BAB II
TINJAUAN PUSTAKA

2.1 *Chatbot*

Chatbot adalah aplikasi komputer yang dirancang untuk berinteraksi dengan pengguna melalui pesan teks atau suara. Dengan memanfaatkan kecerdasan buatan dan pemrosesan bahasa alami, *chatbot* mampu memberikan respons yang cerdas terhadap pertanyaan yang diajukan oleh pengguna (Rianto et al, 2024). Teknologi ini memungkinkan *chatbot* untuk memahami konteks percakapan dan memberikan jawaban yang sesuai, sehingga berfungsi sebagai asisten digital yang dapat membantu berbagai kebutuhan informasi (Guntoro et al, 2020).

Chatbot dibangun untuk menyediakan layanan informasi, dengan fokus pada topik tertentu yang relevan dengan kebutuhan pengguna. Berbagai *chatbot* telah dikembangkan untuk menangani masalah spesifik, baik untuk kebutuhan pribadi maupun bisnis. Dalam pengembangan *chatbot*, model pengetahuan yang terintegrasi digunakan untuk menjawab pertanyaan sesuai dengan konteks yang telah ditentukan sebelumnya. Secara umum, *chatbot* terdiri dari tiga komponen utama yang saling berinteraksi untuk membentuk sistem yang efektif:

a. *User Interface* (UI): Antarmuka pengguna yang menjadi penghubung antara *chatbot* dan pengguna. UI berperan penting dalam memberikan pengalaman interaktif yang baik, memudahkan pengguna dalam berinteraksi dengan *chatbot* melalui aplikasi pesan berbasis teks.

b. *Artificial Intelligence* (AI): AI memberi *chatbot* untuk memahami setiap interaksi dengan pengguna, menjadikannya lebih dari sekadar aplikasi statis. AI memberikan kecerdasan pada *chatbot* sehingga aplikasi dapat belajar dan menyesuaikan respons berdasarkan input dari pengguna.

c. Integrasi: Dengan mengintegrasikan *chatbot* ke sistem lain, fitur *chatbot* dapat lebih diperluas, menyediakan informasi yang lebih komprehensif. Integrasi ini memperkaya kemampuan *chatbot*, seperti dalam proyek ini

yang mengimplementasikan *chatbot* dengan layanan berbasis cloud untuk memberikan kemudahan akses dan skalabilitas.

Chatbot sangat efektif dalam menyelesaikan permasalahan yang berfokus pada komunikasi dengan pengguna. Dalam konteks bisnis, *chatbot* dapat meningkatkan pelayanan pelanggan dan memperbaiki pengalaman berinteraksi dengan perusahaan (Rianto et al., 2024). Dengan kemampuan untuk menangani tugas yang terprediksi dan spesifik, *chatbot* menjadi solusi yang efisien di dunia bisnis yang terus berkembang pesat. Melalui penggunaan *chatbot*, bisnis dapat mempermudah pelanggan untuk mengakses informasi dan berinteraksi tanpa hambatan waktu.

2.2 Python

Python merupakan bahasa pemrograman tingkat tinggi yang diciptakan oleh Guido van Rossum dan dirilis pertama kali pada tahun 1991. Bahasa ini dikenal karena sintaksisnya yang sederhana dan keterbacaan kode yang tinggi, sehingga memudahkan programmer dalam menulis dan memahami kode. Python mendukung berbagai paradigma pemrograman, termasuk pemrograman berorientasi objek, prosedural, dan fungsional (Alfarizi et al., 2023). Kelebihan utama dari Python adalah kemampuannya untuk digunakan dalam berbagai aplikasi, mulai dari pengembangan web hingga analisis data dan kecerdasan buatan.

