informasi akan membuat pekerjaan lebih mudah dilakukan. Salah satu jenis inovasi teknologi layanan informasi adalah chatbot. Chatbot adalah salah satu contoh aplikasi yang menggunakan kecerdasan buatan (AI). Penggunaan *chatbot* di Indonesia masih tergolong rendah. Jumlah UMKM di Indonesia yang menggunakan chatbot hanya 15%, menurut survei Pancake. Mahasiswa yang mengambil mata kuliah tugas akhir di UNISSULA khususnya Program Studi Teknik Informatika harus mengikuti banyak prosedur administratif. Buku panduan tugas akhir untuk teknik informatika pada dasarnya cukup lengkap Namun, mahasiswa membutuhkan waktu yang lama untuk memilih apa yang mereka butuhkan. Penulis menggunakan model BERT versi Indonesia yaitu IndoBERT. Dengan dikembangkannya sistem ini diharapkan akan tercipta sistem *chatbot* Sebagai layanan tugas akhir yang akan memudahkan pengguna khususnya mahasiswa Teknik Informatika UNISSULA untuk mendapatkan informasi terkait Tugas Akhir yang mampu berkomunikasi dengan user dan dapat menghasilkan performa yang baik dari hasil sistem *chatbot* menggunakan metode IndoBERT dengan sampel data dari dataset buku panduan Tugas Akhir Program Studi teknik Informatika UNISSULA.
Keyword: Sistem *chatbot*, Teknologi informasi, Layanan tugas akhir, BERT, IndoBERT

ABSTRACT

In today's digital era, fast and efficient access to information is essential. Information Technology-based services will make work easier to do. One type of information service technology innovation is chatbot. Chatbot is one example of an application that uses artificial intelligence (AI). The use of chatbots in Indonesia is still relatively low. The number of MSMEs in Indonesia that use chatbots is only 15%, according to a Pancake survey. Students who take final project courses at UNISSULA, especially the Informatics Engineering Study Program, must follow many administrative procedures. The final project guidebook for informatics engineering is basically quite complete However, students take a long time to choose what they need. The author uses the Indonesian version of the BERT model, namely IndoBERT. With the development of this system, it is hoped that a chatbot system will be created as a final project service that will make it easier for users, especially UNISSULA Informatics Engineering students, to get information related to the Final Project that is able to communicate with users and can produce good performance from the results of the chatbot system using the IndoBERT method with sample data from the UNISSULA Informatics Engineering Study Program Final Project guidebook dataset.
Keyword: Sistem chatbot, Teknologi informasi, Layanan tugas akhir, BERT, IndoBERT

BAB I
PENDAHULUAN

1.1 Latar Belakang

Di era digital saat ini, akses informasi yang cepat dan efisien sangatlah penting. Orang-orang menelusuri informasi tentang berbagai topik, termasuk berita, produk, layanan, atau tutorial. Namun, menemukan informasi ini melalui situs web atau aplikasi dapat memakan waktu lama.

Pelayanan berbasis teknologi informasi di tempat kerja akan mampu memudahkan pelaksanaan pekerjaan. Aplikasi *Chatbot* merupakan salah satu bentuk kemajuan teknologi layanan informasi (Wahyudi dan Fitria, 2020).

Chatbot adalah program atau layanan yang berinteraksi dengan pengguna melalui teks (Amalia *dkk.*, 2019). Kecerdasan buatan (AI) adalah cabang kecerdasan buatan yang digunakan untuk memecahkan masalah dengan meniru kecerdasan benda mati dan organisme hidup. Khususnya, pemrosesan bahasa alami (NLP) adalah teknik yang digunakan komputer untuk memproses bahasa yang biasa digunakan manusia dalam percakapan sehari-hari. Ini memungkinkan chatbot untuk berkomunikasi, memahami, dan merespons dengan lebih baik daripada manusia (Hikmah *dkk.*, 2023).

Penggunaan *chatbot* di Indonesia masih tergolong rendah. Jumlah UMKM di Indonesia yang menggunakan *chatbot* hanya 15%, menurut survei Pancake (Wintoro *dkk.*, 2022). Ini menunjukkan bahwa banyak bisnis belum menggunakan *chatbot* untuk meningkatkan operasi mereka, termasuk Program Studi Teknik Informatika UNISSULA. Program Studi Teknik Informatika UNISSULA tetap beroperasi secara konvensional, terutama dalam hal informasi tugas akhir. Mahasiswa yang mengambil mata kuliah tugas akhir di UNISSULA harus mengikuti banyak prosedur administratif. Pada dasarnya, buku panduan tugas akhir untuk program studi teknik informatika cukup lengkap sehingga mahasiswa dapat membacanya untuk mendapatkan berbagai informasi. Namun, informasi dalam buku panduan kompleks, sehingga mahasiswa membutuhkan waktu yang lama untuk memilah informasi yang mereka butuhkan.

Metode yang digunakan dalam pengembangan *chatbot* juga semakin bervariasi. Beberapa teknik yang umum digunakan adalah *Bidirectional Long Short-Term Memory*, *Recurrent Neural Network with Long Short-Term Memory* (RNN-LSTM), dan turunan dari LSTM, *Transformers*. *Transformers* adalah algoritma *deep learning* yang mengandalkan mekanisme *parallel multi-head attention* model transduksi urutan pertama yang sepenuhnya didasarkan pada *attention*(Vaswani *dkk.*, 2017).

Pada penelitian sebelumnya menggunakan dataset yang sama dengan metode LSTM yang diproses secara *sequence by sequence* dan masih berpaku pada *long term dependency*, pemrosesan ini sangat lama dan menyebabkan paralesasi dengan GPU tidak dapat dilakukan dan *long term dependency* menyebabkan pengolahan teks semakin banyak menjadi tidak efektif maka dari itu, penelitian ini menggunakan metode *transformers* dengan model *Bidirectional Encoder Representations from Transformers* (BERT) yang arsitekturnya menghilangkan semua ketergantungan jangka panjang dan proses berulang. BERT adalah teknik pra-pelatihan berbasis jaringan saraf untuk pemrosesan bahasa alami (NLP) yang disebut representasi *encoder* dua arah dari transformator (BERT). Lahirnya BERT bertujuan untuk membantu komputer memahami bahasa dengan lebih mudah seperti manusia (Afifa *dkk.*, 2023). Penulis menggunakan model BERT versi Indonesia yaitu IndoBERT. IndoBERT adalah modifikasi BERT Base yang sudah ada dengan menggunakan kofigurasi BERT-Base (*uncased*), IndoBERT dilatih dengan lebih dari 220 juta kata.

IndoBERT merupakan pilihan yang menarik untuk sistem yang lebih maju. Dengan dikembangkannya sistem ini diharapkan akan tercipta sistem *chatbot* yang akan memudahkan pengguna khususnya mahasiswa Teknik Informatika UNISSULA untuk mendapatkan informasi terkait Tugas Akhir.

1.2 Perumusan Masalah

Dirumuskan masalah selama penelitian ini yaitu:

1. Bagaimana *chatbot* sebagai Layanan Tugas Akhir Program Studi Teknik Informatika UNISSULA menggunakan metode BERT versi Indonesia mampu berkomunikasi dengan user.
2. Bagaimana performa dari hasil sistem *chatbot* menggunakan metode IndoBERT dengan sampel data dari dataset buku panduan Tugas Akhir Program Studi teknik Informatika UNISSULA.

1.3 Pembatasan Masalah

Dalam penelitian kali ini ada beberapa batasan masalah yang ditetapkan. Adapun batasan masalah dalam tugas akhir ini adalah:

1. Pertanyaan dan jawaban menggunakan bahasa Indonesia.
2. *Chatbot* tidak menerima input perhitungan atau karakter.
3. *Chatbot* tidak mengidentifikasi orang berdasarkan jenis kelamin, umur, atau nama.
4. Dataset terdiri dari data pertanyaan dan jawaban berdasarkan pada sumber panduan Tugas Akhir Program Studi teknik Informatika UNISSULA.
5. Topik Dialog dibatasi seputar informasi yang ada pada panduan Tugas Akhir Program Studi teknik Informatika UNISSULA.

1.4 Tujuan

Penelitian ini bertujuan untuk membangun sistem *Chatbot* layanan Tugas Akhir prodi Teknik Informatika yang dapat memberikan informasi terkait Tugas Akhir yang berpacu pada buku Panduan Tugas Akhir Prodi Teknik Informatika UNISSULA bagi mahasiswa untuk mempersingkat waktu dalam memberikan jawaban dari pertanyaan mengenai tugas akhir.

1.5 Manfaat

Manfaatnya antara lain efisiensi waktu bagi mahasiswa dalam mendapatkan informasi Tugas Akhir, kemudahan akses informasi, peningkatan kualitas informasi, pemberian bantuan 24/7, penyederhanaan proses komunikasi, peluang pengembangan keterampilan AI, dan peningkatan pengalaman mahasiswa dalam menyelesaikan studi.

1.6 Sistematika Penulisan

Sistematika penulisan berikut akan digunakan oleh penulis saat membuat laporan tugas akhir:

Tabel 1. 1 Sistematika Penulisan

BAB I	:	PENDAHULUAN
		Menguraikan latar belakang masalah agar dapat dipilih sebagai judul penelitian, perumusan masalah untuk menguraikan masalah yang akan dipecahkan, batasan masalah agar ruang lingkup masalah tidak terlalu luas, tujuan yang ingin dicapai, manfaat dari pembuatan sistem, dan sistematika penulisan yang mencakup uraian dari penulisan laporan tugas akhir.
BAB II	:	TINJAUAN PUSTAKA DAN DASAR TEORI
		Sebagai acuan untuk menyusun Tugas Akhir, mempelajari tinjauan literatur dan dasar teori yang digunakan untuk mendukung analisis masalah.
BAB III	:	METODE PPENELITIAN
		Memuat tentang analisis proses sistem chatbot sebagai layanan tugas akhir menggunakan model BERT. Analisa ini mencakup perancangan sistem dan desain antarmuka.
BAB IV	:	HASIL DAN PEMBAHASAN
		Memuat hasil pengujian program, pembahasan tentang prosedur kerja program, dan tampilannya.
BAB V	:	KESIMPULAN DAN SARAN
		Memuat tentang kesimpulan dan saran dari penulis terhadap penelitian yang telah dilakukan.

BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Chatbot adalah aplikasi dari bidang pemrosesan bahasa alami (NLP), *machine learning*, AI, dan rekayasa perangkat lunak. Mereka dapat berfungsi ketika menerima input bahasa alami yang terdiri dari beberapa kata dengan susunan berbeda yang digunakan dalam kalimat serta dapat mempelajari input sebelumnya yang memiliki arti yang sama menurut (Nugroho *dkk.*, 2021).

Dalam penelitian (Wijayanto *dkk.*, 2020), pengembangan *chatbot* informasi objek wisata diuji dengan menggunakan tiga metrik: unit, integrasi, dan validasi. Ketiga metrik tersebut menghasilkan nilai 100% validitas. Hasil pengujian kompatibilitas menunjukkan bahwa sistem dapat berjalan pada delapan jenis perambah.

Pada penelitian (Putu *dkk.*, (2023) tentang analisis sentimen pada Twitter sistem akan mengidentifikasi pesan para pengguna apakah termasuk perundungan siber atau tidak. Tingkat persentase akurasinya adalah 81%. Tingkat persentase presisinya adalah 80,67% untuk rata-rata kategori persentase presisinya adalah 80,33% untuk rata rata kategori pada data uji. Tingkat persentase *recall*nya adalah 80,33%. Tingkat *f1-score* persentasenya adalah 80,67% untuk rata-rata persentasenya adalah 80,33%. Dengan memberikan keluaran dalam bentuk kelas yang telah ditentukan, seperti netral, bahasa kasar, atau mengandung ujaran kebencian, sistem dapat membantu mengidentifikasi masukan yang mengarah pada perundungan siber. Selain itu, sistem memberikan persentase untuk kategori yang didapatkan melalui program.

Studi (Rayyan, 2022) akan membuat aplikasi yang dapat mendeteksi emosi berbasis indoBERT-lite untuk mencegah kesalahpahaman yang dapat menyebabkan konflik dan kerusakan hubungan saat berkomunikasi melalui teks. Berdasarkan hasil pengujian dengan jumlah *epoch* sejumlah 5, *batch size* 16, *learning rate* 0,000003, *optimizer* ADAM, dan menggunakan IndoBERT-liteLARGE fase pertama, didapat nilai *accuracy, Fi score, recall,* dan *precision*

pada *training set* masing- masing sejumlah 89%, 89%. 89%, dan 90%, sedangkan pada *validation set* didapat 70%, 71%, 70%, dan 72%. Hasil pengujian oleh 6 responden dengan total 30 kalimat menunjukkan model memiliki akurasi sebesar 76,7%.

Sistem ini akan menggunakan metode pemrosesan bahasa natural (NLP). Salah satu bidang ilmu AI yang berfokus pada pengolahan bahasa natural adalah Natural Language Processing (NLP). Bahasa yang biasa digunakan manusia untuk berbicara satu sama lain disebut bahasa natural. Dalam penelitian sebelumnya yang dilakukan untuk menyaring gejala depresi pada pengguna Twitter, pendekatan leksikon dan proses pemrosesan bahasa natural menjadi lebih efektif daripada pendekatan machine learning. (Rohman *dkk.*, 2019).

Hasil penelitian (Afifa *dkk.,* 2023) menunjukkan bahwa sistem chatbot yang dirancang dengan menggunakan algoritma (BERT) dan framework RASA memungkinkan chatbot untuk menjawab pertanyaan yang diberikan oleh pengguna. Hasil dari pengujian Non-Response Rate menunjukkan bahwa sistem dapat mengenali berbagai pola kalimat dengan akurasi sebesar 85%. Ini disebabkan oleh adanya proses BERT, yang membantu menganalisis kalimat yang dimasukkan oleh pengguna..

Penelitian oleh (Isa *dkk.,* 2022) bertujuan untuk mengurangi penyebaran berita palsu di Indonesia, dengan membuat deteksi berita palsu menggunakan indoBERT sebagai solusinya. *Dataset* diperoleh dari turnbackhoax.id dengan 3.465 berita palsu dan 766 berita asli. Dari Penelitian ini telah dibuktikan bahwa membuat deteksi berita palsu di Indonesia menggunakan model IndoBERT bisa dilakukan dengan tingkat akurasi sebesar 94,66%.

Dari keenam hasil penelitian diatas dapat diambil kesimpulan bahwa teknik *Natural Language Processing* (NLP) dengan menggunakan arsitektur *Bidirectional Encoder Representations from Transformers* (BERT) dan algoritma indoBERT cocok jika diimplementasikan pada sistem *chatbot* karena memudahkan memahami informasi yang dibutuhkan sesuai dengan bahasa alami user. NLP juga memiliki akurasi lebih baik dari pada menggunakan metode *machine learning*. Harapannya, penelitian ini dapat menghasilkan sistem *chatbot*

yang memberikan kemudahan pada mahasiswa FTI untuk mencari informasi mengenai panduan akademik yang mereka butuhkan secara cepat dan akurat.

2.2 Dasar Teori

2.2.1 *Natural Language Processing*

Salah satu cabang ilmu kecerdasan buatan (AI) yang dikenal sebagai pengolahan bahasa natural (NLP) berkonsentrasi pada pengolahan bahasa natural, yaitu bahasa yang biasa digunakan manusia untuk berbicara satu sama lain. (Rohman *dkk.,* 2019).

Dalam bidang NLP, terdapat berbagai topik yang dibahas, salah satunya adalah parsing. *Parsing* adalah proses membedah kalimat menjadi bagian-bagian komponennya berdasarkan kategori kata *(parts of speech)* untuk melihat tata bahasanya. Analisis ini bertujuan untuk membantu tugas pemrosesan yang lebih kompleks pada tahap selanjutnya, yang disebut pemrosesan hilir.

2.2.2 *Chatbot*

Layanan informasi sangat penting untuk lingkungan pendidikan karena dapat meningkatkan kinerja mahasiswa. *Chatbot* dan sistem lain seperti itu muncul sebagai hasil kemajuan teknologi yang membantu orang menjawab pertanyaan.

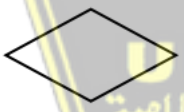
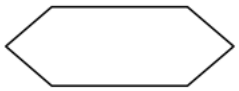
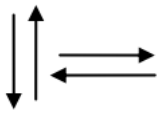
Di era teknologi saat ini, *chatbot* masih menjadi perangkat lunak komputer. Untuk tujuan ini, perusahaan teknologi telah membuat aplikasi mereka sendiri, seperti Siri milik Apple dan Alexa milik Amazon. Menurut buku tentang Sistem Percakapan Otomatis Berbasis Layanan Pesan Instan (Muhammad *dkk.,* 2022), *chatbot* berfungsi sebagai penjawab sotomatis yang menjawab pertanyaan pengguna sesuai konteks, baik secara lisan maupun teks.

Chatbot biasanya memiliki obrolan singkat, tetapi ini berbeda dari pembicaraan nyata, yang melibatkan pertemuan langsung. *Chatbot* tidak benar-benar berbicara, meskipun mereka dirancang untuk meniru orang. *Chatbot* dianggap sebagai jenis kecerdasan buatan (AI) karena mereka dapat "belajar" dan meningkatkan akurasi dengan data tambahan.

