BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

2.1.1 Aplikasi *Chatbot* sebagai Layanan Informasi Sekolah

2.1.1.1 Pengertian dan Jenis-Jenis *Chatbot*

Chatbot, yang juga dikenal sebagai *chatterbots*, diciptakan sebagai upaya untuk menyimulasikan percakapan manusia. *Chatbot* semakin banyak digunakan di berbagai bidang seperti pendidikan, bisnis, dan *e-commerce*, di mana *Chatbot* bertindak sebagai asisten *online* otomatis untuk melengkapi dan menggantikan layanan yang disediakan oleh manusia (McTear, 2021:21).

Chatbot dapat digolongkan sebagai jenis *question-answering system* (Q&A) atau sistem tanya jawab. Sistem tanya jawab dalam aplikasi *Chatbot* melibatkan dua komponen utama: 1) pengetahuan umum; dan 2) pengetahuan untuk tugas-tugas spesifik. Pengetahuan umum akan membantu *Chatbot* dengan pertanyaan dan fakta sehari-hari, sedangkan pengetahuan untuk tugas-tugas spesifik yang dapat memberikan jawaban terperinci berdasarkan *database* informasi yang besar, layaknya seorang ahli di bidang tertentu (Lee, 2023:225).

Chatbot umumnya dikelompokkan ke dalam beberapa kategori utama. (Crowder & Carbone, 2023) mengelompokkan *Chatbot* ke dalam lima kategori berikut:

1. *Menu-Based Chatbots*, yang menggunakan model *decision tree* untuk membimbing pengguna melalui serangkaian pilihan menu hingga keputusan akhir dibuat.
2. *Rule-Based Chatbots*, yang memanfaatkan aturan linguistik yang telah ditentukan untuk memahami pertanyaan pengguna dan memberikan tanggapan tertulis.
3. *Machine Learning-Based Chatbots*, yang memanfaatkan teknologi AI sederhana. *Chatbot* ini akan belajar dari interaksi dengan pengguna dan menyimpan detail percakapan berbasis teks untuk meningkatkan pemahaman dan respons mereka.
4. *Voice-Based Chatbots*, yang menggunakan suara sebagai antarmuka utama. *Chatbot* ini memerlukan perangkat lunak kompleks untuk menerjemahkan suara pengguna menjadi teks untuk diproses.
5. *Cognition-Based Chatbots*, yang bertujuan meniru proses berpikir manusia dengan memanfaatkan teknologi AI, seperti *machine learning*, untuk memproses informasi secara mirip dengan kognisi manusia, memungkinkan mereka untuk terlibat dalam percakapan yang lebih kompleks dan bermakna.

2.1.1.2 Aplikasi *Chatbot* sebagai Layanan Informasi Sekolah

Aplikasi *Chatbot* telah menjadi salah satu solusi inovatif untuk memenuhi kebutuhan informasi dalam lingkungan sekolah. Layanan *Chatbot* tersedia 24/7, sehingga dapat diakses kapan pun dibutuhkan, bahkan di luar jam sekolah. Oleh karena itu, pengguna bisa mendapatkan informasi tanpa harus menunggu jam kantor atau kegiatan sekolah. Selain itu, dengan menggunakan *Chatbot*, pengguna

dapat dengan mudah menemukan informasi yang mereka butuhkan tanpa harus melakukan pencarian yang rumit atau navigasi situs web sekolah yang kompleks.

Sebagai layanan informasi sekolah, aplikasi *Chatbot* dapat memberikan bantuan dalam berbagai tugas sekolah (Shah, 2023), meliputi:

1. FAQ Sekolah

   Chatbot dapat menjadi sumber informasi utama untuk pertanyaan umum seputar sekolah, seperti jadwal, lokasi fasilitas, kegiatan ekstrakurikuler yang tersedia, dan informasi umum tentang kebijakan sekolah.

2. Layanan Bantuan Pembelajaran

   Chatbot dapat memberikan bantuan dalam pembelajaran dengan memberikan jawaban atas pertanyaan siswa tentang materi pelajaran dan memberikan saran tentang sumber belajar yang berguna.

3. Sarana Komunikasi

   Chatbot dapat menjadi saluran komunikasi yang efisien antara guru, staf akademik, siswa-siswi, dan orang tua dengan memberikan informasi terkini tentang kegiatan sekolah atau siswa-siswi, dan menjadi perantara komunikasi antara semua pihak terkait di sekolah.

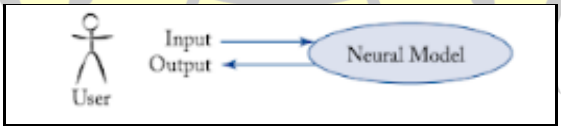
4. Bantuan Tugas-tugas Administratif

   Chatbot dapat membantu dalam tugas-tugas administratif seperti pendaftaran siswa baru, pengaturan jadwal ujian, pengumpulan formulir dan dokumen, serta memberikan informasi terkait prosedur dan persyaratan yang diperlukan.

2.1.1.3 Aplikasi *Chatbot* dengan Pendekatan *Neural Networks*

(McTear, 2021) menyatakan bahwa terdapat tiga pendekatan utama dalam pengembangan sistem *Chatbot*, yaitu: 1) *rule-based*, di mana alur percakapan disusun secara manual berdasarkan aturan-aturan oleh perancang sistem; 2) *statistical data-driven*, di mana strategi percakapan disusun berdasarkan data yang dipelajari sistem; serta 3) *end-to-end neural*, yang merupakan perpaduan dari pendekatan *statistical data-driven* dengan struktur *neural network*.

Sejak tahun 2014, penelitian di bidang sistem tanya jawab dan *Chatbot* mulai didominasi dengan pendekatan *end-to-end neural*. Struktur *neural network* dalam sistem *Chatbot* dibuat untuk meniru otak manusia dengan menggunakan arsitektur *Encoder-Decoder*, yang merupakan model *sequence-to-sequence*. Arsitektur ini menggabungkan *natural memory recall priority* dan konteks dengan mekanisme *attention*. Arsitektur ini memiliki dua tugas utama: memproses *input* (*encoding*) dan menghasilkan *output* (*decoding*), diilustrasikan pada Gambar 2.1. Selain arsitektur *Encoder-Decoder*, *Chatbot* juga umum dikembangkan menggunakan arsitektur RNN seperti LSTM dan GRU (McTear, 2021).



Gambar 2.1 Arsitektur *end-to-end neural* (McTear, 2021:126)

2.1.2 *Natural Language Processing* dengan Pendekatan *Deep Learning*

2.1.2.1 *Artificial Intelligence, Machine Learning*, dan *Deep Learning*

Artificial intelligence (AI) atau kecerdasan buatan adalah bidang studi yang luas, dengan *machine learning* sebagai salah satu subbidang di dalamnya. AI menggabungkan kata "*artificial*" (buatan manusia) dan "*intelligence*" (kemampuan untuk berpikir, memahami, dan merencanakan). Dengan demikian, teknologi atau program apa pun yang memungkinkan mesin meniru pemikiran atau tindakan manusia termasuk dalam AI (Geetha & Sendhilkumar, 2023:3).

