BAB II

TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Large Language Models(LLMs) dan Arsitektur Transformer

Large Language Models atau LLM merupakan salah satu terobosan yang signifikan dalam ranah Deep Learning karena kemampuan yang dapat memodelkan probabilitas bahasa manusia, terobosan itu juga bisa menirukan bagaimana cara manusia membuat kalimat dan mengerti maksud dari kalimat tersebut. Model tersebut berbeda dengan pendekatan statistik lama, di mana LLM dilatih pada data yang sangat besar untuk memahami gaya bahasa dan pola pengetahuan umum.

Fondasi utama dari model modern seperti Llama 3.2 (yang digunakan dalam penelitian ini) adalah arsitektur Transformer, yang diperkenalkan oleh Vaswani *et al*[1]. Keunggulan arsitektur ini menggantikan metode sekuensial lama seperti RNN, menjadikannya standar baru dalam tugas pemrosesan bahasa alami.

Komponen utama Transformer adalah mekanisme *self-attention*. Mekanisme ini krusial bagi penelitian ini karena memungkinkan model memproses input pertanyaan mahasiswa secara utuh (paralel), bukan kata per kata. Hal ini memungkinkan sistem untuk menangkap konteks kalimat yang panjang—seperti pertanyaan akademik yang kompleks—dengan lebih akurat melalui perhitungan *importance weighting*. Dengan adanya *long-range dependencies* inilah kemampuan chatbot yang dapat menjamin jawaban tetap relevan walaupun pertanyaan pengguna berstruktur rumit.

2.2 Llama 3.2

Llama 3.2 adalah model open-source LLM yang dibuat dan dikembangkan oleh Meta. LLama model ini berbeda dengan generasi sebelumnya, dikarenakan Llama 3.2 mengeluarkan model varian yang lebih kecil dan ringkas(seperti 1B dan 3B) yang dirancang untuk skenario perangkat lokal dan efisiensi yang tinggi[3]. Meskipun memiliki jumlah parameter yang lebih kecil dibandingkan model raksasa

lainnya, model ini telah melalui proses pelatihan *instruction-tuning* yang memungkinkannya memiliki kemampuan penalaran (*reasoning*) dan pemahaman instruksi yang sangat kompetitif.

Dalam konteks penelitian ini, fokus diletakkan pada varian Llama 3.2-3B. Varian ini dinilai memiliki keseimbangan arsitektur yang paling optimal untuk eksperimen akademis dengan sumber daya terbatas. Ada Studi yang menunjukkan bahwa Llama 3.2 3B mampu melakukan tugas penalaran secara kompleks seperti *chain-of thought* pada domain medis atau spesifik lainnya, dengan menggunakan metode *resource-efficient fine-tuning* tanpa kehilangan akurasi secara signifikan[13]. Karakteristik efisiensi inilah yang membuat penulis memilih Llama 3.2 untuk menjadi model yang ideal untuk diintegrasikan ke dalam sistem asisten virtual yang berjalan pada lingkungan lokal.

2.3 Tantangan Full Fine-Tuning

Adaptasi *Large Language Model* (LLM) dilakukan melalui proses yang dikenal sebagai *Full Fine-Tuning*. Metode ini mengubah seluruh parameter atau bobot model yang telah dilatih, lalu memperbarui dan melatih ulang menggunakan dataset baru agar sesuai dengan tugas spesifik. Walaupun banyak bukti yang menunjukkan pendekatan ini mendapatkan bukti efektif untuk mendapatkan akurasi yang tinggi, jika diterapkan pada model yang berskala besar maka akan menghadirkan hambatan teknis yang signifikan.

Hambatan utama terletak pada efisiensi sumber daya yang kurang optimal. Sebagaimana dijelaskan dalam survei komprehensif oleh Lialin *et al.*, melakukan pelatihan penuh pada miliaran parameter menuntut kapasitas memori GPU (VRAM) yang sangat besar serta waktu komputasi yang lama[4]. Hal ini menjadikan *full fine-tuning* sangat sulit diakses oleh peneliti atau pengembang yang hanya memiliki perangkat keras terbatas.

Selain masalah komputasi, Hu *et al.* juga mengidentifikasi masalah pada sisi penyimpanan dan *deployment*[5]. Saat melakukan skema Full fine-tuning, setiap kali model tersebut dilatih untuk tugas yang berbeda, maka sistem harus menyimpan salinan penuh dari model tersebut, hal itulah yang membutuhkan penyimpanan pada

perangkat lokal yang cukup besar. Inefisiensi inilah yang menyebabkan kendala besar di saat model harus didistribusikan atau dijalankan di lingkungan dengan penyimpanan yang terbatas.