2.2.3 *Flowchart*

Flowchart adalah langkah-langkah pemecahan masalah yang ditulis dalam simbol-simbol tertentu. *Flowchart* ini akan menampilkan alur dalam program secara logika. Flowchart ini tidak hanya digunakan sebagai alat komunikasi tetapi juga diperlukan sebagai panduan dan sebelum pemahaman yang lebih baik dari komponen-komponen *flowchart* (Khesya, 2021). Berikut ini simbol-simbol dan kegunaan dari simbol-simbol yang sering digunakan untuk menggambarkan suatu algoritma dalam bentuk diagram alir, yaitu:

Tabel 2. 1 Simbol Flowchart dan Fungsinya

No	Simbol	Nama	Fungsi Simbol
1		*"Terminal"*	Awal atau akhir suatu program (Prosedur).
2		*"Output/Input"*	Proses input atau output terlepas dari jenis perangkat.
3		*"Procces"*	Proses operasional computer.
4		"*Decision*"	Untuk menunjukkan bahwa suatu kondisi tertentu mengarah pada dua kemungkinan, ya/tidak.
5		*"Predefined Process"*	Mewakili ketentuan penyimpanan untuk diproses untuk memberikan awal harga.
6		*"Flow"*	Menyatakan jalannya arus suatu proses.

2.2.4 *Data pre processing*

2.2.4.1 *Text Processing*

Pemrosesan teks meliputi meghapus tanda baca dan mengubah semua kalimat menjadi huruf kecil. Tujuan penghapusan tanda baca dalam pemrosesan teks adalah untuk menghilangkan tanda baca dari teks. Karakter non alfanumerik, seperti titik, koma, tanda tanya, tanda seru, tanda kutip, dan tanda baca lainnya, Pembersihan ini dapat membantu dalam pengidentifikasian pola atau informasi yang lebih relevan. Sedangkan mengonversi ke huruf kecil dapat mengurangi kompleksitas dan membuat data lebih seragam.

Dengan menghilangkan karakter non-alphanumeric dan mengonversi ke huruf kecil, data menjadi lebih konsisten, mudah diinterpretasikan, dan siap untuk analisis lanjutan. Prosedur ini sering digunakan sebelum tokenization atau langkah pra-pemrosesan lainnya untuk mendapatkan teks yang sesuai untuk analisis selanjutnya.

2.2.4.2 *Data Labeling*

Dalam metode *supervised learning* untuk *chatbot, data labeling* diperlukan karena Sistem akan mencari pola dari *dataset*, kemudian pola itu akan dijadikan sebagai acuan untuk kumpulan data berikutnya (Abijono *dkk.*, 2021). Ini melibatkan memberikan label atau anotasi pada pasangan pertanyaan dan jawaban dalam percakapan sehingga model *chatbot* dapat belajar merespons dengan benar. Proses *data labeling* adalah langkah penting dalam mengembangkan *chatbot* yang efektif.

2.2.4.3 *Tokenizers*

Tokenisasi untuk model BERT menggunakan teknik tokenisasi WordPiece. Dengan menggunakan WordPiece, model BERT dapat memahami makna dan konteks kata-kata dengan lebih baik karena WordPiece memecah teks menjadi *subword-subword* yang dapat digunakan untuk membentuk token-token yang lebih besar (Jurafsky dan James, 2017).

2.2.4.4 *Padding* dan *Truncation*

Input batch seringkali memiliki panjang yang berbeda, jadi tidak dapat dikonversi ke tensor ukuran tetap. Untuk mengatasi masalah ini, *padding* dan *truncation* digunakan untuk membuat tensor persegi panjang dari kumpulan panjang yang berbeda. *Padding* menambahkan token *padding* khusus untuk memastikan bahwa urutan yang lebih pendek memiliki panjang yang sama dengan urutan terpanjang dalam urutan atau bahkan lebih panjang dari yang dapat diterima model. Sebaliknya, jika kalimat terlalu panjang, lakukan pemotongan *(truncation)* (HuggingFace, 2020).

2.2.4.5 *Attention Masking*

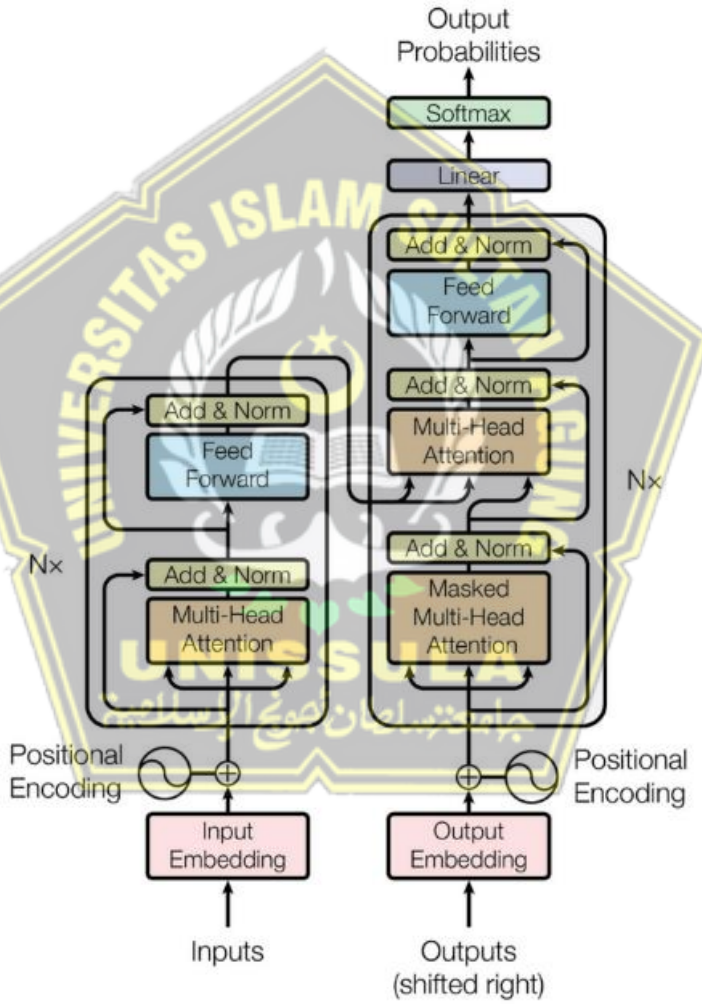
Attention masking adalah strategi yang digunakan dalam tugas *chatbot* yang menggunakan model BERT untuk mengontrol perhatian model terhadap token dalam percakapan. Dua jenis *attention* masking yang dapat digunakan adalah *Look-Ahead Masking*, yang memastikan bahwa model hanya dapat mengakses token saat ini dalam percakapan dan tidak "melihat" token di masa depan. *Masking Padding,* Mengabaikan *padding* token dalam urutan teks untuk menghindari pengaruh pada respons *chatbot* (Vaswani *dkk.*, 2017). Kedua jenis masking ini membantu model BERT memahami percakapan dan membuat respons yang tepat dan relevan.

2.2.4.6 Konversi ke Tensor

BERT sering digunakan bersama dengan *framework* seperti PyTorch atau TensorFlow dalam pemrosesan bahasa alami, jadi *batch* data yang telah diindeks dan diembedding akan diubah menjadi tensor khusus dari *library* yang digunakan, dalam penelitian ini menggunakan *library* PyTorch. Setelah proses konversi ke tensor ini, data teks dapat dimasukkan untuk analisis atau prediksi ke dalam model BERT. Dengan demikian, model dapat melakukan komputasi dengan data dalam format tensor untuk menghasilkan respons atau prediksi yang relevan (HuggingFace, 2020).

2.3 *Transformers*

Transformers adalah model *deep learning* yang menggunakan mekanisme *self-attention*, yang mengidentifikasi hubungan antar kata dalam konteks. Transformers digunakan terutama dalam pemrosesan bahasa alami dan *computer vision*. Model ini dimaksudkan untuk memproses tipe data sekuens seperti bahasa alami, yang memungkinkannya melakukan tugas seperti menerjemahkan bahasa dan merangkum teks. Transformer terdiri dari dua *stacks*(Vaswani *dkk.*, 2017).



Gambar 2. 1 Model Arsitektur Transformers(Vaswani *dkk.*, 2017)

Arsitektur umum ini diikuti oleh transformator dengan menggunakan lapisan *self-attention stacked* dan *point-wise*, sepenuhnya terhubung dengan *encoder* dan *decoder*, seperti yang ditunjukkan di bagian kiri dan kanan Gambar 6.1, masing-masing.

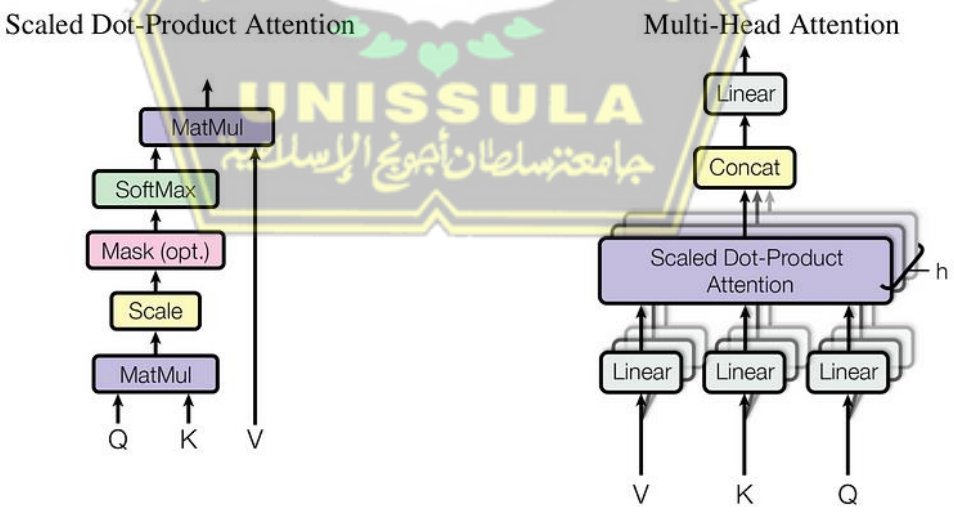
2.3.1.1 *Encoder* dan *Decoder*

Encoder terdiri dari tumpukan N = 6 lapisan yang identik, dengan dua sublapisan di setiap lapisan encoder: yang pertama adalah mekanisme *self-attention multi-head*, dan yang kedua adalah jaringan *feed-forward* yang sederhana dan terhubung secara posisi.

Decoder juga terdiri dari tumpukan N = 6 lapisan yang identik, dengan dua sublapisan di setiap lapisan *encoder*. Selain itu, *decoder* menyisipkan sublapisan ketiga, yang melakukan *multi-head attention* dari *output* tumpukan *encoder.* (Vaswani *dkk.*, 2017).

2.3.1.2 *Attention*

Sederhananya, fungsi *attention* dapat digambarkan sebagai pemetaan *query* dan penetapan *key-value* ke sebuah *output*. Dalam hal ini *query, key, value,* dan *output* adalah vektor, dan jumlah tertimbang dari nilai-nilai dihitung sebagai *output.* Selain itu, fungsi kompatibilitas *query* dengan *key* yang sesuai menentukan bobot yang ditetapkan untuk setiap *value* (Vaswani *dkk.*, 2017).



Gambar 2. 2 (kiri) Scaled Dot-product Attention. (kanan) Multi-head Attention terdiri dari beberapa lapisan perhatian yang berjalan secara paralel(Vaswani *dkk.*, 2017)

2.3.1.3 *Scaled Dot-Product Attention*

Vektor kueri (*query*), vektor kunci (*key*), dan vektor nilai (*value*) adalah tiga vektor yang diterima oleh *Scaled Dot-Product Attention*. Ini adalah ide dasar dari *self-attention*. Perhitungan dilakukan antara vektor *query* dan vektor *key* untuk setiap pasangan *query-key* menghasilkan berbagai skor perhatian yang menunjukkan seberapa banyak komponen dalam data input berkontribusi pada komponen lain dalam konteks *self-attention.* Skalabilitas, atau akar kuadrat dari dimensi vektor kueri, dibagi oleh nilai skor perhatian agar tetap stabil saat dimasukkan ke dalam fungsi *softmax*. Untuk mengontrol banyaknya skor dan memastikan bahwa model dapat belajar perhatian yang halus, ini disebut *"scaling"*. Skor perhatian yang telah disesuaikan dengan skala digunakan sebagai bobot untuk menghitung vektor nilai rata-rata tertimbang, menunjukkan seberapa besar kontribusi setiap nilai terhadap vektor hasil akhir. Proses *Scaled Dot-Product Attention* menghasilkan vektor hasil perhatian. Mekanisme ini diulang untuk setiap pasangan kunci pertanyaan untuk memahami hubungan antara elemen-elemen tersebut. Hasilnya adalah matriks perhatian yang menunjukkan relevansi setiap elemen dalam data. Kueri ini dikemas ke dalam sebuah matriks Q, kunci dan nilai juga dikemas ke dalam matriks K dan V (Vaswani *dkk.*, 2017). Matriks keluaran dihitung sebagai berikut:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

Keterangan :

Q = Kueri

K = Kunci

V = Nilai

d_k = Kueri dan Kunci dimensi

d_v = Nilai dari dimensi

2.3.1.4 *Multi-Head Attention*

Multi-Head Attention menerima tiga vektor sebagai input; ini termasuk vektor kueri (*query*), vektor kunci (*key*), dan vektor nilai (*value*). Sebelum perhitungan perhatian dilakukan, setiap vektor kueri, kunci, dan nilai diubah menjadi sejumlah subruang melalui proyeksi linier. Untuk proyeksi ini, matriks pembobotan yang berbeda digunakan untuk masing-masing kepala(*head*). Setiap kepala menghitung skor perhatian mereka sendiri, yang menghasilkan sejumlah matriks perhatian yang berbeda. Ini dicapai melalui perhitungan produk dot antara vektor kueri dan vektor kunci setelah proyeksi linier. Setelah itu, *softmax* dan *scaling* digunakan untuk perhitungan perhatian. Matriks perhatian akhir dibuat dengan menggabungkan hasil dari setiap kepala. Hasil akhir yang lebih halus dapat dicapai dengan menggunakan proyeksi linier tambahan. Seperti pada *Scaled Dot-Product Attention*, matriks perhatian digunakan untuk mengambil rata-rata tertimbang dari vektor nilai. Ini menghasilkan vektor hasil yang menunjukkan interaksi antara komponen dalam data *input* (Vaswani *dkk*., 2017).

$$MultiHead(Q,K,V) = Concat(head_1,\dots,head_h)W^O$$
$$where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (2)$$

Keterangan:

$MultiHead(Q,K,V)$ = operasi *multi-head attention* mengambil tiga input, yaitu matriks *query* (Q), matriks *key* (K), dan matriks *value* (V).

$Concat(head_1,\dots,head_h)$ = operasi penggabungan (*concatenation*) dari hasil-hasil dari setiap kepala (*head*) yang diberikan oleh fungsi *Attention*. Hasil dari setiap kepala dikonkatenasi menjadi satu vektor besar.

W^O= matriks pembobot akhir yang digunakan untuk mentransformasi hasil penggabungan (*Concat*) dari kepala-kepala *attention* menjadi representasi akhir.

$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$ = perhitungan *attention* pada masing-masing kepala (*head*). Setiap kepala memiliki matriks pembobot sendiri, yaitu QW_i^Q, KW_i^K, dan VW_i^V, yang digunakan untuk mengubah matriks *query* (Q), *key* (K), dan *value* (V) menjadi representasi yang sesuai untuk kepala tersebut.

2.3.1.5 *Position-wise Feed-forward Networks*

Selain sub-lapisan perhatian, masing-masing lapisan *encoder* dan *decoder* terdiri dari jaringan *feed-forward* yang terhubung sepenuhnya, yang diterapkan secara terpisah dan identik ke setiap posisi. Ini terdiri dari dua transformasi linier yang masing-masing memiliki aktivasi ReLU.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \quad (3)$$

Transformasi linier memiliki parameter yang berbeda di setiap lapisan, meskipun mereka identik pada posisi yang berbeda. Dua konvolusi dengan ukuran kernel 1 adalah cara lain untuk menggambarkan hal ini(Vaswani *dkk.*, 2017).

2.3.1.6 *Embeddings and Softmax*

Dengan menggunakan *embedding* yang dipelajari, model ini mengonversi token input dan output menjadi vektor berdimensi. Selain itu, model ini menggunakan transformasi linier dan fungsi *softmax*, yang biasa digunakan untuk mengkonversi *output decoder* ke kemungkinan token berikutnya. Model ini menggunakan matriks bobot yang sama antara dua lapisan penyisipan dan transformasi linier pra-*softmax*. Pada lapisan penyisipan, model mengalikan bobot-bobot tersebut dengan vektor-vektor berdimensi (Vaswani *dkk.*, 2017).