(Geetha & Sendhilkumar, 2023) berpendapat bahwa *machine learning* adalah sistem yang mempelajari model (program) dari data *input* dan *output* yang mengoptimalkan kinerja untuk sekelompok tugas tertentu menggunakan data contoh atau pengalaman masa lalu dengan secara otomatis mendeteksi pola dalam data menggunakan metode komputasi dan statistik. Salah satu cara untuk meninjau *machine learning* yaitu melalui tiga komponen utama: representasi, evaluasi, dan optimalisasi. Representasi mengacu pada bagaimana kita mendefinisikan apa yang kita pelajari. Evaluasi mengukur seberapa baik kinerja proses pembelajaran. Optimasi bertujuan untuk menemukan representasi terbaik berdasarkan kriteria dan evaluasi yang telah ditentukan.

Deep learning muncul pada awal abad ke-21, di saat jaringan saraf dengan banyak lapisan sulit untuk dilatih secara efektif. *Deep neural networks*, dengan jutaan atau bahkan milyaran parameter, menghasilkan kemajuan yang signifikan dalam berbagai domain. Model-model ini telah menjadi dasar dalam penelitian dan aplikasi *machine* learning modern (Bishop & Bishop, 2023).

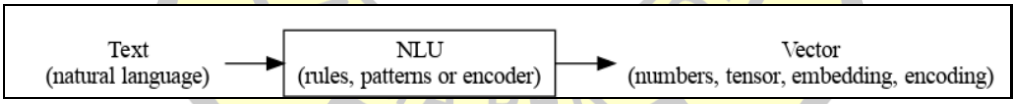
2.1.2.2 *Natural Language Processing* (NLP)

Natural Language Processing (NLP) mencakup penggunaan komputer terhadap bahasa manusia, seperti Bahasa Orang atau Bahasa Inggris. NLP, yang merupakan bagian dari bidang AI, dapat menafsirkan ucapan manusia untuk tujuan *human-computer interaction* (HCI), menciptakan pengetahuan terstruktur untuk tugas-tugas seperti pencarian informasi, peringkasan teks, analisis sentimen, pengenalan suara, *data mining*, *deep learning*, *machine translation*, dan *Q&A chatbot* (Lee, 2023:10).

(Lee, 2023:11) menguraikan tiga komponen utama dari NLP. Ketiga komponen tersebut meliputi:

1. *Natural Language Understanding* (NLU)

   NLU merujuk pada teknik atau metode yang dikembangkan untuk memahami makna bahasa lisan manusia melalui analisis sintaksis, semantik, dan pragmatis.



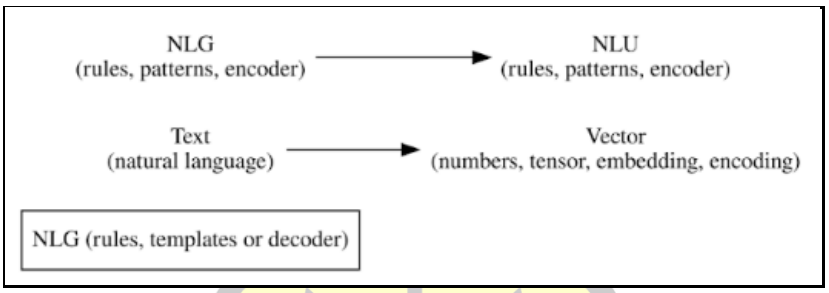
Gambar 2.2 Alur NLU (Lane & Dyshel, 2023)

2. *Knowledge Acquisition and Inferencing* (KAI)

   Setelah bahasa manusia berhasil dipahami oleh NLU, sistem KAI akan melanjutkan proses NLP. Sistem KAI bertujuan untuk mengelola pengetahuan di industri tertentu, seperti layanan pelanggan di bidang asuransi atau medis, tanpa menggunakan *if-then-else query* seperti pada sistem pakar.

3. *Natural Language Generation* (NLG)

NLG adalah proses penyusunan jawaban, tanggapan, dan *feedback* dalam percakapan antara manusia dan mesin. NLG layaknya penerjemah untuk mesin yang bertugas mengubah respons menjadi teks tertulis yang menyerupai bahasa manusia.



Gambar 2.3 Alur NLG (Lane & Dyshel, 2024:5)

Deep learning telah memberikan perkembangan yang signifikan dalam tugas-tugas NLP. Penggunaan *deep learning*, seperti RNN dan *Transformers*, pada NLP memungkinkan penghasilan akurasi yang tinggi dalam masalah-masalah yang kompleks seperti tanya-jawab, pemahaman bacaan, meringkas, dan menyimpulkan bahasa alami. Kini, *deep learning* telah menghadirkan aplikasi-aplikasi inovatif seperti percakapan yang interaktif dan bahkan penulisan perangkat lunak (Lane & Dyshel, 2024).

2.1.2.2.1 *Tokenization*

Tokenization dalam NLP melibatkan penguraian teks menjadi bagian yang lebih kecil dan bermakna yang dikenal sebagai *token*. *Token* dapat berupa kata, tanda baca, emoji, angka, atau elemen lain yang mirip kata. *Tokenization* sangat penting untuk memproses teks dalam tugas-tugas NLP dan sering kali merupakan langkah pertama dalam *pipeline* NLP. *Tokenization* memungkinkan konversi teks yang tidak terstruktur menjadi data numerik terstruktur, sehingga cocok untuk

digunakan sebagai *feature* dalam algoritma *machine learning* (Lane & Dyshel, 2024).

2.1.2.2.2 *Word Embeddings*

(Lane & Dyshel, 2024) mendefinisikan *word embeddings* sebagai vektor yang digunakan untuk merepresentasikan makna, mirip dengan cara otak menyimpan makna. Neuron di otak mengirimkan sinyal untuk mengasosiasikan makna kata-kata, membentuk berbagai koneksi. *Word embeddings* menggambarkan hubungan antara kata-kata ini, yang berfungsi sebagai representasi kasar dari rangkaian jaringan saraf di otak.

2.1.2.3 *Language Model* (LM)

(Arun R, 2024) *Language model* (LM), atau model 18ensit, merupakan komponen utama dari banyak tugas NLP. Pada dasarnya, LM merupakan model yang dapat memprediksi kata berikutnya dalam suatu urutan kata dengan belajar dari data teks. Perkembangan LM telah memungkinkan terwujudnya model-model yang dapat memahami dan menghasilkan teks seperti manusia, menunjukkan kemajuan yang besar pada tugas-tugas NLP. Perkembangan ini dapat dibagi menjadi empat fase sebagai berikut:

1. *Statistical Language Model* (SLM)

   Pada awalnya, LM dibangun dengan menggunakan metode statistk, namun LM dengan metode ini memiliki keterbatasan dalam memahami konteks pada teks yang 18ensiti.

2. *Neural Language Model* (NLM)

   Dengan munculnya metode *neural networks*, LM mampu dirancang sehingga dapat lebih memahami konteks. *Neural networks* memperkenalkan representasi kata yang terdistribusi, di mana kata-kata direpresentasikan sebagai 19 ensit fitur, yang menangkap makna semantiknya.

3. *Pretrained Language Model* (PLM)

   Pada fase ini, mulai muncul model-model *pretrained* seperti ELMo dan BERT. PLM merupakan model *neural networks* yang dilatih pada dataset teks yang besar yang dapat dilatih lebih lanjut dengan dataset yang lebih sedikit dan spesifik untuk tugas NLP tertentu.

