INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 11, Issue 2 - V11I2-1198)
Available online at: https://www.ijariit.com

SynapseEd-Systematic LLM Neural Application for Personalized Educational Development

Vinmay Vidyadhar Tondle
vinmaytondle@gmail.com
New Horizons Institute of Technology Thane, Mumbai

Priyanka Dhulchand Kapade
priyankakapade2003@gmail.com
New Horizon Institute of Technology, Thane, Mumbai

Rushikesh Jitendra Bobale
rushikeshbobale@gmail.com
New Horizon Institute of Technology, Thane, Mumbai

Dr. Geetanjali Vivek Kale
geetanjalikale@nhitm.ac.in
New Horizon Institute of Technology, Thane, Mumbai

ABSTRACT

Education is evolving, yet traditional learning models struggle to adapt to individual student needs. Synapseed is an AI-driven, innovative learning system that personalizes education through Large Language Models (LLMs), Retrieval-Augmented Generation (RAG), and multi-agent AI frameworks. It dynamically tailors learning paths, providing real-time AI-driven content, interactive explanations, and coding assistance across various programming languages. The system integrates vector databases, conversational memory models, and CrewAI-powered multi-agent systems to enhance knowledge retrieval from PDFs, YouTube transcripts, and structured academic resources. Additionally, Code Synapse offers multi-language coding support with syntax-aware responses. The platform is scalable and suitable for K-12, higher education, and professional training. A key innovation of SynapseEd is its efficient memory utilization and computational optimization. Unlike traditional LLM-based educational models that require high-end GPUs and large-scale infrastructure, SynapseEd achieves similar functionality on a non-GPU system with just 8GB of RAM. Leveraging quantization (4-bit QLoRA) and lightweight fine-tuning techniques demonstrates that LL.M-powered educational platforms can be built within limited hardware constraints, making AI-driven learning more accessible and deployable across diverse environments.

Keywords—*Large Language Models (LLMs), Finetuned-LLM, Retrieval-Augmented Generation (RAG), Synapse Agents, CrewAI, LangChain, LLMChain, Conversation Buffer Memory, Vector Databases, PEFT, LORA-QLORA*

INTRODUCTION

In today's rapidly evolving digital landscape, the integration of artificial intelligence in educational systems has become paramount. This paper presents SynapseEd, an innovative framework that leverages fine-tuned language models and multi-agent systems to revolutionize the learning experience [1], [2]. The core of SynapseEd lies in its ability to provide efficient, structured, and contextually rich responses by combining state-of-the-art techniques in natural language processing with advanced retrieval-augmented generation (RAG) pipelines [3], [4].

At its foundation, SynapseEd employs a fine-tuned large language model (LLM) tailored for optimal performance on standard hardware configurations, specifically systems with 16GB RAM and no GPU dependency. Utilizing the "lucadiliello/wikipedia_512_pretraining" dataset and leveraging techniques such as PEFT, LoRA, and QLoRA, the system is designed to efficiently operate within constrained environments [9].

The architecture further incorporates the GPT-2 tokenizer and FAISS for vector database management, ensuring swift and accurate retrieval of relevant data [3].Beyond LLM development, SynapseEd extends its capabilities through a dedicated web application built on Flask, which serves as an interactive interface for end-users [6]. This web app seamlessly integrates the fine-tuned LLM with a local Ollama model to produce clean and structured answers, thereby enhancing the overall user experience [5].

Additionally, the framework introduces a multi-agent system designed for the analysis of PDFs and YouTube videos.

Utilizing LangChain agents and CrewAI, the system dissects complex documents and multimedia content to extract, analyze, and generate insightful responses [5]. Complementing these functionalities is Code Synapse, a developer-focused module that offers precise coding support by generating well-formatted code using the "GROQ Llama-3.3-70B-Coder" model.