2.3.1.7 *Positional Encoding*

Untuk menggunakan urutan urutan, perlu menyuntikkan informasi tentang posisi relatif atau absolut token dalam urutan karena model ini tidak mengandung perulangan dan konvolusi. Untuk melakukannya, "*positional encodings*" harus ditambahkan ke penyematan input di bagian bawah tumpukan *encoder* dan *decoder*. *Positional Encodings* memiliki dimensi yang sama dengan vektor yang sama dengan *embedding*, sehingga keduanya dapat digabungkan. Banyak opsi untuk *positional encodings* yang dipelajari dan diperbaiki. Fungsi sinus dan kosinus digunakan dengan frekuensi yang berbeda dalam tugas ini:

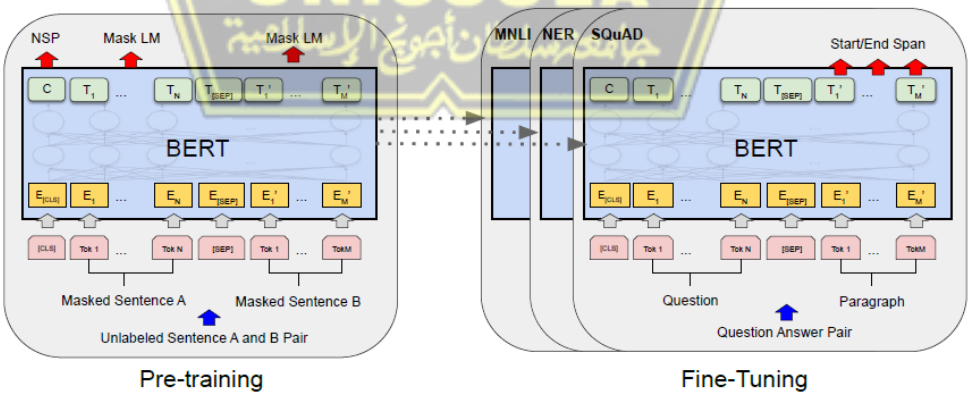
$$PE_{(pos,2i)} = sin\ (pos/10000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = \cos\ (pos/10000^{2i/d_{model}}) \quad (4)$$

di mana dimensi adalah i dan pos adalah posisi. Dengan kata lain, setiap dimensi pengkodean posisi berkorelasi dengan *sinusoid*. Panjang gelombang membentuk perkembangan geometris dari 2π hingga 10000 . 2π. Kami memilih fungsi ini karena kami berhipotesis bahwa fungsi ini akan memungkinkan model belajar dengan mudah untuk hadir dengan posisi relatif, karena PE_{pos+k} dapat digambarkan sebagai fungsi linier PE_{pos} untuk setiap *offset* tetap k (Vaswani *dkk.*, 2017).

2.3.2 *Bidirectional Encoder Representations from Transformers* (BERT)

BERT adalah salah satu model arsitektur Transformer yang mengubah cara kita memahami pemahaman bahasa alami. BERT bertujuan untuk mengajarkan representasi dua arah yang mendalam dari teks yang tidak berlabel dengan mengkondisikan secara bersama-sama disetiap lapisan konteks. Model BERT yang telah dilatih sebelumnya dapat disetel ulang hanya dengan menambahkan satu lapisan keluaran tambahan. Ini memungkinkan untuk membuat model baru untuk berbagai tugas, seperti menjawab pertanyaan dan menyimpulkan bahasa, tanpa mengubah arsitektur tugas yang signifikan (Devlin *dkk.*, 2019).

2.3.2.1 Pre-Training dan *Fine-Tuning*



Gambar 2. 3 Arsitektur *Pre-Training* dan *Fine-Tuning* dalam BERT.

Proses *pre-training* dan penyempurnaan BERT secara keseluruhan menggunakan arsitektur yang sama, terlepas dari lapisan *output* parameter *pre-training* yang sama digunakan untuk menginisialisasi model untuk berbagai tugas

down-stream. Semua parameter disetel dengan baik selama proses *fine-tuning*. [SEP] adalah tanda pemisah khusus (memisahkan pertanyaan/jawaban), dan [CLS] adalah simbol khusus simbol khusus yang disertakan di depan setiap contoh masukan (Devlin dkk., 2019).

Tahap *pre-training* BERT melibatkan pelatihan model pada korpus teks yang sangat besar. Selama tahap ini, model diajari untuk memprediksi kata yang diacak dalam konteks kalimat. Setelah tahap *pre-training*, model BERT yang sudah dilatih dapat di-"*fine-tune*" untuk tugas-tugas spesifik. Selama tahap *fine-tuning*, beberapa lapisan di atas model BERT dapat disesuaikan untuk tugas khusus yang digunakan pada penilitian ini untuk menjawab pertanyaan. Ini memungkinkan BERT untuk menyesuaikan diri dengan tugas tertentu dan menghasilkan hasil yang lebih baik (Devlin *dkk.*, 2019).

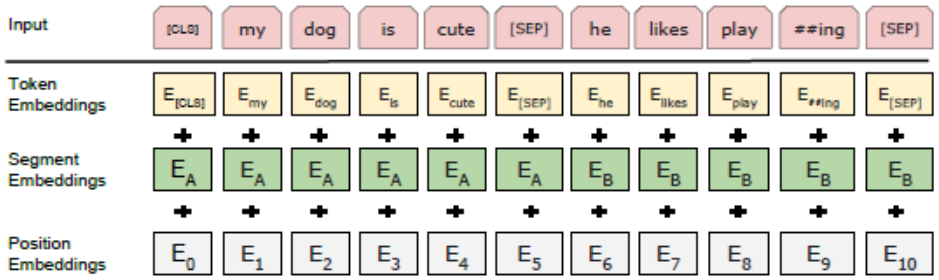
2.3.2.2 Bidirectional Context

BERT unggul dalam berbagai tugas pemrosesan bahasa alami karena "*bidirectional context*", kemampuan model untuk memahami kata-kata dalam konteks dua arah, yang memungkinkan pemahaman konteks yang lebih luas dan pemrosesan bahasa yang lebih efisien (Devlin *dkk.*, 2019).

2.3.2.3 Maksed Language Model

Masked Language Model (MLM) adalah tugas *pre-training* yang mengajarkan model untuk memprediksi kata-kata yang diacak dalam konteks kalimat. Beberapa token dalam kalimat diacak, dan model diajarkan untuk memprediksi kata-kata yang diacak berdasarkan konteks sekitarnya. Ketika MLM bekerja, model BERT harus memahami hubungan antar kata dalam konteks kalimat, yang merupakan dasar pemahaman bahasa yang mendalam (Devlin *dkk.*, 2019).

2.3.2.4 Next Sentences Model (NSP)



Gambar 2. 4 Penyematan input BERT adalah jumlah penyematan token, segmentasi, dan posisi(Devlin *dkk.*, 2019).

Next Sentence Prediction (NSP) adalah salah satu tugas *pre-training* yang membantu model BERT memahami hubungan antar kalimat dan konteks lintas kalimat. Tugas ini membantu model BERT memprediksi apakah kalimat kedua adalah kalimat berikutnya dari kalimat pertama. Ini adalah komponen penting dari tugas pre-training BERT untuk memahami struktur bahasa (Devlin *dkk.*, 2019) .

2.3.2.5 Self Attention Mechanism

Model seperti BERT (*Bidirectional Encoder Representations from Transformers*) menggunakan mekanisme *self-attention*, yang merupakan komponen penting dari arsitektur transformer. Ini adalah metode yang memungkinkan model untuk "memperhatikan" hubungan antara kata-kata dalam kalimat atau urutan data yang dimasukkan. Mekanisme *self-attention* terdiri dari dua operasi utama yaitu: perhitungan skor perhatian (*attention scores*) dan penggabungan jumlah berat dari nilai vektor-vektor (Vaswani *dkk.*, 2017).

2.3.2.6 Transfer Learning

Transfer learning dalam BERT memungkinkan model untuk memanfaatkan pengetahuan bahasa yang telah diperoleh selama *pre-training* dan mengaplikasikannya pada tugas pemrosesan bahasa yang lebih spesifik melalui *fine-tuning.* Ini membuat BERT menjadi salah satu pendekatan yang paling efektif dalam NLP, karena model sudah memiliki pemahaman bahasa yang mendalam sebelum diadaptasi ke tugas-tugas khusus (Devlin *dkk.*, 2019).

2.3.2.7 Evaluasi Klasifikasi

Evaluasi melibatkan pengukuran kinerja model pada tugas spesifik, seperti klasifikasi teks, menggunakan metrik yang sesuai seperti *precision*, *recall*, *accuracy* dan *F1-score*. Evaluasi adalah cara untuk menilai sejauh mana BERT berhasil dalam tugas tersebut dan memastikan kualitas hasil (Jurafsky dan James, 2017).

Rumus persamaannya sebagai berikut:

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

$$recall = \frac{TP}{TP + FN} \quad (6)$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \quad (7)$$

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP} \quad (8)$$

Keterangan :

True Positive (TP) = adalah jumlah prediksi yang berhasil dibuat oleh model yang memprediksi kelas positif.

True Negatif (TN) = adalah jumlah prediksi yang berhasil dibuat oleh model yang mengantisipasi bahwa kelas negatif akan negatif.

False Positive (FP) = adalah jumlah prediksi yang salah yang dibuat oleh model yang memperkirakan kelas negatif akan menjadi positif.

False Negatif (FN) = adalah jumlah prediksi yang tidak tepat yang dibuat oleh model yang menduga kelas negatif..

2.3.3 IndoBERT

IndoBERT adalah model berbasis transformator dengan gaya BERT, tetapi dilatih secara eksklusif sebagai model bahasa masked menggunakan rangka kerja Huggingface, menggunakan konfigurasi standar untuk BERT-Base (uncased). Model ini memiliki dua belas lapisan tersembunyi yang masing-masing memiliki 768d, dua belas kepala perhatian, dan lapisan *feed-forward* tersembunyi yang memiliki 3.072d. *Framework* Huggingface diubah untuk membaca berbagai aliran

teks untuk berbagai blok dokumen, dan kami mengatur pelatihan dengan 512 token per batch. IndoBERT dilatih dengan 31.923 kosakata WordPiece bahasa Indonesia. Tiga sumber utama, Web Corpus Indonesia (90 juta kata), artikel berita dari Kompas, Tempo, dan Liputan6 (55 juta kata), dan Wikipedia Bahasa Indonesia (74 juta kata), telah digunakan untuk melatih IndoBERT. 1.067.581 contoh latih dan 13.985 contoh pengembangan (tanpa reduplikasi) diperoleh melalui prapemrosesan korpus menjadi 512 blok dokumen bertoken. Kami menggunakan empat GPU Nvidia V100, masing-masing 16GB, dengan ukuran *batch* 128, laju pembelajaran 1e-4, pengoptimal Adam, dan penjadwal linier untuk pelatihan. Kami melatih model sebanyak 2,4 juta langkah (180 *epoch*) selama dua bulan kalender, dan *perplexity* akhir dari set pengembangan adalah 3,97 (sebanding dengan base BERT bahasa Inggris) (Koto *dkk.*, 2020).

BAB III
METODOLOGI PENELITIAN

3.1 Metode Penelitian

Penelitian ini menggunakan metode atau algoritma yang disebut IndoBERT atau versi Indonesia dari *Bidirectional Encoder Representations from Transformers* (BERT) dan menggunakan *Natural Language Processing* (NLP). Keluaran dari sistem ini adalah *Chatbot* yang mampu berkomunikasi dengan user dengan memberikan responses jawaban sesuai dengan apa yang diinputkan atau ditanyakan. Dalam penelitian ini ada beberapa Langkah – langkah yang harus dilakukan seperti yang terlihat dalam *flowchart* gambar 3.1 :

Gambar 3. 1 *Flowchart* Metode Penelitian

Pada gambar *Flowchart* Metode penelitian memperlihatkan langkah – langkah yang harus dilakukan dalam penelitian yaitu Studi Literatur, pengumpulan data, dan pemodelan sistem.

3.1.1 Studi Literatur

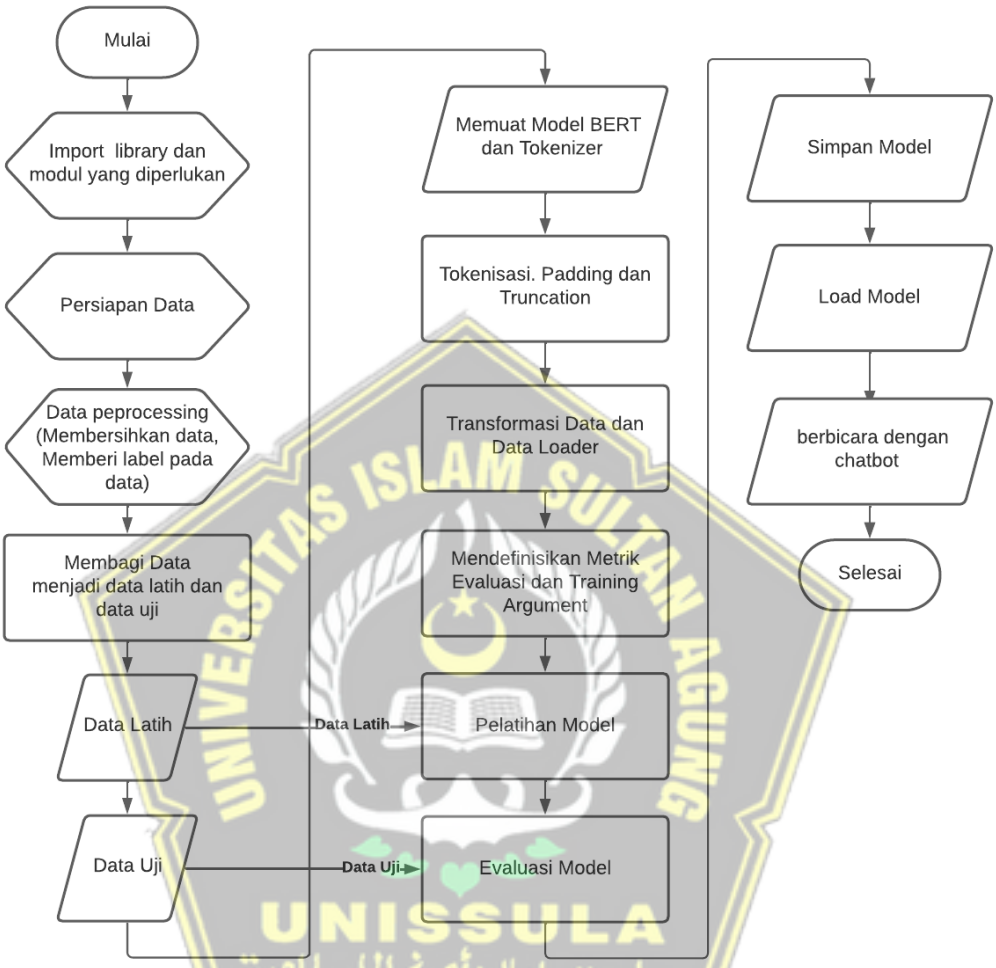
Dalam penelitian ini akan dilakukan tinjauan terhadap beberapa e-book, makalah, jurnal, tesis, dan skripsi terdahulu akan diulas selain mengunjungi berbagai situs web. Tujuan dari tinjauan ini adalah untuk mempelajari teori di balik konsep seperti *Bidirectional Encoder Representations from Transformers* (BERT) dan *Natural Language Processing* (NLP).

3.1.2 Pengumpulan Data

Pada tahap ini, persiapan dan pengembangan model Data yang dikumpulkan berasal dari panduan Tugas Akhir Prodi Teknik Informatika UNISSULA. Data ini akan digunakan untuk membuat dataset yang terdiri dari pertanyaan dan jawaban.

3.1.3 Pemodelan Sistem

Rancangan Pemodelan Sistem dapat dilihat dalam *flowchart* gambar 3.1:



Gambar 3. 2 *Flowchart* pemodelan Sistem

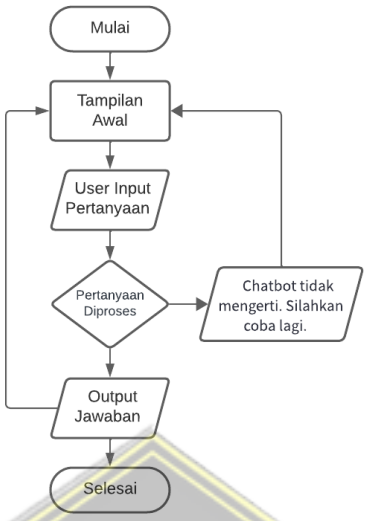
Pada tahap pemodelan sistem dibagi menjadi beberapa tahapan seperti Persiapan Data, *Data Preprocessing*, dan Modelling.

1) Persiapan Data dimulai dari Memuat dataset, peneliti akan memuat dataset berformat file JSON yang terdiri dari *tag*, *patterns*, dan *response* yang akan digunakan untuk melatih *chatbot.* Kemudian Mengekstrak Informasi dari file JSON dan disimpan dalam *Data Frame* untuk memudahkan analisis data. Setelah itu Menampilkan informasi tentang dataset seperti tipe data, jumlah nilai yang tidak null, jumlah kelas, dan jumlah nilai null.