4. *Large Language Model* (LLM)

   LLM seperti GPT-3 pada dasarnya adalah PLM dengan skala lebih besar, yang melibatkan puluhan atau ratusan miliar parameter. Parameter ini disempurnakan melalui pelatihan, sehingga model dapat belajar dari data dan secara efektif melakukan tugas-tugas NLP. Dengan menerapkan arsitektur Transformers, LLM memproses *input* melalui mekanisme *encoding* dan *decoding* untuk menghasilkan *output* yang dapat diprediksi.

2.1.2.4 Parameter LLM untuk Tugas *Text Generation*

(Arun R, 2024) LLM yang dilatih untuk tugas *text generation* menyediakan beberapa parameter yang dapat disesuaikan. Parameter-parameter ini mengendalikan bagaimana model menghasilkan teks, serta mempengaruhi

keberagaman dan kreativitas *output* teks. Berikut ini adalah penjelasan dari beberapa parameter LLM:

1. *Temperature*

   Parameter ini memiliki peran yang penting dalam menentukan tingkat keberagaman dan kreativitas model dalam menghasilkan teks. *Temperature* yang lebih rendah akan menghasilkan *output* yang lebih terfokus dan tepat, sedangkan temperature yang tinggi akan memungkinkan *output* yang lebih fleksibel dan kreatif.

2. *Top-p*

   Parameter *top-p* atau *nucleus sampling* merupakan cara lain untuk mengendalikan seberapa beragam dan kreatif teks yang dihasilkan model. *Top-p* menetapkan batas probabilitas dan hanya mengambil kata-kata dari kelompok terkecil yang memenuhi batas ini. Hal ini membantu model mengubah jumlah kata yang dipertimbangkan saat membuat teks.

3. *Max Tokens*

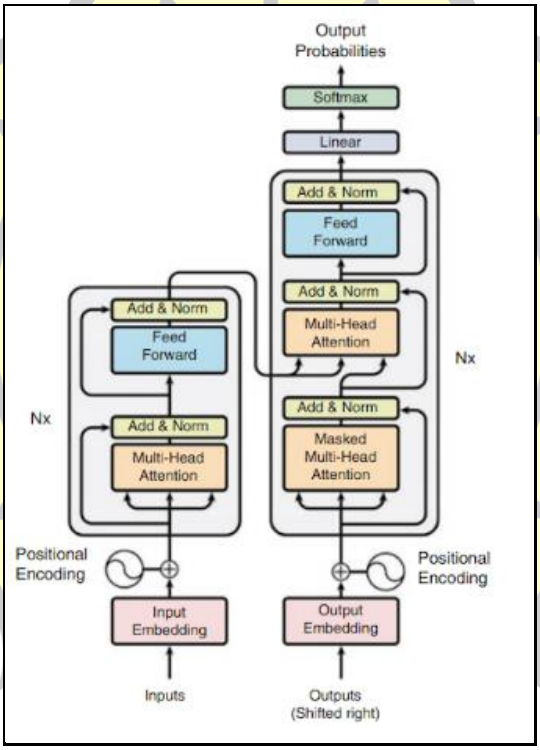
   Parameter ini menetapkan batas jumlah *token* yang dapat dihasilkan dalam *output*. *Max tokens* berguna untuk mengendalikan 20 anjang teks yang dihasilkan dan memastikannya tidak melebihi batas *token* model.

2.1.3 Metode *Transformers*

2.1.3.1 Definisi *Transformers*

Menurut Vaswani dalam (Lee, 2023:188), *Transformers* merupakan jaringan yang didasarkan pada mekanisme *attention* tanpa unit berulang (*recurrent*)

dan konvolusi (*convolutional*). *Attention* memungkinkan jaringan untuk memberikan bobot yang berbeda pada input yang berbeda, dengan koefisien pembobotan yang bergantung pada nilai *input*, sehingga dapat menangkap bias induktif yang kuat yang terkait dengan sekuensial dan bentuk data lainnya. Model ini disebut dengan istilah *Transformers* karena mengubah sekelompok angka dari satu bentuk representasi informasi ke bentuk yang lain, namun dengan ukuran yang sama (Bishop & Bishop, 2023:373). Gambar 2.3 menunjukkan arsitektur *Transformers*.



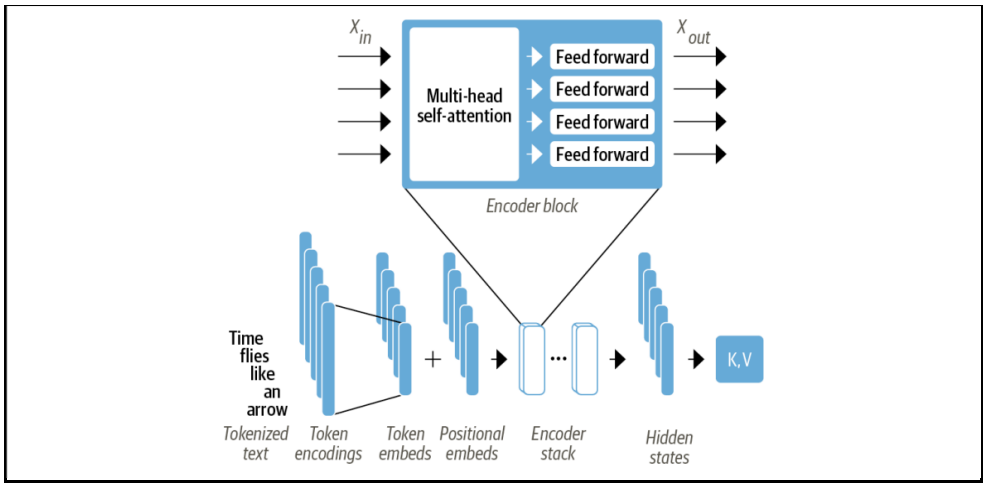
Gambar 2.4 Arsitektur *Transformers* (Lee, 2023:189)

(Tunstall et al, 2022:59) berpendapat bahwa model berbasis *Transformers* pada umumnya termasuk ke dalam tiga kategori utama, yaitu:

1. *Encoder-only*, seperti BERT, mengubah *input* teks menjadi representasi numerik yang berguna untuk tugas-tugas seperti klasifikasi teks atau *named entity recognition*.
2. *Decoder-only*, seperti GPT, memprediksi kata berikutnya dalam urutan yang diberikan teks. Model-model ini sering digunakan untuk tugas *autocompletion*.
3. *Encoder-Decoder*, seperti BART dan T5, yang menggabungkan komponen *encoding* dan *decoding* dan digunakan untuk tugas-tugas seperti penerjemahan dan peringkasan.

2.1.3.2 Arsitektur *Encoder* Model *Transformers*

Lapisan *encoder* model *Transformers* terdiri dari beberapa lapisan *encoder* yang disusun secara berurutan. Setiap lapisan *encoder* menerima *input embedding* dan diteruskan melalui dua sublayer: layer *multi-head self-attention layer* dan layer *feedforward*. *Output embedding* dari setiap lapisan *encoder* mempertahankan ukuran yang sama dengan input yang diberikan (Tunstall et al, 2022:60). Gambar 2.5 menunjukkan ilustrasi lapisan *encoder* model *Transformers*.



Gambar 2.5 Gambar arsitektur *encoder* (Tunstall et al, 2022:60)

Di bawah ini adalah rincian dari komponen-komponen *encoder* yang oleh (Tunstall et al, 2022).