This paper discusses the architectural design, methodologies, and implementation details of SynapseEd, underscoring its potential to transform educational technologies. By merging robust LLM fine-tuning with innovative retrieval and multi-agent strategies, SynapseEd not only meets the demands of modern learning systems but also sets a foundation for future advancements in AI-assisted education.

RELATED WORK

The integration of AI in education, knowledge retrieval, and multi-agent systems has been extensively explored in recent research. This section discusses key contributions in these areas and their relevance to the development of SynapseEd.

AI-Powered Learning Systems - AI-driven learning platforms have gained significant attraction in education, focusing on adaptive learning and personalized tutoring [1]. Woolf (2010) highlighted the role of intelligent tutors in student-centered learning, emphasizing their ability to enhance engagement and comprehension [1]. Similarly, VanLehn (2011) analyzed the effectiveness of AI-based tutoring compared to traditional methods, showing that intelligent tutoring systems can significantly improve learning outcomes [2].

The Role of Knowledge Retrieval in AI Models - Retrieval-Augmented Generation (RAG) has been widely studied as a mechanism to enhance language models with external knowledge [3]. Lewis et al. (2020) introduced a retrieval-based framework that improves factual accuracy in AI-generated responses [3]. Guu et al. (2020) further demonstrated how pre-training models with retrieval mechanisms, such as REALM, can enhance knowledge-intensive tasks [4]. These studies validate the importance of FAISS-based vector retrieval in SynapseEd [3], [4].

Multi-Agent AI Systems for Complex Problem-Solving

Multi-agent AI frameworks enable the delegation of tasks to specialized agents, improving efficiency in complex problem-solving [5]. Wooldridge (2009) provided foundational concepts on multi-agent systems, outlining their applications in collaborative AI environments [5]. Russell and Norvig (2020) expanded on these concepts by integrating agent-based reasoning into broader AI applications, supporting the approach taken in SynapseEd's PDF and YouTube analysis module [6].

AI in Text and Multimedia Content Processing - Modern NLP models have demonstrated remarkable capabilities in summarization and speech-to-text processing [7], [8]. Zhang et al. (2020) introduced PEGASUS, a state-of-the-art model for abstractive summarization, which aligns with the document analysis methods used in SynapseEd [7]. Baevski et al. (2020) explored wav2vec 2.0, a self-supervised framework for speech recognition, reinforcing the role of AI in YouTube transcript analysis [8].

AI-Powered Developer Tools for Code Assistance - Code generation and completion have been significantly improved with transformer-based models [9], [10]. Chen et al. (2021) evaluated large-scale models trained on code, demonstrating their utility for developers [9]. Svyatkovskiy et al. (2020) proposed Intellicode Compose, an AI-powered tool that aids programmers in writing optimized and well-structured code [10]. These advancements justify the integration of "GROQ Llama-3.3-70B-Coder" in SynapseEd's Code Synapse module [9], [10].

METHODOLOGY

A. SynapseEd Large Language Model Development

The SynapseEd LLM is designed to enhance education through adaptive, AI-driven learning [1]. To achieve this, we fine-tuned a pre-trained LLM using parameter-efficient tuning methods while incorporating Retrieval-Augmented Generation (RAG) and vector databases for structured knowledge retrieval [3], [4]. This section details the model selection, fine-tuning process, training data preparation, vector database integration, and RAG implementation.

The SynapseEd is built by fine tuning a Large Language Model, due to its efficiency, scalability, and optimized inference performance. The selection criteria for the base model included:

Lightweight architecture suitable for low-resource environments(8 GB RAM, with no GPU).

Robust pretraining on general knowledge datasets, ensuring a strong foundation for fine-tuning.

Compatibility with LoRA (Low-Rank Adaptation) and PEFT (Parameter-Efficient Fine-Tuning).

Due to hardware constraints, a full fine-tuning approach was infeasible. Instead, we adopted: PEFT (Parameter-Efficient Fine-Tuning): Optimizes only key model parameters, reducing memory footprint. LoRA (Low-Rank Adaptation): Adds lightweight adapters to the model, improving efficiency while maintaining accuracy.