Pendekatan ini juga rentan terhadap fenomena *catastrophic forgetting.* Karena seluruh bobot diubah secara drastis untuk menyesuaikan data baru dan spesifik tadi, model juga beresiko untuk melupakan pengetahuan umum atau kemampuan bahasa dasar yang telah dilatih selama pra-pelatihan.

2.4 Parameter-Efficient Fine-Tuning(PEFT)

Dengan adanya masalah diatas maka muncullah solusi atas tingginya kebutuhan sumber daya pada metode pelatihan konvensional, pendekatan alternatif ini dikenal sebagai Parameter-Efficient Fine-Tuning (PEFT). PEFT menawarkan strategi yang jauh lebih hemat dengan melakukan teknik membekukan sebagian besar parameter model pratelatih dan hanya melatih sejumlah kecil parameter tambahan, hal ini yang membedakan antara full fine-tuning dengan PEFT. Karena tambahan parameter seringkali kurang dari 1% dari total parameter.

Konsep ini dijelaskan secara mendalam oleh Lialin *et al.* mereka menyatakan bahwa PEFT dapat menjembatani celah antara model raksasa dan keterbatasan perangkat keras, mereka telah membuktikan hal itu dengan survei. Hanya dengan memodifikasi sebagian kecil bobot, performa model dalam tugas spesifik dapat ditingkatkan secara signifikan tanpa adanya memori GPU yang besar [4].

Relevansi PEFT semakin terasa dalam skenario komputasi praktis di lingkungan terbatas (*low-resource environments*). Singh *et al.* menekankan bahwa penerapan teknik ini memungkinkan perangkat dengan spesifikasi standar melakukan *deployment* atau adaptasi model LLM secara langsung, yang sebelumnya dianggap mustahil jika menggunakan metode pelatihan penuh[21]. Hal ini menjadikan PEFT sebagai fondasi teknis yang memungkinkan pengembangan asisten AI lokal dalam penelitian ini, tanpa adanya perangkat dengan spesifikasi server atau cloud.

2.5 Low-Rank Adaptation(LoRA)

Low-Rank Adaptation(LoRA) menjadi teknik yang paling menonjol dan banyak diadopsi secara luas, dibandingkan dengan pendekatan efisiensi yang ada. Memiliki konsep dasar yang dibangun di atas sebuah temuan penting yang disebut hipotesis peringkat intrinsik rendah(low intrinsic rank hypothesis).

Hu *et al.*, mereka yang memperkenalkan LoRA dalam penelitian fundamental, mengusulkan bahwa model bahasa memiliki miliaran parameter, perubahan bobot yang diperlukan untuk beradaptasi pada tugas baru sebenarnya sedikit. Dalam proses adaptasi tersebut mereka dapat informasi penting yang dapat direpresentasikan secara efisien dalam ruang dimensi yang jauh lebih rendah[5]. Yang berarti, kita tidak perlu melatih ulang matriks raksasa penuh, melainkan hanya cukup melatih ringkasan dari perubahannya saja.

Dalam teori matematis, LoRA tidak akan mengubah bobot asli model (W_0) secara langsung. Sebaliknya, metode ini hanya menyuntikkan matriks pembaruan (W_0) yang didekomposisi menjadi dua matriks berukuran yang jauh lebih kecil, yaitu matriks $\boldsymbol{A}$ dan matriks $\boldsymbol{B}$. Persamaan pembaruannya dapat dituliskan sebagai berikut:

$$W = W_0 + \Delta W = W_0 + \boldsymbol{B}\,\boldsymbol{A}$$

Di mana matriks $\boldsymbol{A}$ dan $\boldsymbol{B}$ memiliki dimensi rank (rr) yang sangat kecil, jauh lebih kecil dibandingkan dengan dimensi model aslinya. Selama proses pelatihan, bobot asli WW_0 dibekukan (*frozen*) dan tidak disentuh sama sekali, sehingga hanya matriks kecil $\boldsymbol{A}$ dan $\boldsymbol{B}$ yang diperbarui.

Keunggulan praktis utama dari pendekatan ini adalah efisiensi tanpa mengorbankan kecepatan saat penggunaan (*inference*). Hu *et al.* menekankan bahwa setelah pelatihan selesai, matriks adaptasi ($\boldsymbol{BA}$) dapat digabungkan kembali (*merged*) ke dalam bobot asli WW_0. Karena itu model dari hasil *fine-tuning* tidak mengalami latensi tambahan saat dijalankan, karena strukturnya kembali menjadi satu kesatuan model yang utuh[5].