2.1.3.2.1 *Self-Attention*

Mekanisme *attention* memungkinkan model untuk memberikan bobot (*weight*) yang berbeda, atau "*attention*", pada setiap langkah dari urutan *input*, dibanding menghasilkan satu *hidden state*. Hal ini memungkinkan model untuk fokus pada *input token* yang relevan pada setiap langkah *decoding*, sehingga meningkatkan ketepatan antara kata-kata yang diterjemahkan dengan kalimat asal. *Attention* membantu memprioritaskan *state* mana yang akan digunakan.

(Tunstall et al, 2022) menjelaskan bahwa gagasan utama di balik *self-attention* adalah dibandingkan menggunakan *embedding* yang tetap untuk setiap *token*, dapat digunakan seluruh *sequence* untuk menghitung rata-rata bobot dari setiap *embedding*. *Self-attention* digunakan sebagai solusi dari kata-kata yang sama, namun memiliki arti yang berbeda dalam konteks yang berbeda. Dengan menggabungkan *token embedding* dengan bobot yang berbeda, *embedding* yang lebih sesuai konteks akan dihasilkan.

Salah satu metode perhitungan *self-attention* yang banyak digunakan adalah metode *scaled dot-product* yang digunakan pada *paper* "*Attention Is All You Need*." *Self-attention* dihitung dengan rumus di bawah ini:

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

Proses perhitungan attention dimulai dengan mengalikan *Query* (Q) dengan *Key* (K) transpos, kemudian hasilnya dibagi dengan faktor normalisasi d_k. Setelah

itu, hasilnya dilakukan operasi *softmax* untuk menghasilkan *weight* relatif dari setiap *Value* (V) berdasarkan relevansinya dengan Q dan K.

Menggunakan beberapa set proyeksi linier, yang dikenal sebagai *attention head*, memungkinkan model untuk mempertimbangkan berbagai aspek persamaan secara paralel. Perhitungan *multi-head self-attention* dilakukan dengan rumus berikut ini:

$$Multihead(Q,K,V) = Concat(head_1,\dots,head_n)W^0 \quad (2)$$

$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \quad (3)$$

Perhitungan dilakukan dengan memecah *input* menjadi beberapa *attention head* yang masing-masing melakukan perhitungan *attention* secara independen, kemudian menggabungkan (*concat*) hasilnya kembali dan mengubahnya menjadi representasi akhir menggunakan matriks dengan *weight* W^0.

2.1.3.2.2 *Feed-Forward Layer*

Feed-forward layer dalam *encoder* dan *decoder* adalah jaringan saraf dua layer sederhana yang terhubung sepenuhnya. Namun, jaringan ini memproses setiap *embedding* secara independen, bukan keseluruhan *sequence* sebagai vektor tunggal, sehingga menjadikannya lapisan *feed-forward* yang bergantung pada posisinya, atau "*position-wise feed-forward layer*".

2.1.3.2.3 *Layer Normalization*

(Tunstall et al, 2022) menguraikan dua jenis *layer normalization*, menormalkan setiap *input* dalam *batch* agar memiliki rata-rata nol dan varians

terpadu, yang dapat digunakan di model *Transformers*. Kedua jenis *layer* yaitu, *post layer normalization* dan *pre layer normalization.*

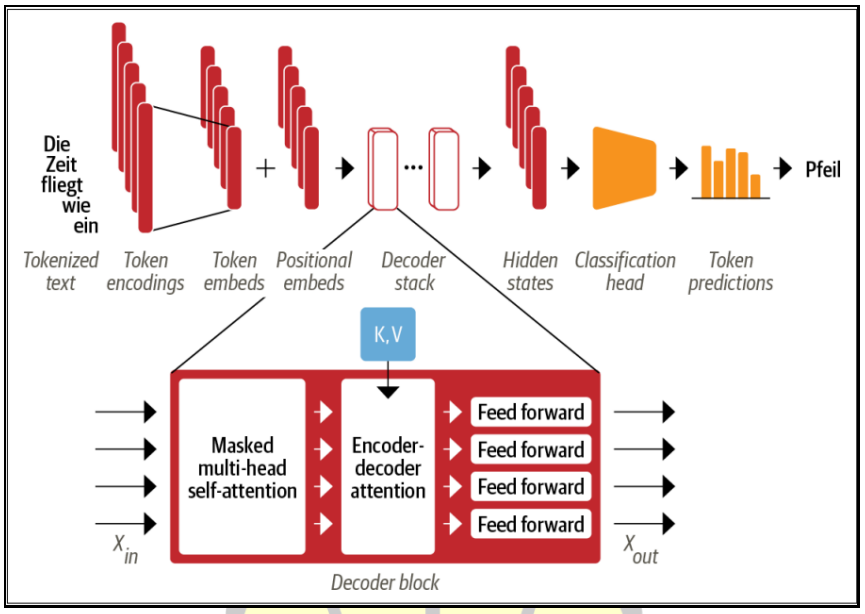
2.1.3.2.4 *Positional Embeddings*

Positional embeddings adalah komponen *Transformers* yang memungkinkan *attention head* dan *feed-forward layer* untuk mempelajari informasi posisi selama proses transformasi. *Positional embeddings* meningkatkan *token embeddings* dengan menggabungkan pola nilai yang bergantung pada posisi dalam sebuah vektor.

2.1.3.3 Arsitektur *Decoder* Model *Transformers*

Gambar 2.6 menunjukkan arsitektur dari *decoder*. (Tunstall et al, 2022) Arsitektur *decoder* model *Transformers* memiliki dua perbedaan utama dari arsitektur *encoder*, yaitu:

1. *Masked multi-head self-attention layer* yang memastikan bahwa *token* yang dihasilkan pada setiap langkah *decoding* hanya dipengaruhi oleh *token* yang dihasilkan sebelumnya dan *token* saat ini yang sedang diprediksi.
2. *Encoder-decoder attention layer* yang melakukan *multi-head attention* terhadap *output key* dan vektor nilai dari tumpukan *encoder*, menggunakan representasi perantara dari *decoder* sebagai pertanyaan. Ini memungkinkan *decoder* untuk mempelajari bagaimana menghubungkan *token* dari dua *sequence* yang berbeda



Gambar 2.6 Gambar arsitektur *decoder* (Tunstall et al, 2022:77)

2.1.3.4 *Transformer-Based Pretrained Model*

Pretrained model merupakan model atau sistem AI yang telah dilatih pada kumpulan data yang besar untuk mempelajari representasi atau pola umum. Model ini dapat disesuaikan dengan data tambahan untuk meningkatkan kinerjanya pada tugas-tugas tertentu. *Pretraining* memungkinkan model untuk menangkap fitur-fitur yang kaya yang dapat digeneralisasi dengan baik untuk data baru, sehingga efektif untuk berbagai tugas *downstream*. Untuk meningkatkan kinerja model, dapat dilakukan proses *fine-tuning* yang melibatkan pelatihan lebih lanjut pada dataset khusus. Pendekatan ini menghemat waktu dan sumber daya komputasi, memanfaatkan pengetahuan yang telah dipelajari sebelumnya untuk aplikasi baru (Thakur et al, 2024).