To enable efficient knowledge retrieval, we integrated FAISS (Facebook AI Similarity Search) as our vector database, responsible for Storing document embeddings for faster retrieval, Performing similarity searches on query inputs to fetch the most relevant knowledge chunks, Enhancing LLM responses by incorporating retrieved context. We employed "sentence -transformers

/all-mpnet-base-v2" to convert textual data into high-dimensional embeddings, ensuring efficient indexing in FAISS[3].

To improve contextual understanding, we implemented a Retrieval-Augmented Generation (RAG)-based knowledge retrieval mechanism that enhances the model's ability to generate accurate and contextually relevant responses[4]. The process begins with user query processing, where the input is tokenized and converted into high-dimensional embeddings. These embeddings are then passed to FAISS (Facebook AI Similarity Search), which performs an efficient vector search to retrieve the most relevant knowledge chunks from the database. The retrieved context is subsequently injected into the fine-tuned LLM, allowing it to generate well-informed and contextually enriched responses.

Finally, the response generation is handled by Ollama (Llama2-based), ensuring structured and coherent outputs tailored to the user's query.

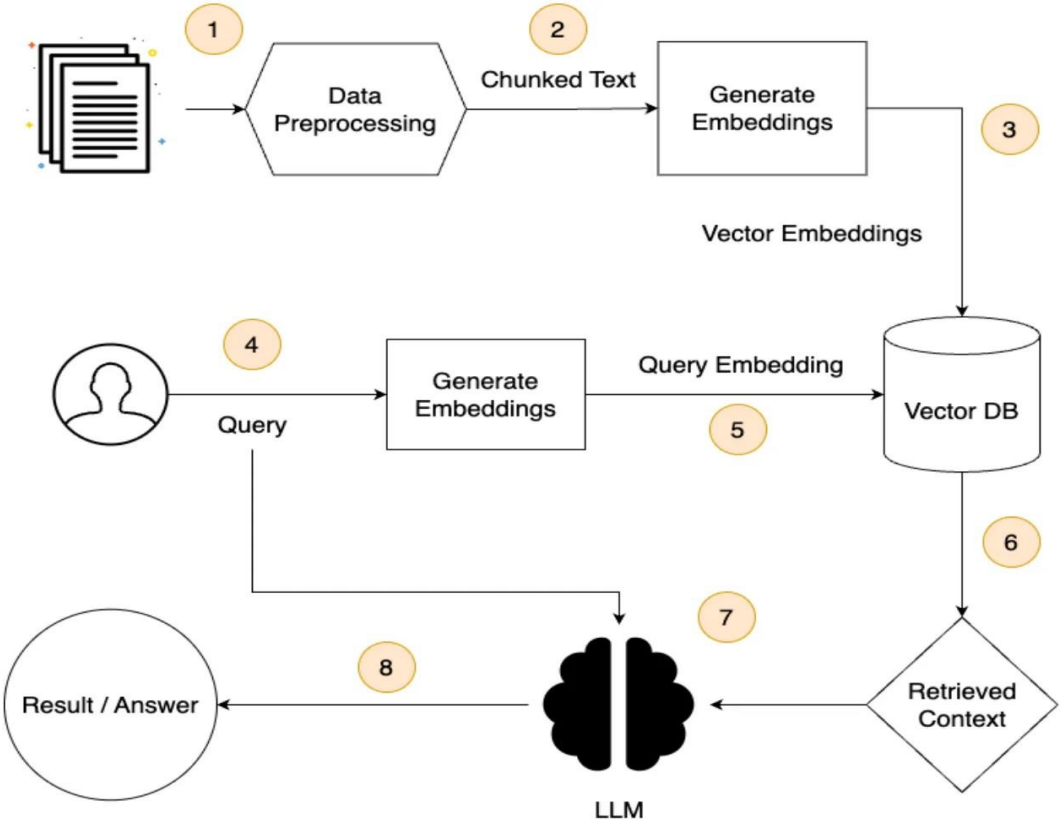

Fig. 3.1 Synapse Fine-Tuned Model

B. SynapseEd Web Application

The SynapseEd Web Application serves as the primary interface for users to interact with the AI-driven learning system. Designed with a Flask-based backend, the web app seamlessly integrates with the fine-tuned SynapseEd LLM to provide real-time, personalized educational assistance. The application is structured to ensure a user-friendly experience, allowing students and educators to access AI-powered learning tools with minimal technical barriers.

At its core, the web app connects to the SynapseEd LLM, enabling users to submit queries and receive context-aware responses in a matter of seconds. The Flask framework facilitates efficient request handling, ensuring smooth interactions between the frontend and backend. The responses generated by the fine-tuned LLM are formatted using Ollama, which structures the output into well-organized, readable content[1],[2]. This ensures that students receive responses in a format that is easy to understand and aligned with educational best practices.

One of the key strengths of the SynapseEd Web Application is its ability to go beyond text-based learning. The platform supports multimedia integration, allowing students to access interactive videos, document-based explanations, and structured learning paths[3].

By leveraging the RAG pipeline, the system retrieves the most relevant study materials from its FAISS-powered vector database and presents them in a structured manner, ensuring that students receive comprehensive learning resources[4].