Bahasa pemrograman Python memiliki beberapa fitur bagi pengembang perangkat lunak. Berikut ini merupakan fitur – fitur yang terdapat pada bahasa pemrograman Python :

1. Multi Paradigm Design: Python mendukung beberapa paradigma pemrograman, termasuk *Object Oriented Programming* (OOP), imperatif, dan pemrograman fungsional, memberikan fleksibilitas kepada pengembang dalam memilih pendekatan yang sesuai untuk proyek mereka.

2. Open Source: Python adalah bahasa pemrograman open source, artinya perangkat lunak ini dapat digunakan, dimodifikasi, dan didistribusikan secara bebas oleh siapa saja.

3. Simplicity: Sintaks Python yang sederhana dan mudah dipahami membuatnya lebih mudah untuk dibaca dan ditulis dibandingkan dengan bahasa pemrograman lain seperti C++ atau Java.

4. Library Support: Python memiliki banyak pustaka standar yang memudahkan pengembang karena tidak perlu menulis seluruh kode dari awal. Pustaka-pustaka ini telah teruji dan digunakan secara luas oleh komunitas pengembang.

5. Portability: Skrip bahasa pemrograman Python dapat dijalankan di sistem operasi seperti Windows, Linux, dan Mac OS tanpa perlu modifikasi besar.

6. Extendable: Python memungkinkan integrasi dengan bahasa pemrograman lain, sehingga pengguna dapat menambahkan atau menggabungkan kode dari bahasa lain ke dalam aplikasi Python mereka.

7. Scalability: Python dirancang untuk dapat menangani masalah dunia nyata baik dalam skala kecil maupun besar, menjadikannya pilihan yang baik untuk berbagai aplikasi.

2.3 Natural Language Processing

Natural Language Processing (NLP) adalah bidang penelitian dan aplikasi yang mengkaji bagaimana komputer dapat digunakan untuk memahami dan memanipulasi teks atau ucapan dalam bahasa alami untuk tujuan yang bermanfaat. Para peneliti di bidang NLP berupaya untuk mengembangkan algoritma dan teknik agar komputer dapat memahami, menginterpretasikan, dan memproses bahasa manusia secara efisien. Hal ini bertujuan untuk menciptakan sistem yang dapat menyelesaikan berbagai tugas berbasis bahasa alami, seperti analisis sentimen, penerjemahan otomatis, dan pengenalan suara, dengan cara yang serupa dengan pemahaman manusia terhadap bahasa (Oktavia et al., 2024).

Menurut Puspitasari et al. (2024) terdapat beberapa area fokus utama dalam penelitian NLP, antara lain:

1. Question Answering System (Sistem Tanya Jawab): Memfasilitasi pengguna untuk memberikan pertanyaan yang akan dijawab secara otomatis oleh komputer. Dengan sistem ini, pengguna dapat menggunakan berbagai macam bahasa alami seperti bahasa Indonesia, Inggris, Mandarin, dan lain-lain.
2. Summarization: Memberikan kemampuan pada komputer untuk membuat ringkasan dari sekumpulan teks, baik dari dokumen atau email, dan menghasilkan ringkasan yang sesuai dengan isi konten dokumen tersebut.
3. Machine Translation (Mesin Penerjemah): Memberi kemampuan pada komputer untuk memahami bahasa alami dan menerjemahkan masukan pengguna ke bahasa alami lainnya. Salah satu contoh produk yang sangat populer dari penerapan area penelitian ini adalah Google Translate.

2.4 *Large Language Model*

Large Language Model (LLM) merupakan model kecerdasan buatan yang dilatih pada dataset teks besar untuk memahami dan menghasilkan bahasa alami dengan memanfaatkan arsitektur transformator. Arsitektur ini membuat LLM lebih efektif dalam menangkap hubungan antar kata dan konteks dalam teks dibandingkan model sebelumnya, seperti jaringan saraf berulang. Mekanisme perhatian dalam transformator berfungsi untuk memfokuskan perhatian pada informasi penting dalam kalimat panjang atau kompleks, meningkatkan akurasi pemahaman makna. Contoh LLM yang populer, yaitu GPT-3 yang dikenal karena kemampuannya menghasilkan teks panjang dengan koherensi tinggi dan BERT yang unggul dalam memahami konteks dua arah pada teks. Model-model ini telah berhasil diaplikasikan dalam berbagai tugas, seperti penerjemahan otomatis, analisis sentimen, dan pembuatan konten (Rosa, 2024).