Pada saat ini, sebagian besar model *pretrained* didasarkan pada arsitektur *Transformers*, yang memungkinkan implementasi *transfer learning* untuk pengembangan sistem baru dengan efisien. Beberapa model *pretrained* berbasis *Transformers*, meliputi:

a. *Bidirectional Encoder Representations from Transformers* (BERT)

b. *Generative Pretrained Transformer* (GPT)

c. *Text-to-Text Transfer Transformer* (T5)

d. *Universal Sentence Encoder* (USE)

2.1.3.5 Jenis-jenis *Transformer-Based Pretrained Model* Terkait

2.1.3.5.1 *Sentence-Transformers* (SBERT)

Sentence-Transformers, atau yang kerap disebut dengan istilah SBERT, adalah jenis model berbasis *Transformers* yang dicetuskan pada *paper* berjudul "*Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*" oleh Reimers & Gurevych (2019). SBERT adalah modifikasi dari model *pretrained* BERT yang menggunakan struktur jaringan *siamese* dan *triplet* untuk mendapatkan *word embedding* yang bermakna secara semantik. Ini memungkinkan sepasang kalimat untuk dibandingkan dengan menggunakan perumusan *cosine similarity* dan mengurangi *computing cost* untuk menemukan kemiripan pasangan dari 65 jam dengan model BERT atau RoBERTa menjadi sekitar 5 detik dengan SBERT, tanpa mengurangi akurasi dari BERT.

Saat ini, terdapat lebih dari 5.000 model SBERT pada *Hugging Face Ecosystem* yang dapat digunakan secara langsung, maupun dilatih lebih lanjut dengan dataset tertentu. Jenis model SBERT yang digunakan dalam penelitian ini

adalah model SBERT *indo-sentence-bert-base* (Arasyi, 2022) yang telah dilatih menggunakan dataset berbahasa Indonesia dan menghasilkan *word embeddings* dengan ukuran dimensi 28ensit 768.

2.1.3.5.2 *Large Language Model Meta AI* (LlaMA)

Large Language Model Meta AI (LlaMA) adalah sekelompok *language model* berbasis *Transformers* yang dikembangkan oleh Meta untuk berbagai tugas NLP. Model LlaMA pertama kali dipublikasikan lewat *paper* yang berjudul "LlaMA*: Open and Efficient Foundation Language Models*" (Touvron et al, 2023). Melalui *paper* tersebut, dibuktikan bahwa model LlaMA mampu bersaing dengan model-model *state-of-the-art* seperti GPT-3, Chinchilla, dan PaLM, meskipun memiliki ukuran model yang jauh lebih kecil dan hanya dilatih menggunakan dataset yang tersedia untuk umum.

Pada 18 April 2024, Meta merilis sekelompok model LlaMA baru, yaitu *LlaMA-3* (Meta@AI, 2023). *LlaMA-3* menggunakan *tokenizer* dengan kosakata 128 ribu *token*, dan dilatih dengan sekuens 8.192 *token*. Terdapat dua jenis model *LlaMA-3*, yaitu model dasar (*pre-trained*) dan model *instruct*, masing-masing dengan dua ukuran berbeda: 8B dan 70B parameter. Jenis model LlaMA yang digunakan pada penelitian ini adalah *LlaMA-3-8B-Instruct*, yang telah dilatih dengan dataset berupa pasangan instruksi (*instruction*) dan jawaban.

2.1.3.5.3 *Mistral-7B*

Mistral-7B merupakan *language model* dengan 7 miliar parameter berbasis *Transformers* yang dikembangkan oleh Mistral. *Mistral-7B* dirancang untuk performa dan efisiensi yang unggul dan mampu melampaui kinerja model terbuka seperti *LlaMA-2-13B* dan *LlaMA-1-34B* di semua *benchmark* dalam hal penalaran, matematika, dan pembuatan kode. Model ini dikembangkan dengan memanfaatkan *grouped-query attention* (GQA) untuk inferensi yang lebih cepat, ditambah dengan *sliding window attention* (SWA) untuk secara efektif menangani urutan dengan 29ensiti yang berubah-ubah dengan biaya inferensi yang lebih rendah (Jiang et al, 2023).

Mistral merilis dua jenis model *Mistral-7B*, yaitu model dasar serta model *instruct* yang telah dilatih lebih lanjut supaya dapat mengikuti instruksi manusia. Versi *Mistral-7B* yang digunakan dalam penelitian ini adalah *Mistral-7B-Instruct-v0.3* yang dirilis pada Mei 2024. Dibandingkan model *Mistral-7B* sebelumnya, *Mistral-7B-Instruct v0.3* dilatih dengan jumlah kosakata yang lebih banyak, yaitu sebanyak 32.768 *token* (Mistral AI, 2024).

2.1.3.5.4 *Zephyr*

Zephyr adalah serangkaian *language model* yang dilatih untuk berperan sebagai asisten yang bermanfaat yang dikembangkan oleh Tim H4 Hugging Face. Model-model *Zephyr-7B* dilatih dari model *Mistral-7B-v0.1* dengan menggunakan proses *distilled supervised fine-tuning* (dSFT) yang dikombinasikan dengan *AI Feedback* (AIF) untuk meningkatkan ketepatan model. Dengan metode pelatihan

ini, *Zephyr-7B* mampu menetapkan standar baru dalam *benchmark chatting* untuk model berparameter 7B, serta berhasil melampaui kinerja *LlaMA-2-Chat-70B* yang merupakan salah satu model *chat* terbuka terbaik (Tunstall et al, 2023).

Pada penelitian ini, jenis model *Zephyr* yang digunakan adalah model *Zephyr-7B-β* (beta). Model *Zephyr-7B-β* merupakan model kedua dalam model seri *Zephyr*, yang dilatih menggunakan kombinasi dataset untuk umum dan dataset sintetis. Pada saat *Zephyr-7B-β* dirilis, model ini berhasil mencapai performa terbaik untuk model berparameter 7B dalam berbagai *benchmark* (HuggingFaceH4, 2023).

2.1.4 Metode *Retrieval Augmented Generation* (RAG)

2.1.4.1 Definisi *Retrieval Augmented Generation* (RAG)

Istilah *Retrieval-Augmented Generation* (RAG) diciptakan dalam *paper* "*Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*" (Lewis et al, 2020). *Paper* ini memperkenalkan pendekatan baru yang menggabungkan *pretrained language model* dengan mekanisme *retrieval* untuk meningkatkan kinerja pada tugas-tugas yang membutuhkan pengetahuan yang kompleks.

Dua unsur utama dari RAG adalah sistem pencarian (*retrieval system*) dan *neural language model*. Sistem pencarian mengekstrak informasi yang relevan dari korpus atau basis data yang besar dengan menggunakan teknik-teknik seperti pencocokan *keyword* dan pencarian semantik untuk menemukan data eksternal yang relevan secara kontekstual. Dalam RAG, informasi yang diambil diintegrasikan dengan pengetahuan *neural language model* yang telah dilatih

sebelumnya, sehingga menciptakan konteks yang lebih komprehensif. Integrasi ini meningkatkan relevansi dan keakuratan konten yang dihasilkan dengan memasukkan informasi yang spesifik dan terkini dari sumber eksternal (Islam, 2023).

Proses vektorisasi sangat penting dalam tahap pencarian sistem RAG, karena proses ini memungkinkan pencarian yang efisien dan menentukan data yang relevan dalam kumpulan data yang besar. Dalam model RAG, vektorisasi melibatkan pemrosesan *query* pengguna dan dokumen ke dalam representasi vektor menggunakan *embedding* yang telah dilatih sebelumnya seperti dengan model BERT atau RoBERTa, yang menangkap makna semantik. Kemiripan *query* dan dokumen diukur dengan menggunakan metrik seperti kesamaan *cosine*, kemudian dokumen dengan peringkat teratas berdasarkan skor ini diambil untuk meningkatkan jawaban dari *language model*.