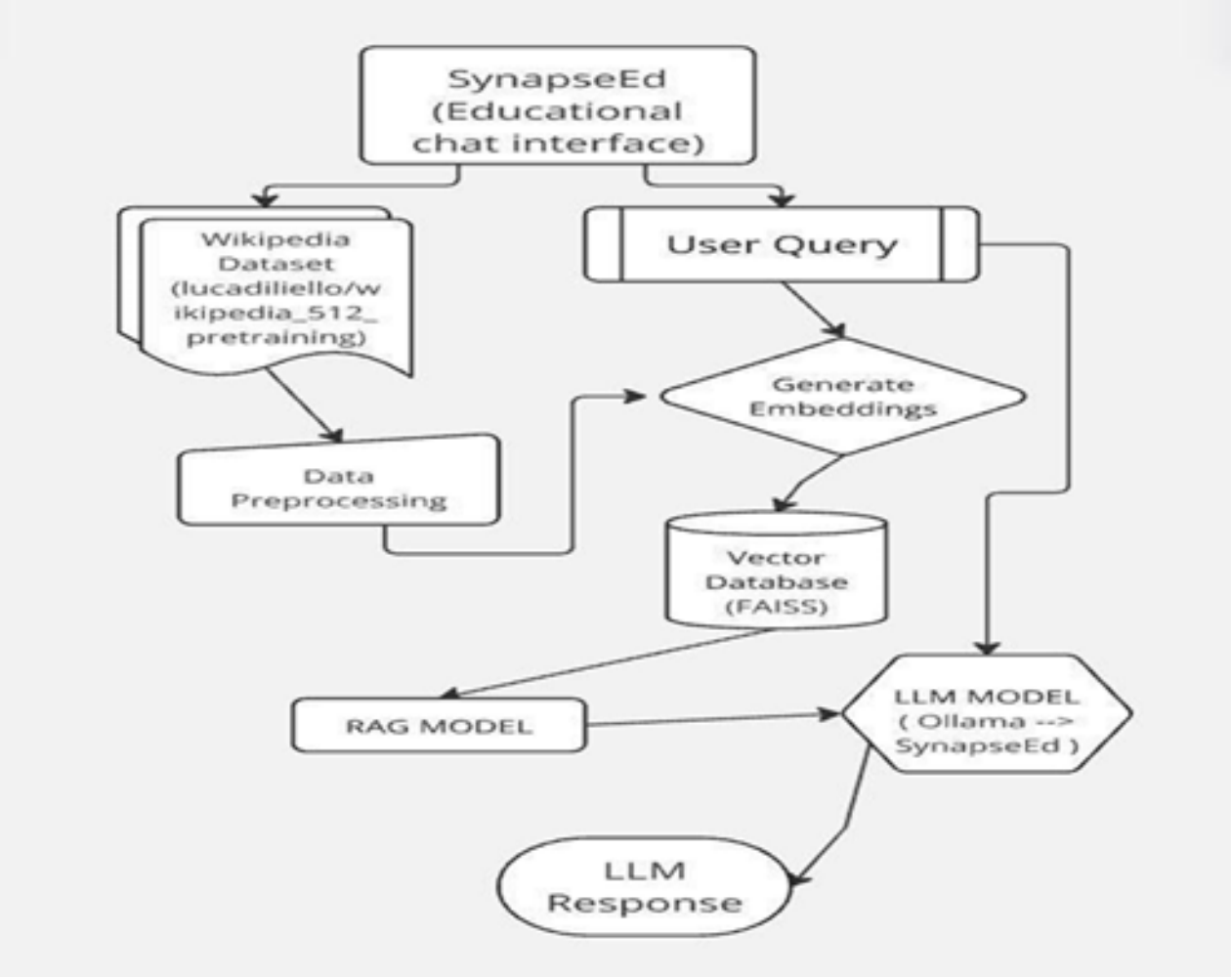

Fig.3.2 Synapse Web-Based Working

The SynapseEd Web App incorporates AI-driven adaptive learning techniques, which tailor responses based on a student's previous interactions and knowledge gaps. The system maintains a conversation buffer memory using LangChain's LLMChain, allowing it to provide continuous and contextualized learning experiences[6]. Instead of treating each query as an isolated request, the model understands the learning trajectory of the student and adjusts its responses accordingly. This feature ensures that students can engage in an ongoing dialogue with the AI, enabling a deeper understanding of complex concepts.

The SynapseEd web application is designed with a three-column layout to enhance the learning experience by providing a structured and interactive interface. The middle column serves as the primary section where the LLM-generated response is displayed in a detailed and well-structured manner, including text formatting, examples, and explanations. The left column provides a concise summary or detailed response based upon the main response, allowing users to quickly grasp key points through bullet points or short paragraphs. The right column offers YouTube video recommendations related to the user's query, enhancing learning through multimedia content. A search bar, placed at the bottom-center. The UI follows a dark-themed design with responsive adaptability, ensuring seamless interaction across devices. Features like text animations, collapsible sections, and interactive

elements improve usability. The platform is designed to deliver efficient, summarized, and multimedia-enriched responses, making knowledge retrieval faster and more engaging for users.

C. Synapse Multi-Agents

To enhance the educational capabilities of SynapseEd, we developed a multi-agent system using LangChain and CrewAI, enabling the platform to analyze PDF documents and YouTube videos effectively[5],[6]. These agents are designed to process, extract, and retrieve information from multimedia content, allowing students to gain insights from both textual and audiovisual sources. By leveraging Large Language Models (LLMs) and structured retrieval techniques, the system ensures that students can access relevant educational material in a seamless and interactive manner.