LLM memiliki kemampuan pemahaman bahasa yang sangat baik sehingga mampu untuk menghasilkan teks yang koheren, relevan, dan menyerupai tulisan manusia. Salah satu aplikasi menarik dari LLM adalah dalam pengembangan sistem tanya jawab berbasis konteks. Dalam penelitian yang dilakukan oleh Aprilia et al. (2023) mengungkapkan bahwa penggunaan LLM mampu mencapai akurasi sebesar 82% dalam memberikan jawaban yang relevan terhadap pertanyaan pengguna.

Temuan ini potensi besar LLM dalam mendukung akses informasi secara luas dan memperdalam pemahaman masyarakat terhadap berbagai bidang pengetahuan.

Namun, meskipun memiliki banyak keunggulan, LLM tidak bebas dari tantangan. Bias dalam data pelatihan dapat memengaruhi kualitas dan netralitas *output*, sementara kebutuhan sumber daya komputasi yang tinggi menjadi kendala dalam implementasi skala besar. Oleh karena itu, penelitian lebih lanjut diperlukan untuk mengatasi masalah ini, seperti pengembangan metode pelatihan yang lebih efisien dan pengurangan bias dalam data (Rosa, 2024).

2.5 Prompt Engineering

Prompt merupakan serangkaian instruksi yang diberikan kepada model bahasa besar (LLM) dengan tujuan menyesuaikan atau meningkatkan performa serta fungsionalitasnya, sebagaimana dijelaskan oleh Schmidt et al. (2023). Prompt ini memainkan peran penting dalam memengaruhi jalannya interaksi berikutnya dengan model dan bentuk keluaran yang dihasilkan, dengan cara menetapkan aturan-aturan dan pedoman spesifik yang membingkai konteks percakapan. Prompt juga memberikan arahan tentang informasi apa yang relevan serta bagaimana format dan isi dari respons yang diharapkan dari LLM.

Sebagai ilustrasi, sebuah prompt dapat mengatur agar model hanya menghasilkan kode yang sesuai dengan gaya pengkodean atau paradigma pemrograman tertentu. Contohnya, prompt dapat memandu LLM untuk menandai kata kunci atau frasa penting dalam sebuah dokumen yang dibuat, kemudian menyajikan informasi tambahan terkait kata kunci tersebut. Dengan menyediakan panduan semacam ini, prompt memungkinkan keluaran yang lebih terstruktur dan bernuansa, yang berfungsi untuk memfasilitasi berbagai tugas dalam rekayasa perangkat lunak (White dkk., 2023).

Selain mengatur jenis keluaran atau memfilter masukan, prompt juga dapat dirancang untuk memperluas cakupan kemampuan LLM. Dengan perancangan yang tepat, prompt dapat menciptakan bentuk interaksi yang sepenuhnya baru. Misalnya, LLM dapat diprogram untuk menghasilkan kuis yang berkaitan dengan konsep atau alat dalam rekayasa perangkat lunak, atau bahkan mensimulasikan

lingkungan terminal Linux. Lebih lanjut, prompt memiliki potensi adaptif, yakni dapat menghasilkan atau merekomendasikan prompt tambahan untuk mengumpulkan informasi lebih banyak atau menciptakan artefak pendukung. Fitur-fitur canggih ini menunjukkan bahwa desain prompt yang efektif sangatlah penting untuk menghasilkan keluaran yang lebih kompleks dan bernilai tinggi, melebihi sekadar pembuatan teks atau kode sederhana (Schmidt et al., 2023).