Berikut ini adalah rumus *cosine* yang umum digunakan untuk mengukur kesamaan *query* dan dokumen:

$$Cosine = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\|\|\vec{B}\|} \quad (4)$$

Representasi vektor dari *query* dan dokumen yang dihasilkan oleh model dikalikan kemudian dibagi dengan perhitungan normalisasi dari kedua vektor. Perhitungan *cosine* akan menghasilkan ini menghasilkan nilai antara -1 dan 1, di mana 1 berarti vektor-vektor tersebut memiliki orientasi yang sama, 0 berarti ortogonal, dan -1 berarti berlawanan arah.

2.1.4.2 Metode RAG sebagai Alternatif *Fine-Tuning* LLM

(Islam, 2023) *Fine-tuning* merupakan proses pelatihan model *pretrained* lebih lanjut pada dataset spesifik untuk menyesuaikannya dengan tugas atau domain tertentu. Namun, untuk melakukan *fine-tuning* LLM seringkali dibutuhkan daya komputasi yang besar untuk pelatihan dan pengoperasian modelnya. Meskipun bukan metode *fine-tuning* konvensional, untuk tugas NLP seperti *question-answering* atau *text-generation*, metode RAG dapat menjadi metode alternatif untuk memberikan pengetahuan baru kepada LLM. Tabel 2.1 menguraikan perbandingan antara metode RAG dengan *fine-tuning* oleh (Islam, 2023).

Tabel 2.1 Tabel Perbandingan Metode RAG dan *Fine-Tuning*

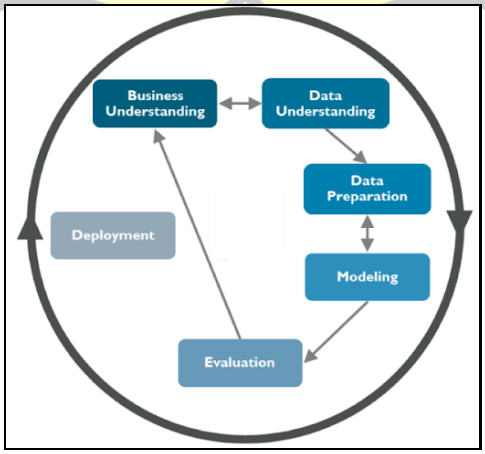
Aspek	**RAG**	**Fine-Tuning**
Pembaruan pengetahuan	Secara langsung memperbarui *knowledge base*, sehingga cocok untuk 32ensit dengan data yang dinamis. Tidak terlalu sering memerlukan pelatihan ulang model-model yang digunakan.	Tergantung pada data statis, memerlukan pelatihan ulang model untuk pembaruan pengetahuan, sehingga kurang cocok untuk 32ensit dengan data dan informasi yang sering berubah.
Pengetahuan eksternal	Mampu menggunakan sumber daya eksternal secara efektif, baik itu dengan basis data terstruktur maupun tidak terstruktur.	Menyatukan pengetahuan eksternal dari pretraining dengan LLM tetapi kurang efektif untuk sumber data yang dinamis.
Pengolahan data	Tidak banyak memerlukan pengolahan data.	Tergantung pada dataset berkualitas tinggi, dan keterbatasan dalam

		kumpulan data dapat memengaruhi kinerja.
Pengaturan model	Menawarkan lebih sedikit pengaturan dalam perilaku model.	Memungkinkan 33ensiti yang lebih besar terhadap perilaku model dan pengetahuan domain. Model dapat disesuaikan dengan kebutuhan spesifik.
Interpretabilitas	Menawarkan kemampuan interpretasi yang lebih tinggi karena penelusuran jawaban ke *knowledge base*.	Mekanisme internal modelnya seringkali tertutup sehingga keputusannya lebih sulit dipahami, menghasilkan interpreasi yang lebih rendah.
Sumber daya komputasi	Membutuhkan sumber daya untuk *retrieval* dan pemeliharaan basis data.	Membutuhkan sumber daya untuk menyiapkan dataset dan daya komputasi untuk *training* LLM.
Ketentuan *latency*	Melibatkan pengambilan data, sehingga dapat meningkatkan *latency*.	Biasanya menawarkan *latency* yang lebih rendah karena beroperasi tanpa memerlukan pengambilan data setelah pelatihan.
Mengurangi halusinasi	Tidak mudah mengalami halusinasi karena ketergantungan pada data factual pada *knowledge base*.	Dapat mengurangi halusinasi dengan pelatihan khusus domain tetapi masih rentan terhadap halusinasi

		dengan *input* yang tidak dikenal.
Masalah etis dan privasi	Permasalahan kerap muncul dari penyimpanan dan *retrieval* teks dari basis data eksternal.	Masalah etika dan privasi dapat muncul dari konten yang sensitive 34ensitive dalam data pelatihan.

2.1.5 CRISP-DM sebagai Tahap Merancang Model *Machine Learning*

Chatbot adalah salah satu jenis *task* pada NLP, dan NLP adalah salah satu bidang yang menggunakan *machine learning*. Tahap merancang model *machine learning* sama dengan tahap merancang sistem *data mining* yaitu dengan menggunakan metode *Cross-Industry Standard Process for Data Mining* (CRISP-DM). CRISP-DM merupakan salah satu metodologi yang umum digunakan dalam proses *data mining* dan *machine learning*. Pada akhir tahun 1990-an, industri terkemuka membuat metodologi untuk melakukan proyek *data mining*, yang kemudian menjadi praktik terbaik dan paling banyak digunakan di bidang ini (Cirillo, 2017:68). Gambar 2.7 menunjukkan keenam tahapan dari CRISP-DM.



Gambar 2.7 Tahap CRISP-DM (Cirillo, 2017:70)

(Cirillo, 2017) mendeskripsikan tahap-tahap CRISP-DM sebagai berikut:

1. *Business Understanding*

   Tahap ini melibatkan menjawab dua pertanyaan: Apa tujuan bisnis dan apa tujuan *data mining*. Selama fase ini, informasi dikumpulkan melalui wawancara dan tinjauan literatur untuk menentukan rencana proyek dan menetapkan tujuan *data mining* yang jelas.

2. *Data Understanding*

   Pada tahap ini, terdapat tiga hal utama yang harus dilakukan, yaitu: 1) pengumpulan data, mengumpulkan data-data yang diperlukan serta mendokumentasikan bagaimana data didapatkan; 2) pendeskripsian data, menjelaskan jenis file dan struktur data yang dikumpulkan; dan 3) eksplorasi data, memeriksa data yang dikumpulkan untuk mendapatkan pemahaman deskriptif.

3. *Data Preparation*

   Data preparation melibatkan persiapan data untuk proses pemodelan. Hal ini mencakup tugas-tugas seperti pembersihan data, validasi data, serta proses pengolahan data lainnya untuk memastikan data memenuhi persyaratan spesifik dari teknik pemodelan yang akan digunakan.

4. *Modeling*

   Tahap *modeling* bertujuan untuk memperoleh informasi dari data untuk menjawab pertanyaan-pertanyaan yang telah ditentukan pada tahap *business understanding*. Model yang digunakan dipilih berdasarkan pada jenis dan distribusi data yang dimiliki.