2) *Data Preprocessing*, dalam tahap ini melibatkan persiapan data untuk pelatihan yang mencakup membersihkan teks dari karakter khusus atau tanda baca yang tidak relevan serta mengonversi semua teks ke huruf kecil, memberi label pada data. Kemudian membagi dataset menjadi dua bagian: data latih dan data uji. Data latih digunakan untuk melatih *chatbot*, data uji digunakan untuk menguji kinerja *chatbot*,
3) Modelling dimulai dari memuat model BERT yang sudah dilatih bersama dengan *tokenizer* yang sesuai, *tokenizer* BERT digunakan untuk transformasi data ke format numerik. Tokenisasi teks, memecah teks menjadi token atau kata – kata yang lebih kecil, Konversi token-token menjadi ID token berdasarkan kamus token model BERT, *Padding* dan *Truncation* untuk memastikan panjang setiap teks seragam. Kemudian membuat *Data Loader* untuk memfasilitasi pelatihan model. Kemudian menentukan Matrik Evaluasi yang akan digunakan untuk mengukur kinerja *chatbot*, seperti akurasi, *F1-score finetune*, *precision*, dan *recall*. Selanjutnya menentukan parameter pelatihan seperti ukuran *batch*, jumlah *epoch*, dan *learning rate*. Kemudian train model dengan data train menggunakan model BERT. Setelah pelatihan selesai evaluasi model menggunakan data uji, hasilnya akan membantu untuk memahami seberapa baik kinerja *chatbot*. kemudian simpan model untuk digunakan nantinya. Ketika ingin menggunakan *chatbot*, bisa memuat model yang sudah dilatih.

3.2 Analisis Sistem

Pada Penelitian ini penulis akan membuat sistem *chatbot* berbasis website yang bertujuan untuk memberikan layanan informasi tugas akhir kepada mahasiswa di program studi Teknik Informatika UNISSULA. Untuk alurnya akan digambarkan pada gambar 3.2 dengan *flowchart*.



Gambar 3. 3 *Flowchart* Sistem *Chatbot*

Pada gambar 3.2 adalah *flowchart* yang menggambarkan proses dari sistem *chatbot.* Ada beberapa tahapan pada gambar 3.2 yaitu :

a. Pertama *User* memasuki Halaman tampilan awal dari website aplikasi *chatbot*
b. Kemudian *User* memasukkan pertanyaan yang ingin ditanyakan pada kolom *input,* setelah itu tekan tombol "*Enter"* atau klik *icon* pesawat kertas yang ada disebelah kanan kolom *input text*.
c. Selanjutnya chatbot akan mulai memproses pertanyaan/inputan yang diberikan oleh *User* dengan mencocokan data jawaban yang sudah dilatih.
d. Jika Pertanyaan yang dikirim memiliki kecocokan dengan data yang sudah dilatih oleh model maka sistem akan memberikan jawaban*/responses* yang sesuai dan akan menampilkan pertanyaan dan jawaban pada kolom percakapan.

3.3 Analisis Kebutuhan

Pada tahap analisis kebutuhan, peneliti memeriksa semua perangkat lunak yang diperlukan untuk membuat aplikasi chatbot ini beroperasi dengan baik dan menghasilkan hasil yang diinginkan. Sistem ini dibuat dengan menggunakan program berikut:

1. Python 3.10.5

   Python adalah bahasa pemrograman tingkat tinggi yang memiliki sintaksis yang sederhana untuk dibaca. Selain mendukung berbagai aplikasi, ia serbaguna dan dapat digunakan di berbagai platform. Penelitian ini menggunakan Python versi 3.10.5 karena bersifat *open source*, komunitasnya besar, dan ada banyak sumber daya online. Python menjadi pilihan populer untuk pengembangan perangkat lunak karena fokusnya pada produktivitas, manajemen memori otomatis, dan integrasi mudah dengan bahasa lain.

2. *Library Transformers*

   Transformers memberikan akses ke model transformer dan alat pemrosesan bahasa alami (NLP) yang kuat. Dalam penelitian ini, memungkinkan penggunaan model NLP canggih seperti BERT untuk pemahaman bahasa yang lebih baik dan generasi jawaban yang lebih kontekstual.

3. *Library Pytorch*

   PyTorch adalah *library open-source* untuk komputasi tensor dan pembelajaran mesin dan menjadi salah satu *framework deep learning* yang populer. *PyTorch* menyediakan *framework deep learning* yang kuat. Dalam penelitian ini, *PyTorch* dapat digunakan untuk membangun dan melatih model *transformers*. Fitur-fitur dinamis *PyTorch* dan dukungan GPU dapat meningkatkan efisiensi dan kinerja model.

4. *Library Pandas*

   Pandas menyediakan struktur data seperti DataFrame untuk analisis data yang efisien. Dalam penelitian ini, Pandas dapat digunakan untuk mengelola dan menganalisis data seperti riwayat percakapan atau dataset yang digunakan untuk melatih model.

5. *Library Json*

   JSON *(JavaScript Object Notation)* digunakan untuk menyimpan dan bertukar data dalam format yang mudah dibaca oleh manusia dan mudah diproses oleh komputer. Dalam penelitian ini, JSON dapat digunakan untuk menyimpan dan mengelola konfigurasi, dataset, atau hasil percakapan.

6. *Library Random*

   Library Random dapat digunakan untuk menghasilkan angka acak. Dalam penelitian ini, bisa berguna untuk menyematkan unsur kejutan atau variasi dalam generasi jawaban atau perilaku chatbot.

7. *Library Scikit-Learn*

   Library scikit-learn digunakan untuk pembelajaran mesin (*machine learning*) dan ditulis dalam bahasa pemrograman Python. Dalam penelitian ini, *library scikit-learn* digunakan untuk mengevaluasi performa model dengan mengukur akurasi, presisi, dan recall model yang telah dilatih.

8. *Visual Studio Code*

   Visual Studio Code dipilih sebagai editor kode yang pada pengembangan aplikasi dalam penelitian ini karena mudah digunakan yang memiliki ekosistem ekstensi yang luas, integrasi Git, terminal terintegrasi, dukungan *framework* dan bahasa pemrograman yang banyak serta pengenal bahasa pemrograman pintar yang memudahkan proses pengembangan.

9. *Google Colaboratory*

   Google Colab menyediakan lingkungan *cloud* yang memungkinkan pengembangan dan pelatihan model di lingkungan dengan akses GPU atau TPU, sangat penting untuk model-model berat seperti *Transformers*. Dalam penelitian ini digunakan untuk menulis dan menjalankan kode program, serta menyimpan dan mendokumentasikan.

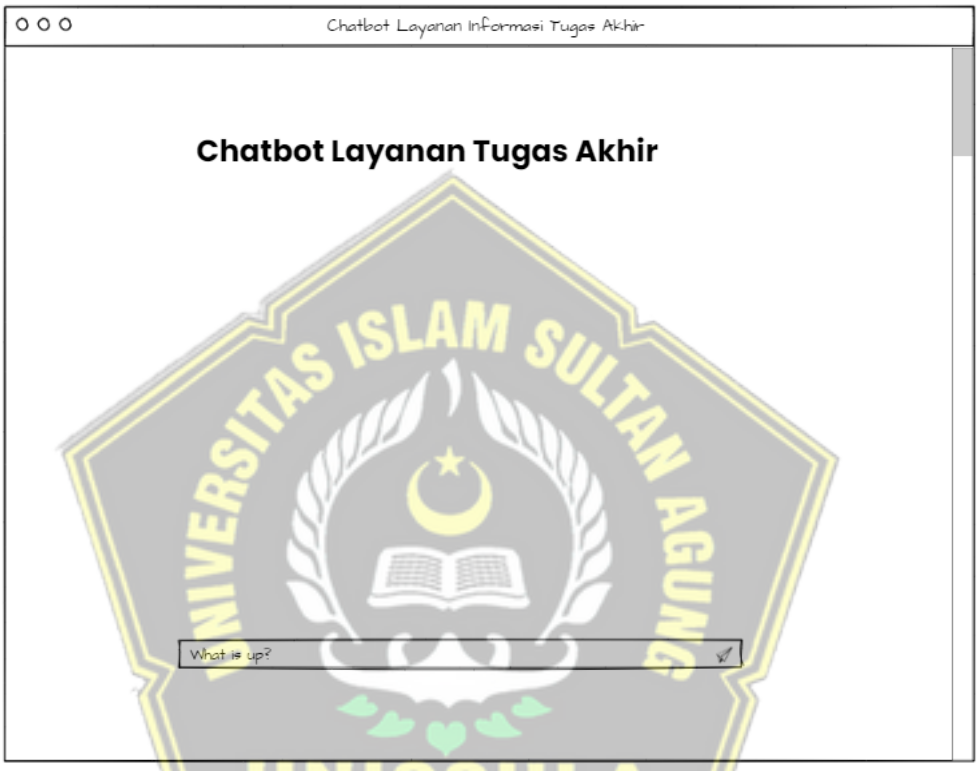
10. *Streamlit*

    Streamlit adalah sebuah *framework open-source* yang memungkinkan Anda membuat antarmuka pengguna (UI) interaktif dalam *Python*. Dalam penelitian ini menggunakan streamlit agar dapat mengubah skrip *Python* menjadi aplikasi web interaktif dengan cepat, tanpa memerlukan pengetahuan mendalam tentang pengembangan web atau UI sehingga bisa menghemat waktu.

3.4 Perancagan *User Interface*

3.4.1 Halaman Awal *Chatbot*

Seperti yang ditunjukkan pada Gambar 3.3, halaman Percakapan *chatbot* adalah halaman di mana sistem *chatbot* menanggapi pertanyaan pengguna dan menampilkan jawaban yang diberikan oleh *chatbot* tersebut.



Gambar 3. 4 Halaman Awal *Chatbot*

Gambar 3.3 menunjukkan halaman awal di mana pengguna melihat Judul. Pada halaman ini, mereka juga dapat melihat bagian input teks yang digunakan untuk memberikan pertanyaan dan icon pesawat ketas berfungsi sebagai tombol untuk mengirim pertanyaan berikutnya.

3.4.2 Halaman Percakapan *Chatbot*

Seperti yang ditunjukkan pada Gambar 3.3, halaman Percakapan *chatbot* adalah halaman di mana sistem *chatbot* menjawab pertanyaan pengguna dan menampilkan hasilnya.

Gambar 3. 5 Halaman *Chatting*

Gambar 3.3 menunjukkan halaman yang menampilkan isi percakapan antara pengguna dan bot. Pada tahap ini, sistem memproses pertanyaan yang dimasukkan oleh pengguna dan memproses jawabannya, yang kemudian ditampilkan di bagian *body* halaman *chatbot*. Penelitian ini akan mendemokan halaman web interaktif yang sudah dibuat kepada pengguna untuk melihat apakah sudah sesuai dengan pengguna.

BAB IV
HASIL DAN ANALISIS PENELITIAN

4.1 Hasil Penelitian

Data yang merupakan jawaban didapatkan dari panduan Tugas Akhir Teknik Informatika UNISSULA yang berbentuk *file pdf* diambil dari *link URL* https://linktr.ee/tugasakhir.

Setelah mendapatkan data dari panduan Tugas Akhir, selanjutnya membuat pertanyaan dan jawaban yang berkaitan dengan panduan Tugas Akhir seperti tahapan dan persyaratan mengambil matakuliah Tugas Akhir, pelaksanaan dan penyunan proposal Tugas Akhir dan laporan Tugas Akhir, proses bimbingan Tugas Akhir, pendaftaran, pelaksanaan, dan juga penilaian sidang Tugas Akhir.

Setelah mendapatkan data dan memahami data panduan Tugas Akhir serta membuat pertanyaan dan jawaban, langkah selanjutnya adalah tahap persiapan data. Data pertanyaan dan jawaban yang sudah dibuat berdasarkan panduan Tugas Akhir kemudian dijadikan *file* JSON yang terdiri dari *intent, tag, patterns,* dan *response.* Jumlah data yang dibuat sebanyak 1021 data yang terdiri dari data *tag* sebagai label, *patterns,* sebagai pertanyaan dan *response* sebagai jawaban. Kemudian dataset berformat *json* dikonversi menjadi *dataframe* seperti pada gambar 4.1.

4.1.1 Persiapan Data

4.1.1.1 Konversi ke dataframe

```
def create_df():
    df = pd.DataFrame({
        'Patterns' : [],
        'Tag' : []
    })
    return df

df = create_df()
df
```

1 to 25 of 3063 entries Filter

index	Patterns	Tag
0	halo	menyapa
1	hei	menyapa
2	hai	menyapa
3	hai, apakah ini layanan informasi tugas akhir	menyapa
4	hei, apakah ini layanan informasi tugas akhir	menyapa
5	halo, apakah ini layanan informasi tugas akhir	menyapa
6	Terima kasih	berakhir
7	Terima kasih infonya	berakhir
8	bye	berakhir
9	bye bro	berakhir
10	oke bye	berakhir
11	selamat tinggal	berakhir
12	sampai jumpa lagi	berakhir
13	oke aku akan menemuimu nanti	berakhir

Gambar 4. 1 Konversi menjadi *dataframe*

Pada gambar 4.1 mengkonversi dataset berformat JSON menjadi *dataframe* kemudian menampilkan dalam bentuk tabel *dataframe* yang berisi list pertanyaan. Langkah berikutnya memeriksa informasi dari dataset dan melihat bentuk dari *dataframe*.

```
[ ]  def print_dfInfo(df, ds_name="df"):
         print(f"The info of {ds_name} dataset\n")
         print(df.info())

     print_dfInfo(df, "Chatbot")
```

```
The info of Chatbot dataset

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1021 entries, 0 to 1020
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Patterns  1021 non-null   object
 1   Tag       1021 non-null   object
dtypes: object(2)
memory usage: 23.9+ KB
None
```

```
[ ]  def num_classes(df, target_col, ds_name="df"):
         print(f"The {ds_name} dataset has {len(df[target_col].unique())} classes")

     num_classes(df, 'Tag', "Chatbot")
```

```
The Chatbot dataset has 29 classes
```

```
[ ]  def check_null(df, ds_name='df'):
         print(f"Null Values in each col in the {ds_name} dataset:\n")
         print(df.isnull().sum())

     check_null(df, "Chatbot")
```

```
Null Values in each col in the Chatbot dataset:

Patterns    0
Tag         0
dtype: int64
```

Gambar 4. 2 Menampilkan informasi dari dataset

Pada gambar 4.2 menampilkan informasi dari dataset termasuk tipe data, nilai yang tidak null, dan jumlah kelas. Pada hasil tangkapan menunjukan jumlah data sebanyak 1020, jumlah nilai yang tidak null ada 0, dan kelas berjumlah 29 kelas. Kemudian memeriksa jumlah baris dan kolom pada dataset seperti pada gambar 4.3

```
def print_shape_df(df, ds_name="df"):
    print(f"{ds_name} dataset has {df.shape[0]} rows and {df.shape[1]} columns")

print_shape_df(df, "Chatbot")
```

```
Chatbot dataset has 1021 rows and 2 columns
```

Gambar 4. 3 cek baris dan kolom dataset

Setelah mengecek jumlah baris dan kolom yang ada pada dataset menggunakan menggunakan metode *f-string. F-string* adalah fitur dalam Python yang memungkinkan penanaman ekspresi Python langsung dalam string dengan memasukkannya dalam tanda kurung kurawal {}. Selanjutnya memasuki tahap pemrosesan data, membersihkan dari karakter khusus dan mengonversi semua teks ke huruf kecil

4.1.2 *Data Preprocessing*

4.1.2.1 *Data Cleaning*

```
df['Patterns'] = df['Patterns'].str.replace('[^a-zA-Z0-9\s]', '', regex=True)
df['Patterns'] = df['Patterns'].str.lower()
df
```

	Patterns	Tag
0	halo	menyapa
1	hei	menyapa
2	hai	menyapa
3	hai apakah ini layanan informasi tugas akhir	menyapa
4	hei apakah ini layanan informasi tugas akhir	menyapa
...	...	...
1016	apakah penting memiliki dosen pembimbing	bimbingan
1017	apa ketentuan bimbingan ta	bimbingan
1018	apakah ada ketentuan dalam bimbingan ta	bimbingan
1019	apa maksud dan tujuan dari bimbingan ta	bimbingan
1020	siapa yang mengarahkan mahasiswa dalam bimbing...	bimbingan

1021 rows × 2 columns

Gambar 4. 4 Proses membersihkan data

Pada gambar 4.2 Data dibersihkan menggunakan fungsi `str.replace('[^a-zA-Z0-9\s]','',regex=True)` digunakan untuk menghapus dari karakter khusus kecuali huruf dan angka, penghapusan tanda baca dan

karakter khusus dilakukan pada kolom *patterns* didalam *dataframe* dan *str.lower* untuk mengonversi semua teks ke huruf kecil kedua fungsi ini masuk kedalam *library* standar *Python*. Pembersihan ini dilakukan agar mempermudah proses pelatihan dan meningkatkan kinerja model. kemudian menampilkan hasil dari pembersihan data dalam bentuk tabel *dataframe*. Proses selanjutnya mengambil daftar unik dari kategori dalam kolom 'Tag' dari *dataframe*.