Menurut Bozkurt & Sharma (2024)., terdapat beberapa rekomendasi penting untuk membuat prompt yang lebih spesifik dan efektif dalam berinteraksi dengan model bahasa besar (LLM), yaitu:

1. Tingkatkan Spesifikasi Prompt: Semakin rinci dan terperinci prompt yang dibuat, semakin besar peluang untuk mendapatkan jawaban yang akurat dan sesuai dengan kebutuhan. Penggunaan deskripsi yang jelas dan detail akan membantu LLM memahami dengan lebih baik konteks dan tujuan pertanyaan.
2. Deskripsikan Pengaturan dan Konteks: Memberikan latar belakang atau konteks yang relevan sangat penting untuk membantu LLM menghasilkan tanggapan yang sesuai. Menyediakan informasi tambahan tentang situasi atau kebutuhan pengguna akan meningkatkan kualitas respons.
3. Eksperimen dengan Gaya Prompt: Mencoba berbagai format atau gaya penulisan prompt dapat membantu menemukan pendekatan yang paling efektif untuk mencapai hasil yang diinginkan.
4. Tetapkan Tujuan Prompt: Sebelum membuat prompt, penting untuk memahami tujuan akhir dari pertanyaan atau instruksi yang diajukan. Hal ini akan memandu struktur prompt dan membantu menjaga fokus.
5. Gunakan Peran dalam Prompt: Mintalah LLM untuk memainkan peran tertentu, bertujuan menciptakan respons yang lebih kontekstual dan relevan dengan situasi tertentu.
6. Iterasi dan Perbaikan: Tinjau respons yang dihasilkan, lakukan perbaikan pada prompt jika diperlukan, dan mintalah LLM untuk memperbarui output berdasarkan tanggapan sebelumnya. Pendekatan iteratif ini meningkatkan akurasi dan relevansi jawaban.

7. Manfaatkan Threads: Untuk percakapan yang lebih kompleks, gunakan utas (threads) agar konteks dan alur komunikasi tetap terjaga.
8. Ajukan Pertanyaan Terbuka: Pertanyaan terbuka sering menghasilkan jawaban lebih lengkap dan komprehensif dibandingkan pertanyaan tertutup.
9. Minta Contoh Spesifik: Jika memerlukan ilustrasi atau penerapan tertentu, secara eksplisit minta contoh spesifik untuk meningkatkan kejelasan dan kegunaan respons.
10. Tentukan Proses atau Waktu: Ketika meminta penjelasan proses atau jadwal, nyatakan batas waktu atau kerangka proses secara eksplisit dalam prompt untuk memandu tanggapan yang sesuai.
11. Tetapkan Harapan yang Realistis: Pastikan untuk memiliki ekspektasi yang sesuai dengan kemampuan model. Menyadari batasan teknologi membantu menciptakan pengalaman yang lebih produktif dan memuaskan.

2.6 Streamlit

Streamlit adalah sebuah framework berbasis Python yang bersifat *open-source*, dirancang untuk memudahkan dalam membangun aplikasi web interaktif di bidang sains data dan machine learning. Salah satu hal menarik dari framework ini adalah pengguna tidak perlu memiliki pengetahuan mendalam tentang teknologi web development, seperti CSS, HTML, atau JavaScript. Untuk menggunakan Streamlit, cukup dengan pengetahuan dasar mengenai bahasa Python (Yulianto, 2021). Dengan Streamlit, pengguna dapat dengan cepat mengubah *script* Python menjadi aplikasi web yang menarik tanpa harus menghabiskan waktu untuk mempelajari kerumitan pengembangan web.