5. *Evaluation*

Pada tahap ini, dilakukan validasi dari hasil yang diperoleh dari tahap *modeling* sebelumnya. Ini melibatkan dua pertanyaan utama: apakah model memiliki kinerja yang memadai dan apakah model tersebut secara efektif menjawab pertanyaan-pertanyaan awal.

6. *Deployment*

Tahap *deployment* adalah tahap terakhir di mana model diimplementasikan ke dalam sistem produksi. Namun, jika tahap sebelumnya mengalami masalah, metode CRISP-DM memungkinkan adanya iterasi. Jika evaluasi menunjukkan kinerja yang tidak memadai, perlu diidentifikasi tahap yang bermasalah dan kembali ke tahap tersebut.

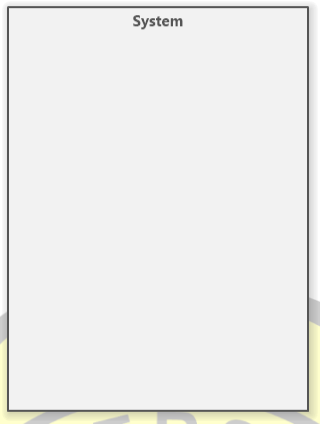
2.1.6 Pemodelan Sistem UML

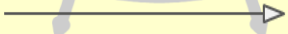
Unified Modeling Language (UML) adalah bahasa visual berupa diagram yang digunakan untuk menggambarkan, memvisualisasikan, membangun, dan mendokumentasikan komponen sistem perangkat lunak. UML membantu mendapatkan pemahaman yang lebih jelas terkait perangkat lunak, sistem, atau produk yang sedang dikembangkan (Sundaramoorthy, 2022:2).

2.1.6.1 *Use Case Diagram*

(Sundaramoorthy, 2022) menyatakan bahwa *use case diagram* berfokus pada identifikasi kebutuhan fungsional dari sistem yang akan dikembangkan. Tabel 2.1 menjelaskan rincian komponen-komponen dari *use case diagram*.

Tabel 2.2 Komponen-komponen dari *use case diagram*.

Orang.	Komponen	Simbol	Fungsi
1.	*System Boundary*		Merepresentasikan ruang lingkup sistem dan mencakup serangkaian fungsionalitas lengkap dari sistem.
2.	*Actor*		Merepresentasikan pengguna yang terlibat dengan sebuah sistem. *Actor* dapat mencakup individu, organisasi, atau sistem eksternal yang berinteraksi dengan aplikasi atau sistem.
3.	*Use Case*		Merepresentasikan fungsi bisnis dalam sebuah sistem. Fungsi-fungsi ini divisualisasikan untuk memastikan bahwa setiap proses bisnis terpisah dan terdefinisi dengan baik.

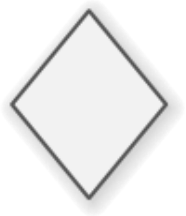
4.	*Association*		Merepresentasikan hubungan antara *actor* dan *use case*.
5.	*Include*		Merepresentasikan situasi di mana satu *use case* menyertakan fungsionalitas dari satu *use case* lainnya.
6.	*Extend*		Menunjukkan bahwa *use case* menambahkan perilaku tambahan terhadap fungsionalitas yang ada dari *use case* dasar.
7.	*Generalization*		Menunjukkan hubungan antara *use case* induk dan satu atau beberapa *use case* anak.

2.1.6.2 *Activity Diagram*

Activity diagram berfokus pada aktivitas berurutan dan paralel yang terlibat dalam setiap persyaratan fungsional dari sistem (Sundaramoorthy, 2022). Tabel 2.2 menjelaskan rincian komponen-komponen dari *activity diagram*.

Tabel 2.3 Komponen-komponen dari *activity diagram.*

Orang.	Komponen	Simbol	Fungsi
1.	*Initial State*		Merepresentasikan keadaan awal sistem.
2.	*Final State*		Merepresentasikan keadaan penghentian sistem.
3.	*Swimlane*		*Swimlane* terdiri dari dua partisi: satu mewakili entitas seperti *actor*, *use case*, atau kelas, dan yang lainnya berfokus pada aktivitas yang terlibat. Ada dua jenis *swimlane*: vertikal dan horizontal.
4.	*Action State*		Merepresentasikan suatu aktivitas atau proses.

5.	*Decision*		Umumnya memiliki satu *input* dan beberapa *output*. Setiap alur *output* dikaitkan dengan sebuah kondisi, dan jika kondisi tersebut terpenuhi, alur akan berlanjut. Output "*else*" dapat ditentukan untuk kasus-kasus di mana tidak ada kondisi lain yang terpenuhi.
6.	*Synchronization* (*Join/Fork*)		*Join* bertujuan untuk menggabungkan beberapa alur aktivitas. *Fork* bertujuan untuk membagi satu alur aktivitas menjadi dua atau lebih aktivitas yang bersamaan.

8.	*Flow Final*		Merepresentasikan penghentian alur yang tidak normal dalam sebuah aktivitas.
9.	*Transition*		Panah yang merepresentasikan pergerakan dari satu aktivitas ke aktivitas selanjutnya.

2.1.7 *Software* dan *Tools* Terkait

2.1.7.1 Bahasa Pemrograman *Python*

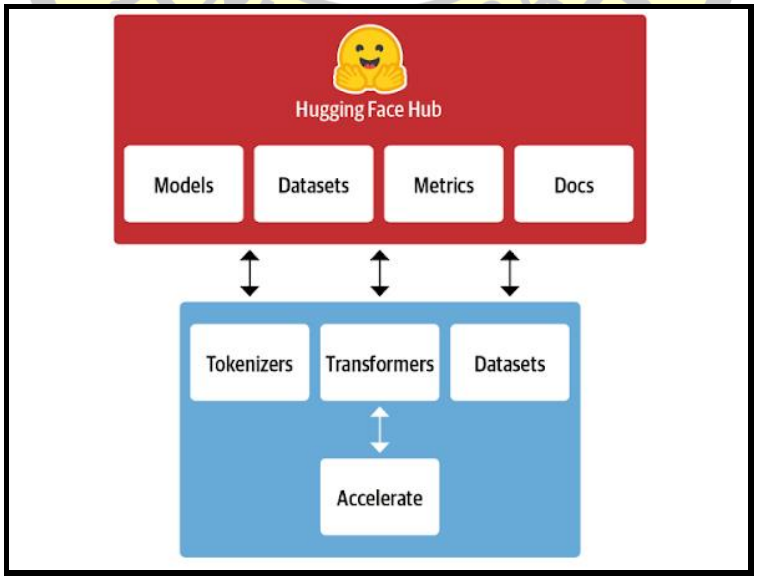
Python adalah bahasa pemrograman yang fungsional, prosedural, dan berorientasi objek, dikembangkan oleh Guido van Rossum pada akhir 1980-an. Bahasa ini digunakan dalam berbagai domain aplikasi seperti pengembangan perangkat lunak, pengembangan web, antarmuka pengguna grafis (GUI), serta aplikasi desktop pendidikan dan ilmiah. Bahasa pemrograman *Python* banyak diminati karena kesederhanaan, ukuran kecil, dan fleksibilitas-nya, serta terkenal karena kemudahan penggunaan dan ketangguhannya. *Python* memiliki banyak *library* bawaan dan modul pihak ketiga yang tersedia untuk berbagai tugas, sehingga memudahkan dan mempercepat pengembangan perangkat lunak (Anam et al, 2023).