4.1.2.2 Memberi Label pada Data

```
labels = df['Tag'].unique().tolist()
labels = [s.strip() for s in labels]
```

Gambar 4. 5 Mengambil daftar unik pada dalam kolom 'Tag'

Pada gambar 4.5 proses mengambil daftar unik dari kolom 'Tag' dalam *dataframe* menggunakan beberapa metode seperti `df2['Tag'].unique()` ini bertujuan untuk mengetahui kategori apa saja yang ada dalam data. Penggunaan `.tolist()` berguna untuk Mengonversi hasil nilai unik yang diperoleh menjadi daftar Python. Metode `tolist()` digunakan untuk mengonversi *array* atau objek *pandas* menjadi daftar. Ini membuat daftar baru bernama *labels* yang berisi nilai unik dari kolom 'Tag', di mana setiap elemen dalam daftar dihapus spasi di awal dan akhirnya. Ini dilakukan dengan menggunakan list comprehension dan metode `strip()` yang menghilangkan spasi di awal dan akhir setiap string. selanjutnya menampilkan hasil pengambilan daftar unik dalam *dataframe*. label kategori tersebut akan digunakan sebagai target dalam pelatihan model. Langkah selanjutnya membuat kamus yang membentuk pemetaan antara ID dan label kategori.

```
num_labels = len(labels)
id2label = {id:label for id, label in enumerate(labels)}
label2id = {label:id for id, label in enumerate(labels)}
```

Gambar 4. 6 Membentuk pemetaan antara ID dan label kategori.

Pada gambar 4.6 menunjukan proses membentuk pemetaan ID dan label Kategori. `num_labels = len(labels)` digunakan untuk menghitung jumlah label yang ada dalam daftar labels `num_labels` sekarang menyimpan informasi tentang berapa banyak kategori yang mungkin ada dalam data. Kode dibawahnya

berguna untuk membuat 2 kamus yang mengaitkan ID dengan label kategori, fungsi `enumerate(labels)` digunakan untuk mengambil setiap elemen dalam daftar *labels* bersama dengan indeksnya. Kamus `id2label` dan `label2id` kemudian dibuat dengan indeks sebagai kunci dan label sebagai nilai. Pemetaan ini membantu mengelola label dan ID dengan cara yang konsisten dan nyaman dalam pengolahan data dan hasil model. Kemudian menambahkan kolom labels pada dataframe seperti pada gambar 4.7.

```
df['labels'] = df['Tag'].map(lambda x: label2id[x.strip()])
df
```

	Patterns	Tag	labels
0	halo	menyapa	0
1	hei	menyapa	0
2	hai	menyapa	0
3	hai apakah ini layanan informasi tugas akhir	menyapa	0
4	hei apakah ini layanan informasi tugas akhir	menyapa	0
...	...	...	...
1016	apakah penting memiliki dosen pembimbing	bimbingan	28
1017	apa ketentuan bimbingan ta	bimbingan	28
1018	apakah ada ketentuan dalam bimbingan ta	bimbingan	28
1019	apa maksud dan tujuan dari bimbingan ta	bimbingan	28
1020	siapa yang mengarahkan mahasiswa dalam bimbing...	bimbingan	28

1021 rows × 3 columns

Gambar 4. 7 Membuat kolom ‘labels’ dalam *dataframe*

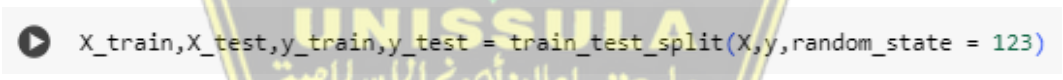
Pada gambar 4.7 menambahkan kolom ‘labels’ ke *dataframe*. Dalam kode `df['Tag'].map(lambda x: label2id[x.strip()])` digunakan untuk mengambil nilai dari kolom ‘Tag’ untuk setiap baris dalam *dataframe* dan memetakannya ke ID kelas menggunakan kamus `label2id`. `lambda x: label2id[x.strip()]` adalah fungsi lambda yang digunakan untuk mengaplikasikan pemetaan label kategori ke ID kelas. `x.strip()` digunakan untuk membersihkan spasi di awal dan akhir label sebelum mencocokkannya dengan kamus. Hasil dari pemetaan ini adalah ID kelas yang sesuai dengan setiap nilai dalam kolom `'Tag'`. `df['labels']` adalah langkah selanjutnya untuk

menetapkan hasil pemetaan ke dalam kolom baru ‘labels’ dalam *dataframe*. Kemudian hasilnya ditampilkan dalam bentuk tabel *dataframe.* Dengan cara ini dapat memudahkan menggunakan ID kelas ini sebagai target dalam pelatihan model. Tahap berikutnya mengambil nilai dari dua kolom dan menyimpannya ke dalam variabel ‘X’ dan ‘y’ kemudian membagi data untuk data latih dan data uji.

4.1.3 Membagi Data menjadi Data Latih dan Data Uji

Gambar 4. 8 Mengambil nilai dari kolom ‘Patterns’ dan ‘labels’

Pada gambar 4.8 merupakan proses pengambilan nilai dari kolom ‘Patterns’ dan ‘labels’ dan menyimpannya dalam variabel X dan y. `X = list(df['Patterns'])` ini menngambil nilai dari kolom ‘Patterns’ dalam dataframe dan menyimpannya dalam daftar ‘X’, kolom ‘Patterns’ berisi teks yang akan digunakan sebagai input untuk model. `y = list(df['labels'])` ini mengambil nilai dari kolom ‘labels’ dalam dataframe dan menyimpannya dalam daftar ‘y’, kolom ‘labels’ berisi label atau kategori yang merupakan target untuk output yang diinginkan dari model. Langkah selanjutnya membagi data untuk data latih dan data uji seperti pada gambar 4.9.



Gambar 4. 9 Membagi data untuk data latih dan data uji

Terlihat pada gambar 4.9 merupakan kode untuk membagi data untuk keperluan data latih dan data uji menggunakan fungsi `train_test_split` dari *library scikit-learn* dimana ‘X’ dan ‘y’ adalah input dan output yang akan dibagi. `random_state=123` berfungsi untuk memberikan keberlanjutan untuk pengulangan pengujian, sehingga setiap kali menjalankan kode, hasil pembagian akan konsisten. Variabel `X_train` dan `y_train` berisikan data pelatihan untuk melatih model, sedangkan `X_test` dan `y_test` akan berisi data pengujian untuk menguji kinerja model yang telah dilatih. Default pembagian adalah 75% untuk data pelatihan dan 25% untuk pengujian. Tahap selanjutnya mengimport model

yang akan digunakan, dimana dalam penelitian ini menggunakan model BERT versi indonesia yaitu indoBERT dari indolem.

4.1.4 Memuat Model BERT dan Tokenizer

```
model_name = "indolem/indobert-base-uncased"
max_len = 256

tokenizer = BertTokenizer.from_pretrained(model_name,
                                          max_length=max_len)

model = BertForSequenceClassification.from_pretrained(model_name,
                                                      num_labels=num_labels,
                                                      id2label=id2label,
                                                      label2id = label2id)
```

Gambar 4. 10 Mengimport model indoBERT

Pada gambar 4.10 mengimport model BERT yang dipilih yaitu "indolem/indobert-base-uncased" yang merupakan versi BERT yang telah di-pretrained dalam bahasa indonesia. `max_len=25` adalah panjang maksimum token yang akan diterima oleh model. `Tokenizer = BertTokenizer.from_pretrained (model_name, max_length=max_len)` kode ini membuat instance dari BertTokenizer menggunakan model yang telah di-pretrained (`model_name`) dan panjang maksimum token (`max_len`). Tokenizer ini akan digunakan untuk mengonversi teks menjadi token yang dapat dimengerti oleh model BERT. Pada kode `model = BertForSequenceClassification.from_pretrained(model_name,num_labels=num_labels, id2label=id2label, label2id=label2id)` membuat instance dari *BertForSequenceClassification* yang merupakan model BERT yang telah di-pretrained untuk tugas klasifikasi urutan. Parameter `from_pretrained` digunakan untuk memuat bobot dan konfigurasi model dari model yang telah di-*pretrained* (`model_name`). `num_labels` adalah jumlah label atau kategori yang akan diprediksi oleh model. id2label dan label2id merupakan kamus yang berisi pemetaan anatara ID kelas dan label kategori yang telah dibuat sebelumnya. Ini berguna untuk memahami represenasi label kategori selama pelatihan dan evaluasi model.

Pemilihan model BERT untuk klasifikasi teks dalam penelitian ini yang membuat sistem *chatbot* memiliki beberapa pertimbangan karena model klasifikasi dapat membantu chatbot untuk memahami konteks dan niat pengguna lebih baik, sehingga memungkinkan chatbot memberikan respons yang lebih relevan dan kontekstual. Tahap selanjutnya mengonversi data latih dan data uji menjadi representasi yang dapat dimengerti oleh model.

4.1.5 Transformasi Data dan Data Loader

4.1.5.1 Konversi Teks dan Menyamakan Panjang Teks

```
train_encoding = tokenizer(X_train, truncation=True, padding=True)
test_encoding = tokenizer(X_test, truncation=True, padding=True)
full_data = tokenizer(X, truncation=True, padding=True)
```

Gambar 4. 11 Mengonversi teks dan menyamakan panjang teks

Pada gambar 4.11 terlihat potongan kode yang menggunakan tokenizer BERT untuk mengonversi teks dari data latih (`X_train`), data uji (`X_test`), dan daftar 'X' yang berisikan nilai input menjadi representasi yang dapat dimengerti oleh model yang terdiri dari token ID yang merupakan representasi numerik dari setiap kata atau sub-phrase dalam teks, mask yang berupa array biner digunakan untuk menunjukkan bagian teks yang asli(*non-padded*) memiliki nilai '0' dan bagian yang hanya hasil dari padding memiliki nilai '0', dan segmen yang digunakan untuk membedakan bagian-bagian teks yang berasal dari kalimat atau bagian teks yang berbeda. `truncation=True` digunakan untuk memotong atau membatasi panjang teks yang melebihi panjang maksimum yang ditentukan tokenizer. `padding=True` menambahkan token padding ke teks sehingga semua teks memiliki panjang yang sama, ini diperlukan untuk pemrosesan *batch* pada model.

Dengan menerapkan token ID, mask, dan segmen ini, model dapat memproses teks dengan cara yang memungkinkan pemahaman kontekstual, representasi yang kaya, dan model BERT dapat memahami hubungan dan makna yang kompleks dalam teks, dan hasil dari representasi ini dapat digunakan sebagai input untuk model BERT pada tahap pelatihan atau evaluasi model. langkah selanjutnya mendefinisikan dataloader yang digunakan untuk mempersiapkan data untuk pelatihan dan pengujian model.

4.1.5.2 Data Loader

```
class DataLoader(Dataset):

    def __init__(self, encodings, labels):

        self.encodings = encodings
        self.labels = labels

    def __getitem__(self, idx):

        item = {key: torch.tensor(val[idx]) for key, val in self.encodings.items()}
        item['labels'] = torch.tensor(self.labels[idx])
        return item

    def __len__(self):

        return len(self.labels)
```

Gambar 4. 12 Mempersiapkan data untuk pelatihan dan pengujian model

Pada gambar 4.12 menampilkan kode yang mendefinisikan kelas `DataLoader` yang merupakan turunan dari kelas dataset yang berguna untuk mempersiapkan data untuk pelatihan dan pengujian model.

Pada potongan kode `__init__(self, encodings, labels)` ini menggunakan metode *constructor* yang digunakan untuk menginisialisasi objek dataloader. `self` digunakan untuk merujuk pada instance objek yang baru dibuat, `encodings` merupakan hasil representasi token dari teks yang sudah dikonversi sebelumnya, `labels` merupakan target yang sesuai dengan teks pada setiap indeks. `__getitem__(self, idx)` potongan kode ini digunakan untuk mengambil item dari dataset pada indeks tertentu, `idx` merupakan indeks item yang ingin diambil. kemudian membuat dictionary item yang berisi representasi *token encodings* dan labels untuk item pada indeks `idx` , lalu mengonversi nilai dalam *dictionary* ke tensor sebelum mengembalikan item. `__len__(self)`

metode ini digunakan untuk mengembalikan panjang total dataset dan digunakan ketika memanggil `len()` pada objek dataset.

Dengan menggunakan kelas DataLoader ini bisa untuk menggabungkan representasi token dan label ke dalam objek dataset yang dapat digunakan untuk pelatihan atau pengujian pada model. Langkah selanjutnya membuat 3 variabel yang menyimpan *instance* dari kelas DataLoader seperti pada gambar 4.13.

4.1.5.3 Membuat Tiga Variabel dari kelas DataLoader

```
train_dataloader = DataLoader(train_encoding, y_train)
test_dataloader = DataLoader(test_encoding, y_test)
fullDataLoader = DataLoader(full_data, y_test)
```

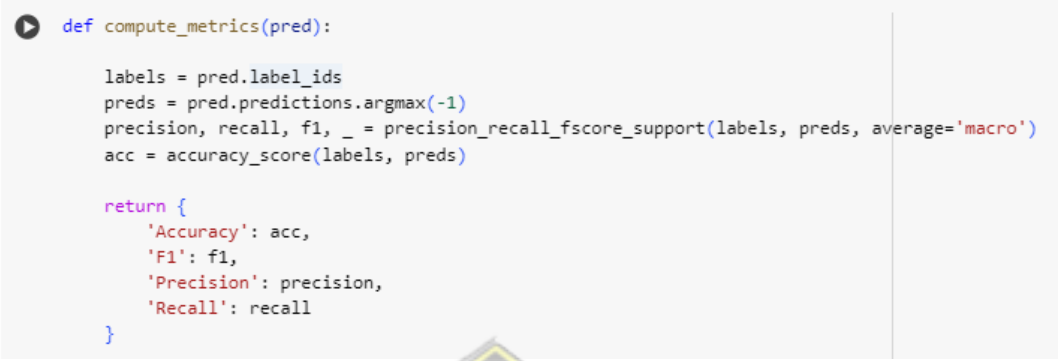
Gambar 4. 13 Membuat tiga variabel dari kelas DataLoader

Pada gambar 4.13 terlihat potongan kode yang berfungsi untuk membuat 3 variabel yaitu `train_dataloader`, `test_dataloader` dan `fullDataLoader` yang menyimpan *instance* dari kelas `DataLoader`. `train_encoding` adalah hasil representasi token dari teks dari data latih menggunakan tokenizer sedangkan `test_encoding` dari data uji. `y_train` merupakan label atau target yang sesuai dengan teks pada setiap indeks dalam data latih sedangkan `y_test` dari dalam data uji. `full_data` adalah representasi token dari semua data latih yang dihasilkan menggunakan tokenizer dan sudah disamakan panjangnya.

Setelah membuat 3 variabel ini selanjutnya akan digunakan untuk melatih dan menguji model sebagai input untuk fungsi pelatihan dan pengujian, yang mengambil sampel dalam bentuk batch selama proses pelatihan dan evaluasi model. Tahap selanjutnya membuat fungsi untuk menghitung sejumlah metrik evaluasi dari prediksi model.

4.1.6 Mendefinisikan Metrik Evaluasi dan Training Argument

4.1.6.1 Mendefinisikan Metrik Evaluasi



```
def compute_metrics(pred):

    labels = pred.label_ids
    preds = pred.predictions.argmax(-1)
    precision, recall, f1, _ = precision_recall_fscore_support(labels, preds, average='macro')
    acc = accuracy_score(labels, preds)

    return {
        'Accuracy': acc,
        'F1': f1,
        'Precision': precision,
        'Recall': recall
    }
```

Gambar 4. 14 Membuat fungsi untuk mengevaluasi kinerja model

Pada gambar 4.14 menampilkan sebuah fungsi yang memiliki satu parameter yaitu pred yang berguna untuk menghitung metrik evaluasi dari prediksi model serta memantau performa model selama pelatihan. `labels = pred.label_ids` membuat variabel `labels` yang mengambil label aktual dari prediksi. `preds = pred.predictions.argmax(-1)` berfungsi untuk memprediksi kelas dengan probabilitas tertinggi dari objek prediksi `pred`, `argmax(-1)` digunakan untuk mendapatkan indeks kelas dengan probabilitas tertinggi. Menghitung metrik *precision, recall,* dan *F1-score* menggunakan fungsi `precision_recall_fscore_support` dari *library scikit-learn* , `average='macro'` digunakan untuk menghitung metrik secara rata-rata untuk semua kelas. `acc = accuracy_score(labels, preds)` berfungsi untuk menghitung akurasi dengan membandingkan label aktual dengan prediksi. fungsi `return` berfungsi untuk mengembalikan hasil metrik dalam bentuk *dictionary* yang mencakup akurasi yang mengukur sejauh mana prediksi model cocok dengan label, *F1-score*, *precision*, dan *recall* memberikan informasi tentang performa model pada masing-masing kelas. Langkah selanjutnya mengkonfigurasikan pengaturan untuk proses pelatihan dan evaluasi model.