2.7 Amazon Web Services (AWS)

Amazon Web Services (AWS) adalah platform layanan cloud computing yang disediakan oleh Amazon menawarkan berbagai layanan dan produk yang dapat digunakan untuk kebutuhan komputasi, penyimpanan, dan pengelolaan data. AWS menjadi salah satu penyedia layanan cloud terbesar secara global dan terus berkembang seiring meningkatnya kebutuhan solusi berbasis cloud di berbagai sektor, termasuk pendidikan, bisnis, dan teknologi informasi (Mubarok, 2022).

Salah satu keunggulan AWS adalah fleksibilitas dan skalabilitasnya yang memberi pengguna akses ke sumber daya sesuai kebutuhan. AWS menyediakan layanan seperti Elastic Compute Cloud (EC2), Simple Storage Service (S3), dan Relational Database Service (RDS) yang dirancang untuk mendukung pengembangan aplikasi dan website secara efisien (Saedudin & Almaarif, 2020). Dengan model pembayaran "pay-as-you-go", pengguna hanya dikenakan biaya sesuai layanan yang digunakan, sehingga membantu mengurangi biaya operasional (Setiawan & Fajriyah, 2022).

2.7.1 *High Availability*

High Availability (HA) adalah sistem yang dirancang untuk terus beroperasi tanpa gangguan, meskipun terjadi kegagalan pada salah satu komponen. Di AWS, HA dicapai melalui redundansi, pemantauan otomatis, dan mekanisme failover. Pendekatan umum mencakup penggunaan *cluster* server yang tersebar di beberapa *Availability Zone* untuk mencegah Single Point of Failure. AWS mendukung HA melalui layanan seperti Application Load Balancing dan Amazon EC2 yang mendukung pembangunan aplikasi dengan tingkat ketersediaan hingga 99,9% (Guntoro et al., 2020).

Implementasi *High Availability* (HA) tidak hanya melibatkan konfigurasi komponen sistem untuk memastikan redundansi dan toleransi terhadap kegagalan, tetapi juga memerlukan pengujian dan pemantauan berkala guna menjaga kinerja yang optimal. Pengujian dilakukan untuk memastikan bahwa setiap komponen, seperti server, load balancer, dan penyimpanan data, dapat menangani gangguan tanpa mengorbankan ketersediaan layanan. Dalam penelitian oleh Supriyadi et al. (2023), disebutkan bahwa penerapan HA menggunakan layanan AWS EC2 berhasil meningkatkan ketersediaan layanan hingga 99,8%, yang mencerminkan efektivitas teknologi cloud dalam mempertahankan kontinuitas operasional. Keberhasilan ini menegaskan pentingnya strategi HA dalam menghadapi beban kerja yang dinamis dan kebutuhan uptime yang tinggi pada aplikasi modern.

2.7.2 *Auto Scaling*

Auto Scaling adalah fitur yang secara otomatis menyesuaikan jumlah instance server sesuai dengan kebutuhan beban kerja saat itu. Fitur ini berfungsi untuk menambah kapasitas komputasi ketika permintaan pengguna meningkat dan menguranginya saat beban kerja menurun, tanpa memerlukan intervensi manual (Setiawan & Fajriyah, 2022). Kemampuan ini sangat penting dalam konteks aplikasi web yang menghadapi fluktuasi lalu lintas, seperti platform e-commerce atau aplikasi media sosial, karena memastikan kinerja tetap optimal dan biaya operasional lebih efisien.

Penelitian oleh Ramadhani et al. (2023) menunjukkan bahwa penerapan *Auto Scaling* pada web server berbasis AWS dapat meningkatkan kinerja sistem secara signifikan. Dalam studi tersebut, pengujian dilakukan dengan menggunakan beberapa instance server dan menunjukkan bahwa sistem dapat menangani hingga 60 permintaan per detik dengan ketersediaan tinggi saat menggunakan metode *Auto Scaling*. Hasil ini menegaskan pentingnya *Auto Scaling* dalam mengoptimalkan sumber daya dan memastikan bahwa aplikasi tetap responsif di bawah beban kerja yang tinggi.