2.1.7.2 *Streamlit*

Streamlit adalah *library Python* yang dirancang untuk membuat aplikasi web interaktif dengan cepat. *Streamlit* menawarkan *interface* yang intuitif dan mudah digunakan yang memungkinkan pengguna untuk membuat aplikasi web dengan kode yang minimal. *Streamlit* menyediakan *widget* dan komponen bawaan seperti *slider*, *dropdown*, dan input teks, yang memungkinkan pembuatan visualisasi dan *dashboard* interaktif. Selain itu, *Streamlit* juga mendukung pembaruan data secara *real-time*, sehingga cocok untuk aplikasi berbasis data yang dinamis (Moscato, 2024).

2.1.7.3 *Hugging Face Ecosystem*

Hugging Face ecosystem merujuk pada kumpulan alat, *library*, model, dan referensi terkait NLP yang dikembangkan oleh Hugging Face. Gambar 2.9 mengilustrasikan gambaran umum dari *Hugging Face ecosystem*.



Gambar 2.8 Gambaran umum *Hugging Face ecosystem* (Tunstall et al, 2022:15)

2.1.7.4 *LangChain*

LangChain adalah sebuah *framework* yang dirancang untuk meningkatkan kemampuan *Large Language Models* (LLMs) dengan menangani kekurangannya. LLM, meskipun canggih, memiliki kendala seperti panjang *context window* yang terbatas dan ketidakmampuan untuk berinteraksi dengan sumber data eksternal. *LangChain* mengurangi masalah ini melalui berbagai modul seperti *Model I/O*, *Retrieval*, *Chains*, *Memory*, *Agent*, dan *Callback*. Modul-modul ini secara kolektif memperluas fungsionalitas LLM, membuatnya lebih fleksibel dan dapat beradaptasi dengan berbagai tugas dan sumber data yang lebih luas (Vasilev, 2023).

2.1.7.5 *MongoDB*

MongoDB adalah basis data *NoSQL*, non-relasional yang umum digunakan yang dikenal dengan keserbagunaan dan fitur-fiturnya yang tangguh. *MongoDB* berfungsi sebagai penyimpan *key-value* dan database JSON, sehingga cocok digunakan pada aplikasi modern. *MongoDB* menawarkan fitur-fitur bawaan seperti *machine learning* dan kemampuan AI, *streaming*, fungsi-fungsi *serverless*, sinkronisasi perangkat, dan pencarian teks utuh. Meskipun non-relasional, *MongoDB* dapat secara efektif menangani data relasional, dan menyediakan sumber daya yang luas untuk mempelajari cara memodelkan dan mengelola data ini (Aleksendrić et al, 2024).

2.2 Kajian Literatur

Tabel di bawah ini menyajikan ringkasan referensi penelitian sebelumnya yang terkait dengan pembuatan *Chatbot* serta penggunaan metode *Transformers*.

Tabel 2.4 *Paper* penelitian terkait.

Orang.	**Penulis**	**Tahun**	**Judul**	**Metode**	**Publikasi**
1.	Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin	2017	Attention Is All You Need	*Transformers*	31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA
	Hasil: *Paper* ini memperkenalkan model *Transformers*, sebuah arsitektur jaringan saraf baru yang hanya didasarkan pada *self-attention mechanism*, tanpa lapisan berulang (*recurrent*) atau konvolusi (*convolution*). Model *Transformers* mencapai kinerja terbaik pada berbagai tugas penerjemahan, melampaui model sebelumnya seperti *recurrent neural network* (RNN) dan *convolutional neural network* (CNN) dalam hal akurasi dan efisiensi				

	komputasi. Selain itu, arsitektur *Transformers* memfasilitasi paralelisasi, sehingga sangat fleksibel dan cocok untuk aplikasi skala besar.				
	Keterbatasan penelitian: *Transformers* dapat menghabiskan *computation cost* yang tinggi untuk melatih dan menggunakannya. Selain itu, *Transformers* memiliki panjang yang tetap, tidak seperti RNN yang memiliki panjang berubah-ubah.				
2.	Guntoro, Loneli Costaner, Lisnawita	2020	Aplikasi Chatbot untuk Layanan Informasi dan Akademik Kampus Berbasis Artificial Intelligence Markup Language (AIML)	AIML	Digital Zone: Jurnal Teknologi Informasi & Komunikasi, Volume 11, Nomor 2 November 2020:291-300 (SINTA 3)
	Hasil: Implementasi dan pengujian aplikasi *Chatbot* menggunakan metode *Artificial Intelligence Markup Language* (AIML) menunjukkan hasil yang positif. Aplikasi ini mampu menjawab pertanyaan sesuai dengan pengetahuan yang telah diberikan, serta memungkinkan percakapan online melalui browser. *Chatbot* dapat memberikan informasi tentang pendaftaran mahasiswa di Universitas Lancang Kuning, termasuk alamat kampus, syarat pendaftaran, langkah-langkah pendaftaran, program studi, jalur kuliah, jadwal kuliah, dan proses pendaftaran. Pengujian aplikasi menggunakan metode *whitebox* dan *blackbox* mencapai tingkat				

	keberhasilan 100%, sementara pengujian UAT mencapai 95%, menunjukkan bahwa aplikasi dapat memberikan jawaban yang memadai berdasarkan pengetahuan yang telah diberikan sebelumnya.				
	Keterbatasan penelitian: Metode AIML didasarkan pada metode *rule-based* di mana *developer* secara manual mendefinisikan aturan tentang bagaimana *Chatbot* harus menanggapi *input* tertentu. Ini bisa sangat kaku karena AIML sangat bergantung pada pencocokan pola dan tidak menyediakan kemampuan NLP yang mumpuni. Selain itu, dibutuhkan upaya manual yang cukup besar untuk membuat dan memelihara rangkaian aturan yang komprehensif. *Chatbot* AIML juga umumnya tidak memiliki pemahaman konteks, *Chatbot* tidak mengingat interaksi sebelumnya atau memahami konteks percakapan.				
3.	Wiwin Surwaningsih, Raka Aditya P., Fadhil Yusuf R.	2022	RoBERTa: language modelling in building Indonesian question-answering systems	ALBERT, RoBERTa, ELECTRA	TELKOMNIKA Telecommunication Computing Electronics and Control Vol. 20, Orang. 6, December 2022, pp. 1248~1255 (SINTA 1)
	Hasil: Kinerja model ALBERT, RoBERTa, dan ELECTRA menggunakan bahasa Orang, Malaysia, dan Esperanto dievaluasi. Untuk hasil terbaik dalam membangun QAS Orang, model bahasa yang digunakan adalah RoBERTa yang menghasilkan performa EM 84,6% dan F1 86,2%. Hasil ini juga dapat terus meningkat hingga 100% jika model dilatih sepenuhnya. Namun, dalam implementasi kasus dunia nyata, penggunaan RoBERTa saja belum cukup untuk mendapatkan jawaban yang tepat. Hal ini karena RoBERTa				

	dilatih untuk menemukan jawaban dari konteks yang sesuai dengan pertanyaan.
	Keterbatasan penelitian: Model RoBERTa pada penelitian ini belum dilatih sepenuhnya pada dataset yang lebih spesifik, sehingga kinerjanya belum cukup dalam implementasi kasus dunia nyata.

TEKNOLOGI INFORMASI

UNIVERSITAS DARMA PERSADA