2.6 Ollama dan Manajemen Model Lokal

Penelitian ini menjalankan Large Language Models(LLM) pada lingkungan lokal secara efisien. Ollama adalah kerangka kerja *open-source* yang dirancang khusus untuk menyederhanakan proses eksekusi dan pengelolaan model bahsa besar di perangkat pribadi, tanpa bergantung dengan layanan cloud[6].

Ollama memiliki komponen inti yang disebut dengan *Modelfile.* Komponen ini berfungsi sebagai berkas cetak biru yang mendefinisikan bagaimana sebuah model itu dibangun, di dalamnya pengembang dapat mengatur parameter sistem maupun pesan pembuka bahkan sampai format percakapan.

Fitur yang paling krusial dalam penelitian ini adalah dukungan terhadap instruksi ADAPTER. Hal ini memungkinkan penulis untuk menghemat waktu dan penyimpanan, dikarenakan sistem akan memuat file adapter hasil fine-tuning(LoRA) dan menempelkannya secara dinamis ke model dasar Llama 3.2 3B saat run time. Mekanisme inilah yang menyederhanakan proses *deployment,* karena tidak perlu melakukan penggabungan model secara manual. Hanya cukup dengan satu baris konfigurasi pada Modelfile, model dasar bisa beroperasi dengan pengetahuan yang telah dilatih[6].

2.7 Unity Engine dan integrasi API

Unity adalah perangkat lunak pengembang yang handal untuk menciptakan pengalaman interaktif, bagus dalam format dua dimensi(2D) maupun tiga dimensi(3D). Meskipun fungsi utamanya adalah sebagai mesin pengembang gim (*Game Engine*), fleksibilitas arsitekturnya menjadikan Unity platform yang ideal untuk membangun antarmuka pengguna yang dinamis dalam sistem asisten virtual.

Relevansi utama Unity dalam penelitian ini terletak pada kemampuan komunikasi jaringannya. Sesuai dengan yang telah diterapkan oleh Patel *et al.* dalam pengembangan asisten ai interaktif, Unity dapat berkomunikasi sebagai klien yang berkomunikasi dengan layanan *backend* eksternal menggunakan protokol standar HTTP[25].

Unity memiliki fitur *UnityWebRequest*, yang memungkinkan aplikasi mengirimkan data teks ke *endpoint* API Ollama dan menerima respons dalam format

JSON secara *asynchronous*. Mekanisme ini memastikan proses pertukaran data antara antarmuka visual di Unity dan pemrosesan AI di server lokal berjalan lancar tanpa mengganggu responsivitas aplikasi. Oleh karena itu, penulis memilih Unity sebagai antarmuka karena kemudahannya dalam menjembatani komunikasi antara *frontend* dan *backend*.

2.8 Penelitian Terdahulu dan Posisi Penelitian

Pengembangan sistem asisten virtual di lingkungan pendidikan sebenarnya bukanlah hal baru. Studi yang dilakukan oleh Rahman dan Nugroho, misalnya, telah berhasil mengembangkan asisten virtual akademik yang terintegrasi dengan platform interaktif[24]. Banyak dari mayoritas penelitian serupa memiliki keterbatasan teknik, yakni bergantungan pada model berbasis cloud atau penggunaan model arsitektur model lama yang kurang responsif terhadap konteks bahasa alami yang kompleks.

Di sisi lain, aspek efisiensi infrastruktur mulai menjadi fokus utama. Park *et al.* penelitiannya menyatakan betapa pentingnya teknik efisiensi seperti PEFT untuk memungkinkan *deployment* chatbot cerdas pada perangkat lokal yang terbatas[23]. Meskipun teknologinya tersedia, implementasi praktis yang menggabungkan efisiensi model lokal ini dengan antarmuka visual 3D yang imersif—seperti yang ditawarkan oleh *Unity Engine*—masih jarang ditemukan dalam literatur sistem informasi akademik, sebagaimana diindikasikan dalam studi Patel *et al.*[25].

Sebagai upaya mengatasi masalah tersebut, penelitian ini hadir untuk mengisi kekosongan riset (*research gap*) tersebut melalui pendekatan yang lebih komprehensif. Kebaruan (*novelty*) yang ditawarkan penelitian ini terletak pada integrasi spesifik empat elemen kunci: (1) adopsi model *state-of-the-art* Llama 3.2 yang ringkas namun cerdas, (2) penerapan teknik LoRA untuk adaptasi domain data akademik UMM yang hemat komputasi, (3) pemanfaatan arsitektur Ollama untuk manajemen *adapter* yang fleksibel, serta (4) penyajian antarmuka interaktif berbasis *Unity*. Kombinasi yang telah dirancang khusus untuk menciptakan solusi cerdas dan efisien, dan layak diterapkan di infrastruktur kampus di Indonesia.