4.1.6.2 Mendefinisikan Training Argument

```
training_args = TrainingArguments(
    output_dir='/content/drive/MyDrive/Chatbot/output',
    do_train=True,
    do_eval=True,
    num_train_epochs=75,
    per_device_train_batch_size=32,
    per_device_eval_batch_size=16,
    warmup_steps=100,
    weight_decay=0.05,
    logging_strategy='steps',
    logging_dir='/content/drive/MyDrive/Chatbot/multi-class-logs',
    logging_steps=50,
    evaluation_strategy="steps",
    eval_steps=50,
    save_strategy="steps",
    load_best_model_at_end=True
)
```

Gambar 4. 15 Mengkonfigurasikan *hyperparameter* untuk proses pelatihan dan evaluasi model

Terlihat pada gambar 4.15 menampilkan objek *TrainingArguments* yang berisikan *hyperparameter. Hyperparameter* adalah parameter yang tidak diatur selama pelatihan model, tetapi harus ditentukan sebelum memulai proses pelatihan. Terlihat `output_dir` dan `logging_dir` menunjukkan direktori tempat model, hasil pelatihan, dan log pelatihan akan disimpan. `do_train` dan `do_eval` menentukan apakah proses pelatihan dan evaluasi akan dilakukan (`True`) atau tidak (`False`) dalam penelitian ini memilih dilakukan. `num_train_epochs=75` parameter untuk menentukan jumlah *epoch* atau iterasi penelitian ini menentukan 75 *epoch*. `per_device_train_batch_size` dan `per_device_eval_batch_size` parameter ini menentukan ukuran batch yang diproses pada setiap GPU dalam tiap langkah pelatihan atau evaluasi, penelitian ini menentukan ukuran batch 32 untuk pelatihan dan 16 untuk evaluasi. `warmup_steps=100` parameter ini untuk menentukan jumlah 100 langkah pada fase pemanasan yang digunakan untuk menyesuaikan pembelajaran. `weight_decay=0.05` parameter ini menentukan nilai koefisien *weight decay* yang digunakan selama pelatihan untuk mencegah *overfitting*. `logging_strategy='steps'` dan `logging_steps=50` berguna untuk melakukan logging setiap 50 langkah selama pelatihan. `evaluation_strategy="steps"` dan `eval_steps=50` berguna untuk melakukan

evaluasi setiap 50 langkah selama pengujian. `save_strategy="steps"` ini berguna untuk menyimpan model setiap 50 langkah karena secara implisit diatur oleh parameter `logging_steps`. `load_best_model_at_end=True` ini berguna untuk meload model terbaik pada akhir pelatihan. Langkah selanjutnya membuat objek *Trainer* yang berfungsi untuk menyediakan model dan konfigurasi yang sudah dibuat sebelumnya.

4.1.7 Pelatihan Model

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataloader,
    eval_dataset=test_dataloader,
    compute_metrics= compute_metrics
)
```

Gambar 4. 16 Membuat objek *Trainer*

Pada gambar 4.16 membuat objek *Trainer* menggunakan *library Transformers* dengan parameter yang sudah didefinisikan sebelumnya seperti `model=model` , `args=training_args` parameter ini adalah objek *TrainingArguments* yang berisi konfigurasi untuk proses pelatihan, `train_dataset=train_dataloader` parameter ini adalah dataset pelatihan yang akan digunakan selama pelatihan, `eval_dataset=test_dataloader` parameter ini adalah dataset evaluasi yang akan digunakan selama evaluasi model pada set pengujian. `compute_metrics=compute_metrics` parameter ini adalah fungsi yang digunakan untuk menghitung metrik evaluasi pada set pengujian. Objek *Trainer* ini dapat digunakan untuk melatih dan mengevaluasi model dengan memanggil metode `tariner.train()` seperti pada gambar 4.17.

4.1.7.1 Menjalankan Pelatihan

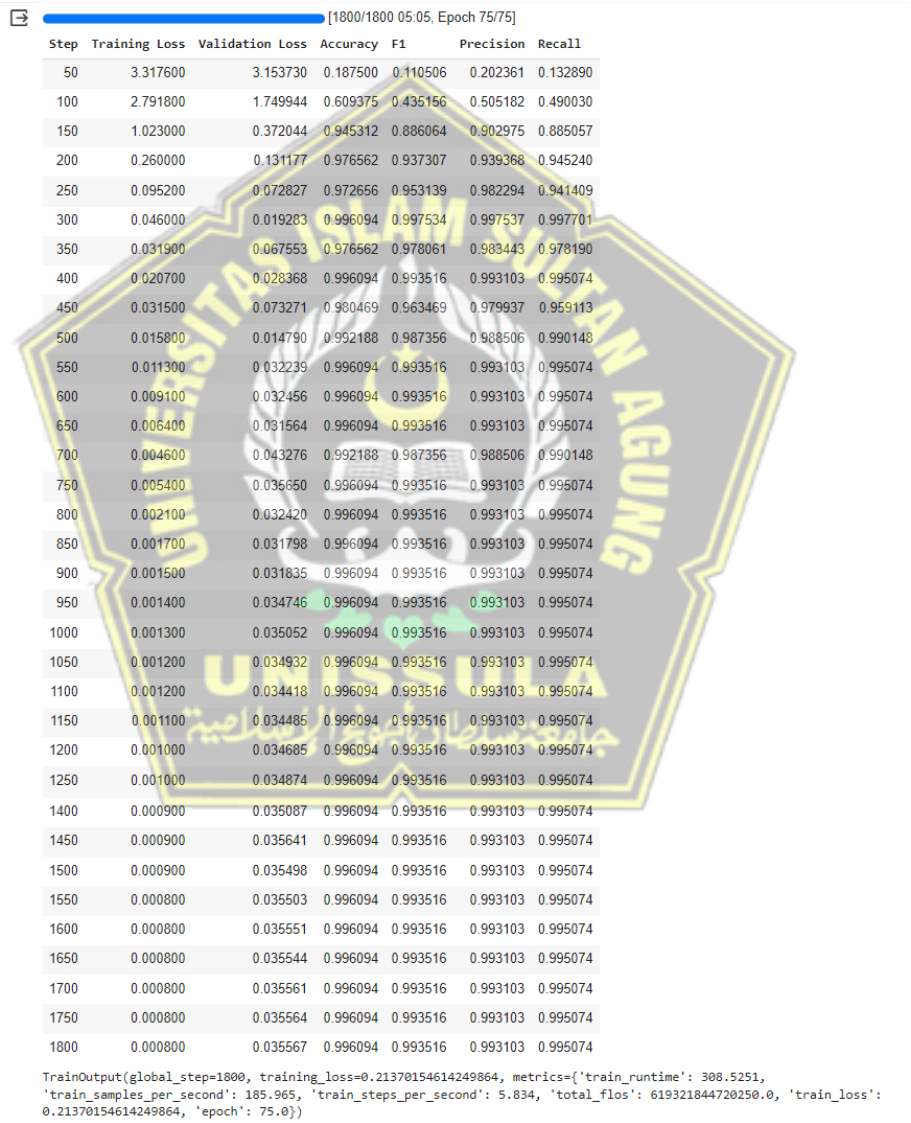
```
trainer.train()
```

Gambar 4. 17 Menjalankan fungsi train untuk memulai proses pelatihan

Pada gambar 4.17 memanggil metode `trainer.train()` untuk memulai proses pelatihan model, pada proses ini menggunakan data latih dan data uji yang sudah disiapkan sebelumnya serta melatih model selama 75 *epoch* sesuai dengan

jumlah yang telah ditentukan dalam konfigurasi pelatihan, *warm-up learning rate* akan diterapkan pada awal pelatihan, penyimpanan model dan evaluasi pada set pengujian akan dilakukan sesuai sesuai dengan strategi yang sudah ditentukan pada konfigurasi pelatihan. Selama pelatihan metrik-metrik performa seperti *loss, accuary, F1-score, precision,* dan *recall* dapat dilihat pada log pelatihan, seperti yang terlihat pada gambar 4.18.

4.1.7.2 Menampilkan Log Pelatihan


[1800/1800 05:05, Epoch 75/75]

Step	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
50	3.317600	3.153730	0.187500	0.110506	0.202361	0.132890
100	2.791800	1.749944	0.609375	0.435156	0.505182	0.490030
150	1.023000	0.372044	0.945312	0.886064	0.902975	0.885057
200	0.260000	0.131177	0.976562	0.937307	0.939368	0.945240
250	0.095200	0.072827	0.972656	0.953139	0.982294	0.941409
300	0.046000	0.019283	0.996094	0.997534	0.997537	0.997701
350	0.031900	0.067553	0.976562	0.978061	0.983443	0.978190
400	0.020700	0.028368	0.996094	0.993516	0.993103	0.995074
450	0.031500	0.073271	0.980469	0.963469	0.979937	0.959113
500	0.015800	0.014790	0.992188	0.987356	0.988506	0.990148
550	0.011300	0.032239	0.996094	0.993516	0.993103	0.995074
600	0.009100	0.032456	0.996094	0.993516	0.993103	0.995074
650	0.006400	0.031564	0.996094	0.993516	0.993103	0.995074
700	0.004600	0.043276	0.992188	0.987356	0.988506	0.990148
750	0.005400	0.036650	0.996094	0.993516	0.993103	0.995074
800	0.002100	0.032420	0.996094	0.993516	0.993103	0.995074
850	0.001700	0.031798	0.996094	0.993516	0.993103	0.995074
900	0.001500	0.031835	0.996094	0.993516	0.993103	0.995074
950	0.001400	0.034746	0.996094	0.993516	0.993103	0.995074
1000	0.001300	0.035052	0.996094	0.993516	0.993103	0.995074
1050	0.001200	0.034932	0.996094	0.993516	0.993103	0.995074
1100	0.001200	0.034418	0.996094	0.993516	0.993103	0.995074
1150	0.001100	0.034485	0.996094	0.993516	0.993103	0.995074
1200	0.001000	0.034685	0.996094	0.993516	0.993103	0.995074
1250	0.001000	0.034874	0.996094	0.993516	0.993103	0.995074
1400	0.000900	0.035087	0.996094	0.993516	0.993103	0.995074
1450	0.000900	0.035641	0.996094	0.993516	0.993103	0.995074
1500	0.000900	0.035498	0.996094	0.993516	0.993103	0.995074
1550	0.000800	0.035503	0.996094	0.993516	0.993103	0.995074
1600	0.000800	0.035551	0.996094	0.993516	0.993103	0.995074
1650	0.000800	0.035544	0.996094	0.993516	0.993103	0.995074
1700	0.000800	0.035561	0.996094	0.993516	0.993103	0.995074
1750	0.000800	0.035564	0.996094	0.993516	0.993103	0.995074
1800	0.000800	0.035567	0.996094	0.993516	0.993103	0.995074

TrainOutput(global_step=1800, training_loss=0.21370154614249864, metrics={'train_runtime': 308.5251, 'train_samples_per_second': 185.965, 'train_steps_per_second': 5.834, 'total_flos': 619321844720250.0, 'train_loss': 0.21370154614249864, 'epoch': 75.0})

Gambar 4. 18 Proses pelatihan model

Pada gambar 4.18 adalah tampilan selama proses pelatihan yang berlangsung selama 75 *epoch* dan pada setiap 50 *step* akan mencatat metrik

evaluasi seperti *training loss, validation loss, accuracy, F1-score, precision,* dan *recall*. *Output* ini memberikan gambaran keseluruhan tentang kinerja model pada tahap pelatihan terakhir, termasuk durasi pelatihan, kecepatan pemrosesan sampel, jumlah operasi mengambang, dan nilai *loss.* Kemudian langkah selanjutnya mengevaluasi hasil kinerja model pada data latih dan data uji.

4.1.8 Evaluasi Model

```
q=[trainer.evaluate(eval_dataset=df) for df in [train_dataloader, test_dataloader]]

pd.DataFrame(q, index=["train","test"]).iloc[:,:5]
```

[48/48 00:01]

	eval_loss	eval_Accuracy	eval_F1	eval_Precision	eval_Recall
train	0.004568	0.998693	0.998967	0.998922	0.999042
test	0.016182	0.992188	0.993189	0.994013	0.993034

Gambar 4. 19 Mengevaluasi model pada data latih dan data uji

Terlihat pada gambar 4.19 menggunakan objek *trainer* untuk mengevalusi model pada dua dataset yang berbeda yaitu `train_dataloader` (data latih) dan `test_dataloader` (data uji). fungsi `trainer.evaluate` digunakan untuk mengevaluasi model pada data latih dan data uji. Kemudian hasil evaluasi tersebut dimasukkan ke dalam sebuah list `q`. Setelah itu membuat DataFrame dari list `q`. Hasil akhirnya adalah DataFrame yang berisi hasil evaluasi model pada dataset pelatihan dan pengujian, dengan label "train" dan "test". DataFrame ini memberikan gambaran tentang seberapa baik model berperforma pada kedua dataset.

4.2 Hasil Evaluasi

Tabel 4. 1 Hasil evaluasi kinerja model

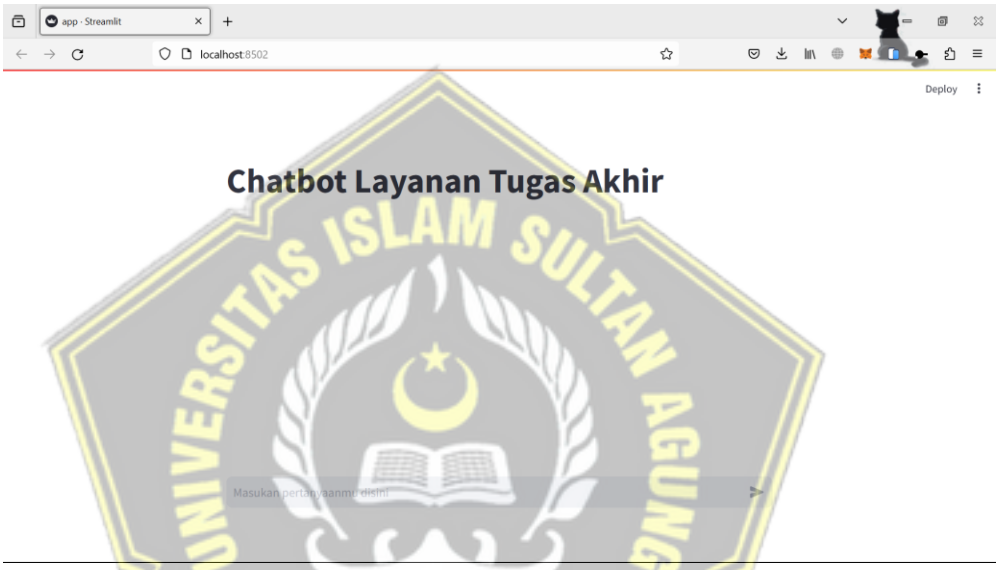
	Loss	Accuracy	F1	Precision	Recall
Data Latih	0.004568	99,87%	99,99%	99,99%	99,9%
Data Uji	0.016182	99,22%	99,32%	99,4%	99,3%

Pada tabel 4.1 menampilkan hasil evaluasi kinerja model pada dua data yaitu data latih dan data uji. Dapat disimpulkan dari hasil yang ditampilkan menunjukkan bahwa model memiliki kinerja yang sangat baik pada kedua data, dengan akurasi tinggi dan tingkat loss yang rendah menandakan bahwa model dapat memprediksi label dengan sangat baik, nilai presisi yang tinggi menunjukan bahwa model memiliki kemampuan yang baik untuk mengidentifikasi data positif dengan benar. Selain itu *F1-score* yang tinggi menandakan performa yang baik pada keseluruhan dari performa model.

4.3 Hasil Perancangan *User Interface*

Setelah melalui tahap pemodelan sistem, Selanjutnya diimplementasikan pada *platform website*. Hasil implementasi antarmuka ini merujuk pada desain *wireframe* pada BAB 3 yang dikembangkan untuk *platform website.* Berikut ini merupakan hasil tangkapan layar dari demo *Chatbot* yang dilakukan.

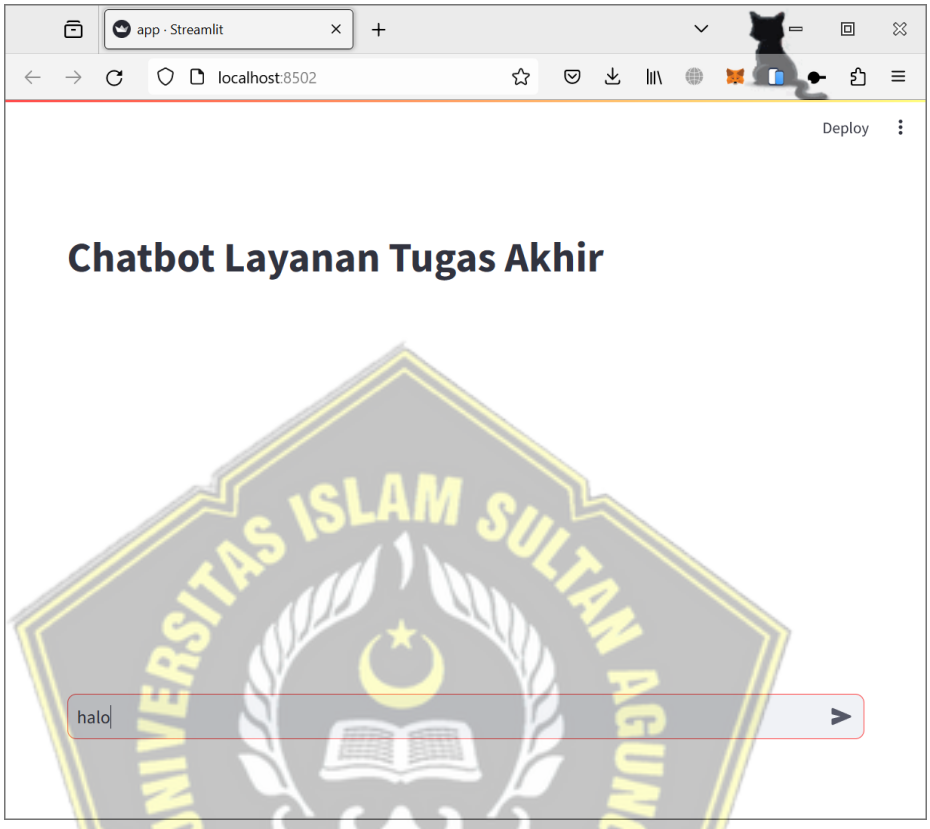
1. Halaman Awal *Chatbot*



Gambar 4. 20 Tampilan Halaman Awal *Chatbot*

Pada gambar 4.2 merupakan Implementasi desain Halaman Awal *Chatbot* yang dilihat oleh pengguna pertama kali, yang menampilkan judul aplikasi yang diletakkan pada bagian *appbar* menggunakan fungsi *st.title.* Pada bagian *body* terdapat halaman percakapan yang nanti berisikan riwayat percakapan, kemudian pada bagian bawah terdapat *input text* yang berguna untuk memberikan pertanyaan dan terdapat *icon* pesawat kertas sebagai tombol untuk mengirim pertanyaan.

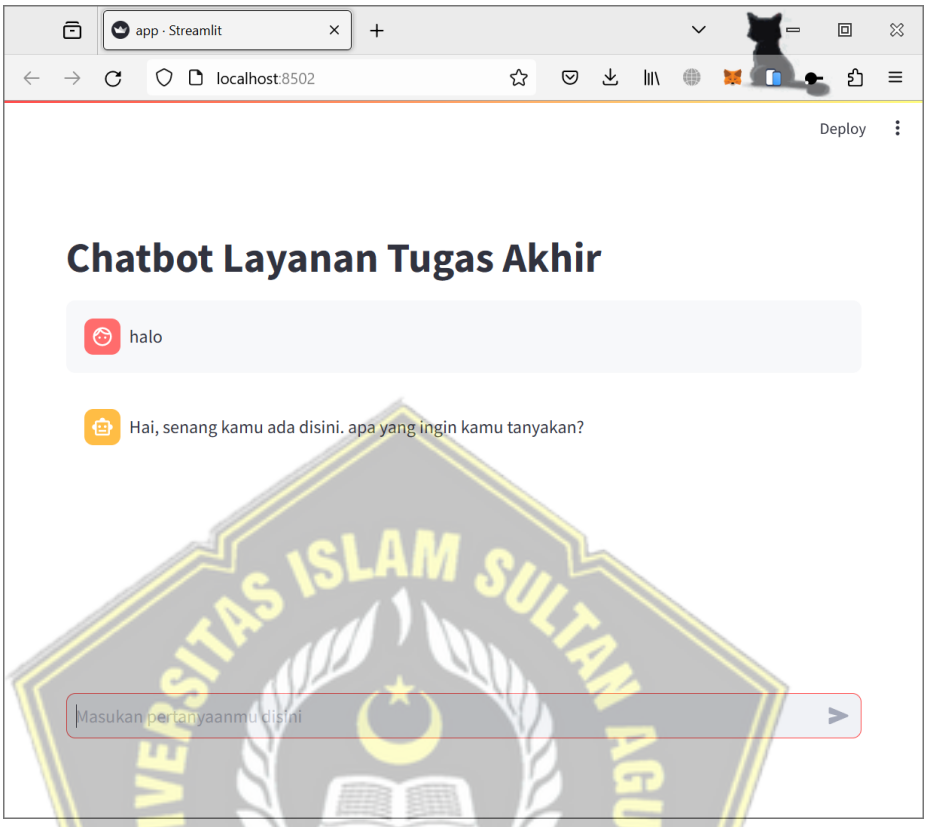
2. Halaman saat memasukkan pertanyaan



Gambar 4. 21 Memberikan pertanyaan melalui kolom *input text*

Pada gambar 4.3 merupakan hasil tangkapan layar saat menyapa *Chatbot* dengan menginputkan pertanyaan "halo", dari pertanyaan yang telah dikirim, *Chatbot* akan memproses dengan memberikan *responses* yang sesuai dengan pertanyaan seperti pada gambar 4.4.

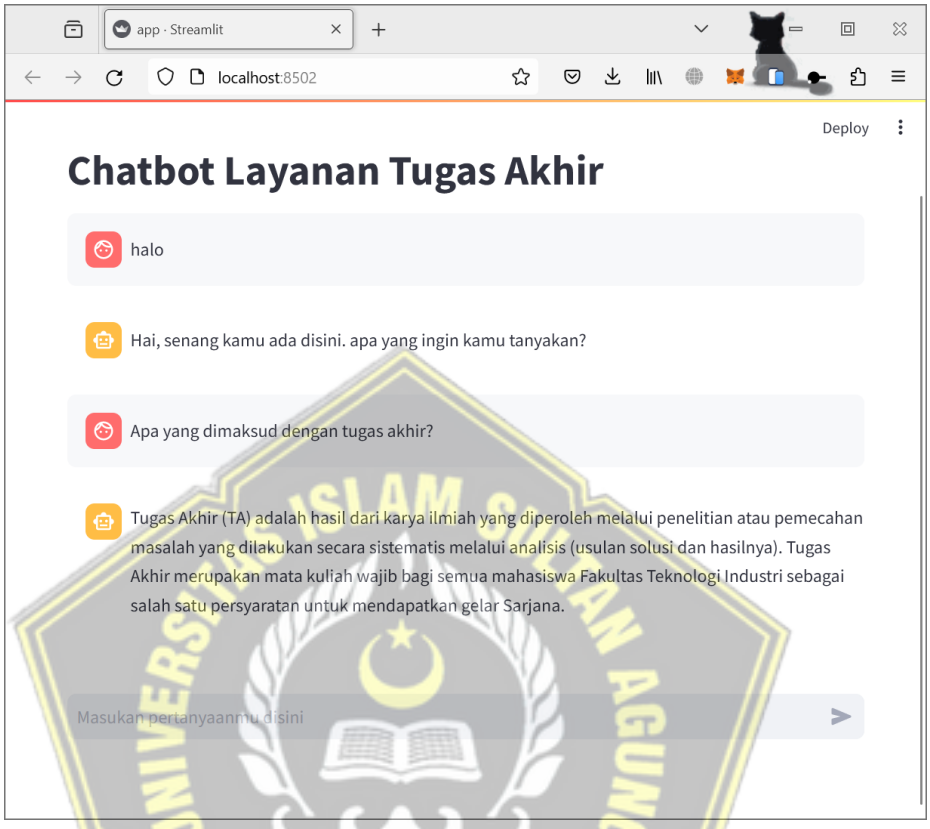
3. Halaman saat *Chatbot* memberikan responses dari pertanyaan yang diberikan



Gambar 4. 22 *Chatbot* memberikan *responses* dari input pertanyaan "halo"

Pada gambar 4.4 *Chatbot* telah memproses pertanyaan "halo" dan memberikan *responses* yang baik dan akurat sesuai dengan pertanyaan. Selanjutnya mencoba memberikan pertanyaan lain yang berhubungan dengan Tugas Akhir.

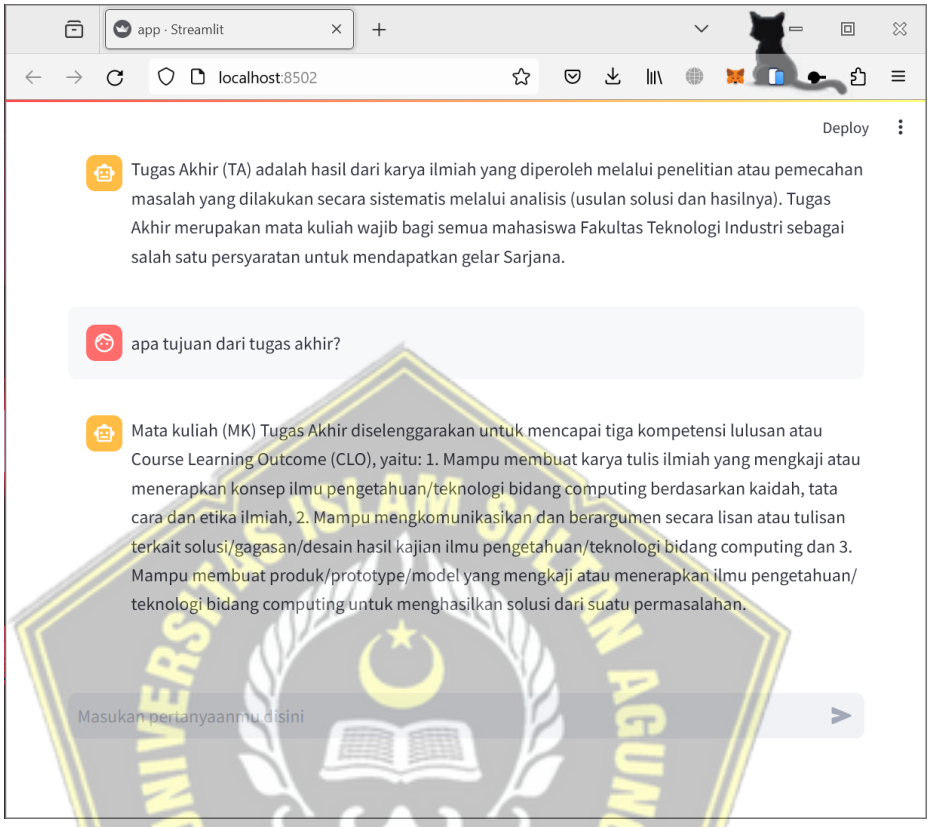
4. Halaman saat *Chatbot* memberikan responses dari pertanyaan tentang Tugas Akhir



Gambar 4. 23 memberikan pertanyaan tentang Tugas Akhir

Pada gambar 4.5 menampilkan hasil tangkapan layar dari demo mencoba memberikan pertanyaan tentang Tugas Akhir. *Chatbot* telah memproses pertanyaan dan dapat memberikan *responses* yang baik dan akurat sesuai dengan pertanyaan yang diberikan.

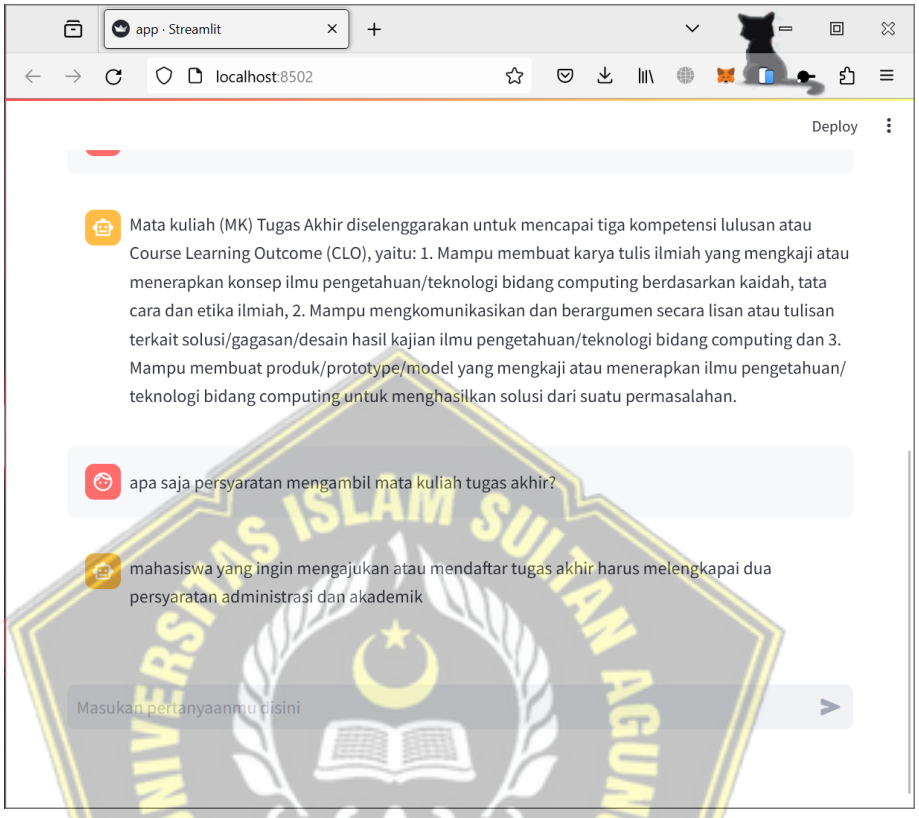
5. Halaman saat *Chatbot* memberikan *responses* dari pertanyaan tentang Tugas Akhir



Gambar 4. 24 memberikan pertanyaan ke 2 tentang Tugas Akhir

Pada percakapan yang terjadi pada gambar 4.6, setelah menginputkan pertanyaan mengenai Tugas Akhir lagi. *Chatbot* memberikan responses yang sesuai dan akurat sesuai dengan pertanyaan yang dimasukan.

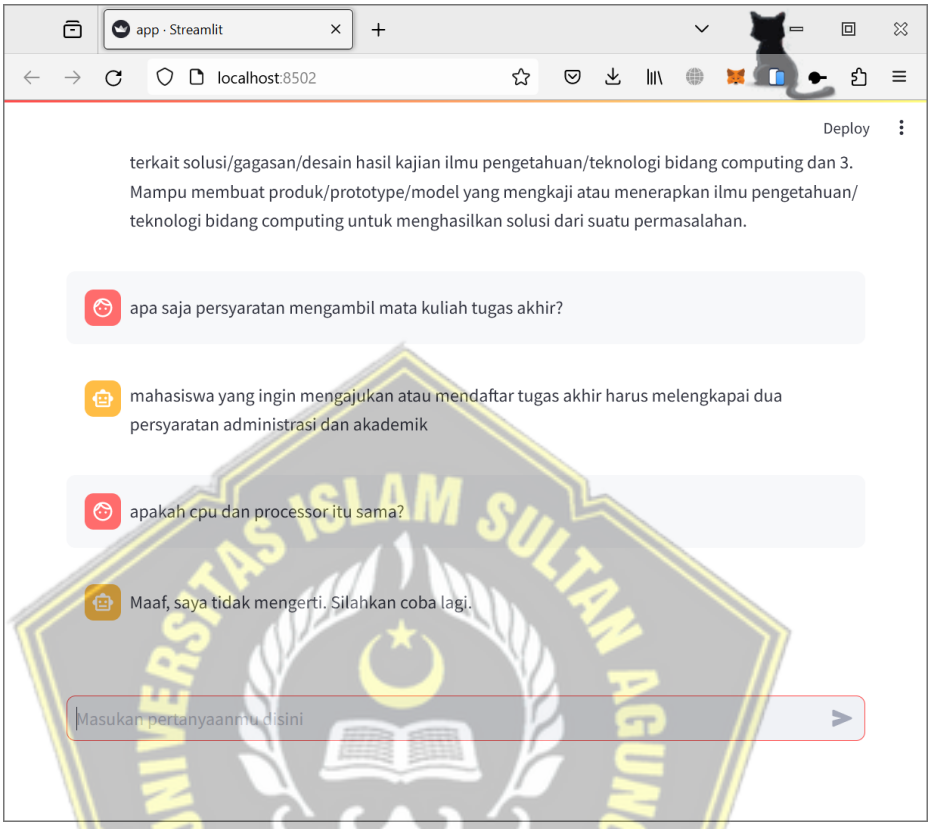
6. Halaman saat *Chatbot* memberikan responses dari pertanyaan tentang Persyaratan Tugas Akhir



Gambar 4. 25 Memberikan pertanyaan tentang persyaratan Tugas Akhir

Pada percakapan yang terjadi pada gambar 4.7, setelah menginputkan pertanyaan mengenai persyaratan Tugas Akhir, *Chatbot* kembali memberikan responses yang sesuai dan akurat sesuai dengan pertanyaan yang dimasukan.

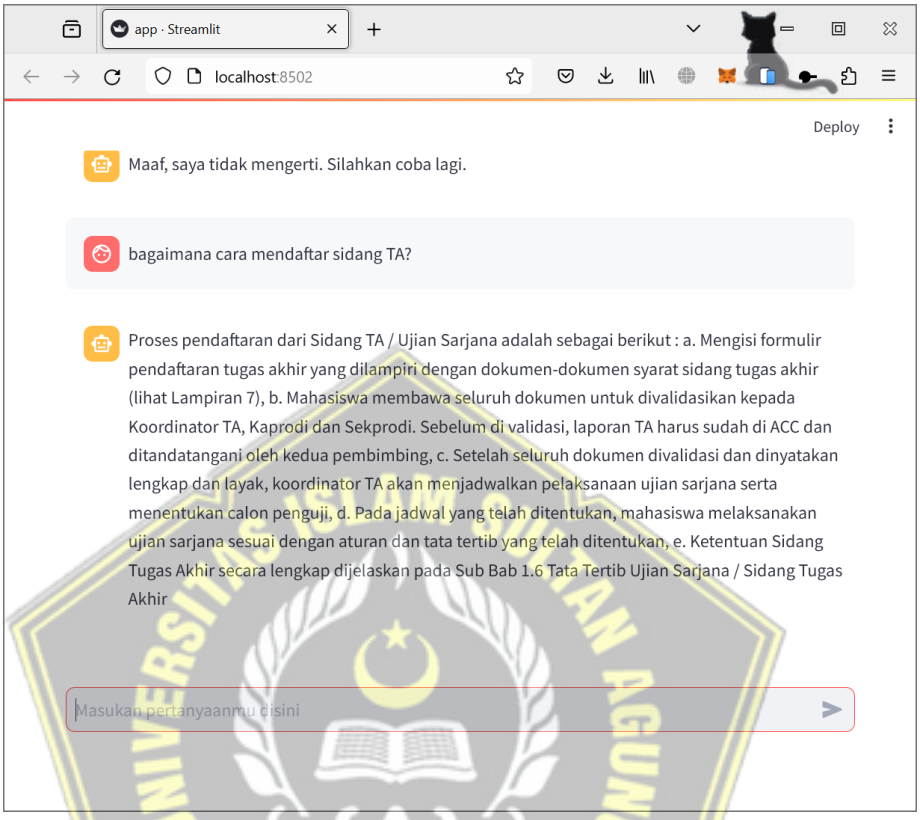
7. Halaman saat *Chatbot* memberikan responses dari pertanyaan diluar topik Tugas Akhir



Gambar 4. 26 Memberikan pertanyaan diluar topik Tugas Akhir

Pada gambar 4.8, mencoba memberikan pertanyaan diluar topik Tugas Akhir dan hasilnya *Chatbot* memberikan responses berupa jawaban “Maaf, saya tidak mengerti. Silahkan coba lagi” jawaban ini tidak sesuai dengan pertanyaan dikarenakan pertanyaan yang diberikan diluar topik tentang Tugas Akhir.

8. Halaman saat *Chatbot* memberikan responses dari pertanyaan tentang sidang Tugas Akhir



Gambar 4. 27 Memberikan pertanyaan tentang sidang Tugas Akhir

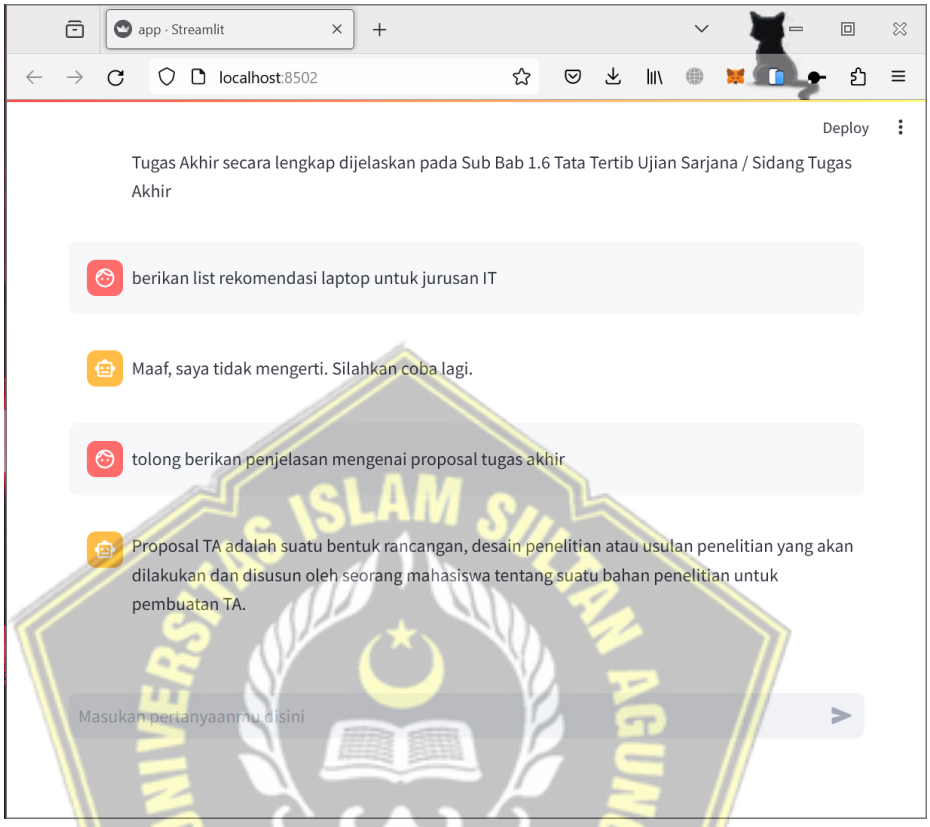
Pada gambar 4.9, mencoba kembali memberikan pertanyaan tentang sidang Tugas Akhir, *Chatbot* kembali memberikan responses yang baik dan akurat sesuai dengan pertanyaan yang diberikan. Selanjutnya saya akan mencoba memberikan perintah kepada *chatbot* untuk menjelaskan sesuatu diluar topik pembahasan tugas akhir.

9. Halaman saat *Chatbot* memberikan responses dari pertanyaan diluar topik Tugas Akhir

Gambar 4. 28 Memberikan perintah diluar topik tugas akhir

Terlihat pada gambar 4.28 mencoba memasukan pertanyaan berupa perintah diluar topik pembahasan tugas akhir dan hasilnya *Chatbot* memberikan responses berupa jawaban “Maaf, saya tidak mengerti. Silahkan coba lagi.” jawaban ini tidak sesuai dengan pertanyaan dikarenakan pertanyaan yang diberikan diluar topik tentang Tugas Akhir. Kemudian saya akan memasukan pertanyaan berupa perintah yang sesuai dengan topik tentang tugas akhir.

10. Halaman saat *Chatbot* memberikan responses dari pertanyaan tentang proposal Tugas Akhir



Gambar 4. 29 Memberikan pertanyaan tentang proposal tugas akhir

Pada gambar 4.29 mencoba memberikan pertanyaan berupa perintah mengenai definisi dari proposal tugas akhir dan hasilnya *Chatbot* kembali dapat merespon dengan baik dan memberikan jawaban yang akurat dan sesuai. Dari beberapa hasil tangkapan layar saat mendemokan *Chatbot* dapat disimpulkan bahwa *Chatbot* mampu bekerja dengan baik dengan memberikan responses yang akurat jika pertanyaan yang diberikan berkaitan dengan Tugas Akhir.

4.4 Hasil Uji Coba Sistem

Pengujian *black box* dilakukan pada tahap pengujian sistem *Chatbot* Layanan Informasi Tugas Akhir program studi Teknik Informatika UNISSULA. Pengujian *black box* digunakan untuk memasukkan *input* dan menguji apakah fungsi-fungsi sistem yang sedang berjalan sudah sesuai dengan tujuan dan tercermin dalam hasil keluaran. Rencana yang akan dibuat termasuk *input*, hasil yang diharapkan, *output*, dan kesimpulan dari *input text* dan hasil percakapan ditunjukkan pada table 4.3.

Tabel 4. 2 Pengujian *black box*

Input	**Hasil yang diharapkan**	***Output***	**Kesimpulan**
Menginput pertanyaan "halo"	Menampilkan Pertanyaan dan jawaban di bagian *body*	Pertanyaan dan jawaban ditampilkan pada bagian *body*	*Chatbot* meresponses dengan akurat
Menginput pertanyaan "apa yang dimaksud dengan tugas akhir?"	Menampilkan Pertanyaan dan jawaban di bagian *body*	Pertanyaan dan jawaban ditampilkan pada bagian *body*	*Chatbot* meresponses dengan akurat
Menginput pertanyaan "apa tujuan dari tugas akhir?"	Menampilkan Pertanyaan dan jawaban di bagian *body*	Pertanyaan dan jawaban ditampilkan pada bagian *body*	*Chatbot* meresponses dengan akurat
Menginput pertanyaan "apa saja persyaratan mengambil mata kuliah tugas akhir?"	Menampilkan Pertanyaan dan jawaban di bagian *body*	Pertanyaan dan jawaban ditampilkan pada bagian *body*	*Chatbot* meresponses dengan akurat
Menginput pertanyaan "apakah cpu dan processor itu sama?"	Menampilkan Pertanyaan dan jawaban di bagian *body*	Pertanyaan dan jawaban ditampilkan pada bagian *body*	*Chatbot* meresponses tidak akurat
Menginput pertanyaan "bagaimana cara mendaftar sidang TA?	Menampilkan Pertanyaan dan jawaban di bagian *body*	Pertanyaan dan jawaban ditampilkan pada bagian *body*	*Chatbot* meresponses dengan akurat
Menginput pertanyaan "berikan list rekomendasi laptop untuk jurusan IT"	Menampilkan Pertanyaan dan jawaban di bagian *body*	Pertanyaan dan jawaban ditampilkan pada bagian *body*	*Chatbot* meresponses tidak akurat

Input	**Hasil yang diharapkan**	*Output*	**Kesimpulan**
Menginput pertanyaan "tolong berikan penjelasan mengenai proposal tugas akhir"	Menampilkan Pertanyaan dan jawaban di bagian *body*	Pertanyaan dan jawaban ditampilkan pada bagian *body*	*Chatbot* meresponses dengan akurat

Hasil pengujian black box ditunjukkan pada tabel 4.1. Enam pertanyaan diberikan dapat dijawab dengan baik dan memberikan jawaban yang akurat, tetapi dua pertanyaan menghasilkan tanggapan atau jawaban yang tidak akurat. karena data model yang sudah dilatih tidak mengandung pertanyaan yang diberikan. Dengan demikian, *Chatbot* Layanan Informasi Tugas Akhir akan memberikan tanggapan yang akurat untuk pertanyaan yang berkaitan dengan tugas akhir.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian ini, dapat diambil kesimpulan tentang Sistem Chatbot sebagai Layanan Tugas Akhir program studi Teknik Informatika dengan menggunakan metode *Bidirectional Encoder Representations from Transformers* (BERT), *chatbot* yang dihasilkan mampu memberikan kinerja yang baik dalam mengklasifikasikan pertanyaan dan respon dalam percakapan. Hasil evaluasi menunjukkan bahwa model BERT dengan akurasi 99,22%, F1-score 99,3%, presisi 99,4%, dan recall 99,3% mampu memahami konteks dengan baik dan memberikan jawaban yang relevan. Selain itu, Model BERT ini dapat digunakan sebagai dasar untuk dikembangkan lebih lanjut dalam membangun *chatbot* responsif.

5.2 Saran

Saran untuk pengembangan sistem chatbot layanan informasi tugas akhir menggunakan metode *Bidirectional Encoder Representations from Transformers* (BERT) adalah dengan memperbarui pendekatan dataset. Salah satu pendekatan yang menarik adalah menghasilkan dataset secara otomatis melalui model generatif seperti *Generative Pre-trained Transformer* (GPT) atau model serupa. Dataset ini dapat mencakup jawaban dan pertanyaan yang dihasilkan secara otomatis. Serta menciptakan kerangka kerja yang lebih dinamis dan responsif. melibatkan interaksi aktif pengguna, menangani kesalahan dan ketidakpastian, melakukan evaluasi berkelanjutan, memperhatikan privasi dan keamanan data, Dataset juga bisa dibuat terintegrasi dengan *database* sehingga bisa diupdate secara otomatis dan untuk mendukung pembaruan dataset secara periodik. diharapkan dapat meningkatkan kemampuan chatbot dalam memberikan jawaban yang lebih kontekstual dan relevan.

DAFTAR PUSTAKA

Abijono, H., Santoso, P. dan Anggreini, N. L. (2021) "Algoritma Supervised Learning Dan Unsupervised Learning Dalam Pengolahan Data," *Jurnal Teknologi Terapan: G-Tech*, 4(2), hal. 315–318. doi: 10.33379/gtech.v4i2.635.

Afifa, N., Saputra, E. R. dan Nugrahaeni, A. R. (2023) "Implementasi NLP Pada Chatbot Layanan Akademik Dengan Algoritma Bert Implementation Of NLP On Academic Service Chatbot With Bertalgorithm," 10(1), hal. 383–387.

Amalia, E. dan Wibowo, D. (2019) "Rancang Bangun Chatbot Untuk Meningkatkan Performa Bisnis," *Jurnal Ilmiah Teknologi Informasi Asia*, 13(2), hal. 137–142.

Devlin, J. *dkk.* (2019) "BERT: Pre-training of deep bidirectional transformers for language understanding," *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1(Mlm), hal. 4171–4186.

Hikmah, A., Azmi, F. dan Nugrahaeni, R. A. (2023) "Implementasi Natural Language Processing Pada Chatbot Untuk Layanan Akademik," *e-Proceeding of Engineering*, 10(1), hal. 371–382.

HuggingFace (2020a) *Transformers Documentation, huggingface.co*. Tersedia pada: https://huggingface.co/docs/transformers/pad_truncation.

HuggingFace (2020b) *Transformers Documentation, huggingface.co*. Tersedia pada: https://huggingface.co/docs/transformers/preprocessing.

Isa, S. M., Nico, G. dan Permana, M. (2022) "Indobert for Indonesian Fake News Detection," *ICIC Express Letters*, 16(3), hal. 289–297. doi: 10.24507/icicel.16.03.289.

Jurafsky, D. dan James, H. M. (2017) *Speech and Language Processing, The 1870 Ghost Dance*. doi: 10.2307/j.ctt1djmg7w.13.

Khesya, N. (2021) "Mengenal Flowchart Dan Pseudocode Dalam Algoritma Dan Pemrograman," *Osf.Io*.

Koto, F. *dkk.* (2020) "IndoLEM and IndoBERT: A Benchmark Dataset and Pre-

trained Language Model for Indonesian NLP," *COLING 2020 - 28th International Conference on Computational Linguistics, Proceedings of the Conference*, hal. 757–770. doi: 10.18653/v1/2020.coling-main.66.

Muhammad, R., Ardimansyah, M. I. dan Afini, A. (2022) *Mengembangkan Sistem Percakapan Otomatis Berbasis Layanan Pesan Instan*. Tersedia pada: https://books.google.com/books?hl=en&lr=&id=BDWnEAAAQBAJ&oi=fnd&pg=PA1&dq=Chatbot+dengan+tingkat+AI+yang+rendah+hanya+dapat+memberikan+respons+terbatas+terhadap+pertanyaan+dan+instruksi+yang+diberikan+oleh+pengguna.&ots=kxwPOf-nRW&sig=KRLXYGoAqXuNmDS9ex7l.

Nugroho, M. A. *dkk.* (2021) "Pengembangan Aplikasi Qna Untuk Pendaftaran Mahasiswa Baru Stmik Akakom," *Journal of Information System Management (JOISM)*, 3(1), hal. 18–23. doi: 10.24076/joism.2021v3i1.408.

Putu, N. *dkk.* (2023) "Analisis Sentimen terhadap Perundungan Siber pada Twitter menggunakan Algoritma Bidirectional Encoder Representations from Transformer (BERT)," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 7(2), hal. 909–916. Tersedia pada: https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/12345.

Rayyan, F. (2022) *Pengembangan chatbot untuk aplikasi online chat telegram dengan pendekatan klasifikasi emosi pada teks menggunakan metode indobert-lite, Repository.Uinjkt.Ac.Id*. Tersedia pada: https://repository.uinjkt.ac.id/dspace/handle/123456789/65718%0Ahttps://repository.uinjkt.ac.id/dspace/bitstream/123456789/65718/1/FAISHAL RAYYAN-FST.pdf.

Rohman, A. N., Utami, E. dan Raharjo, S. (2019) "Deteksi Kondisi Emosi pada Media Sosial Menggunakan Pendekatan Leksikon dan Natural Language Processing," *Eksplora Informatika*, 9(1), hal. 70–76. doi: 10.30864/eksplora.v9i1.277.

Vaswani, A. *dkk.* (2017) "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017-Decem(Nips), hal. 5999–6009.

Wahyudi, H. dan Fitria, L. (2020) "Aplikasi Chatbot Berbasis Web Menggunakan Metode Dialogflow," hal. 7.

Wijayanto, R., Pradana, F. dan Bachtiar, F. A. (2020) “Pembangunan Sistem Chatbot Informasi Objek Wisata Kota Malang berbasis Web,” 4(5).

Wintoro, P. B. *dkk.* (2022) “Implementasi Long Short-Term Memory pada Chatbot Informasi Akademik Teknik Informatika Unila,” *EXPERT: Jurnal Manajemen Sistem Informasi dan Teknologi*, 12(1), hal. 68. doi: 10.36448/expert.v12i1.2593.