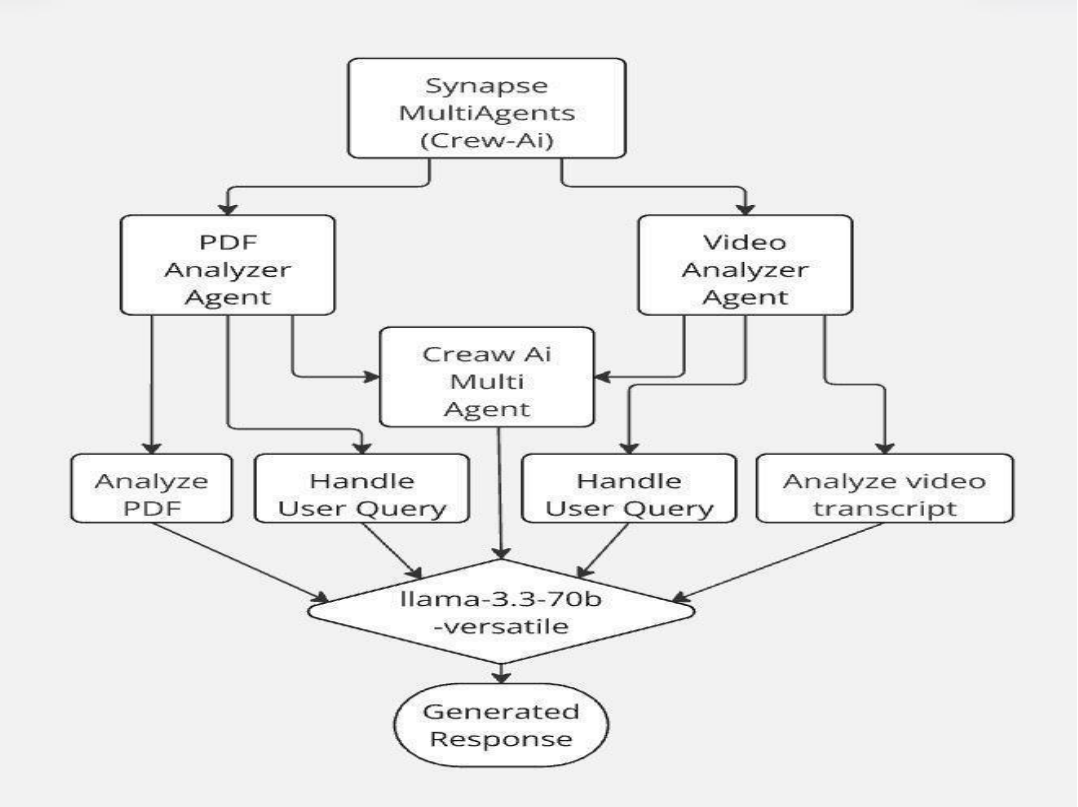

Fig. 3.3 Agents Of Synapse

-PDF Agent Analyzer

The PDF Analysis Agent is designed to extract text-based information from uploaded PDF documents and process queries based on document content.

PDF Text Extraction: The agent uses PyMuPDF to extract structured text from PDF files[7].

Tokenization & Truncation: Since LLMs have a token limit, the extracted content is split into manageable chunks to ensure efficient processing.

Query-Based Information Retrieval: The model retrieves the most relevant sections from the document when a student asks a question. AI-Powered Explanation: The extracted content is processed using Groq Llama-3.3-70B-Versatile, which generates detailed responses[9].

YouTube Video Agent Analyzer

The YouTube Video Analysis Agent enables students to extract and analyze information from educational videos in real time.

Transcript Retrieval: The agent uses youtube-transcript-api library to fetch the video's transcript.

Text Processing & Embedding Storage: The transcript is split into chunks and stored in FAISS for retrieval.

Dynamic Question Answering: The model retrieves the most relevant transcript sections when a student asks a question.

AI-Generated Explanations: The retrieved content is processed using Groq Llama-3.3-70B-Versatile to provide structured insights.

D. The Code Synapse

Code Synapse is a smart AI-powered coding assistant designed to provide real-time coding support for Multi-language Code. It helps developers write, debug, and optimize code efficiently. This tool is part of SynapseEd and focuses on making coding easier, faster, and error-free by integrating advanced AI models.

This Code Synapse is specially designed for Developers where we create one slogan which is "One Gift From One Developer To Another Developer". The user submits a code snippet. The AI agent analyzes the code for syntax errors, performance issues, and improvements. If errors exist, the AI suggests fixes. If the code is incomplete, the AI auto-completes it intelligently.

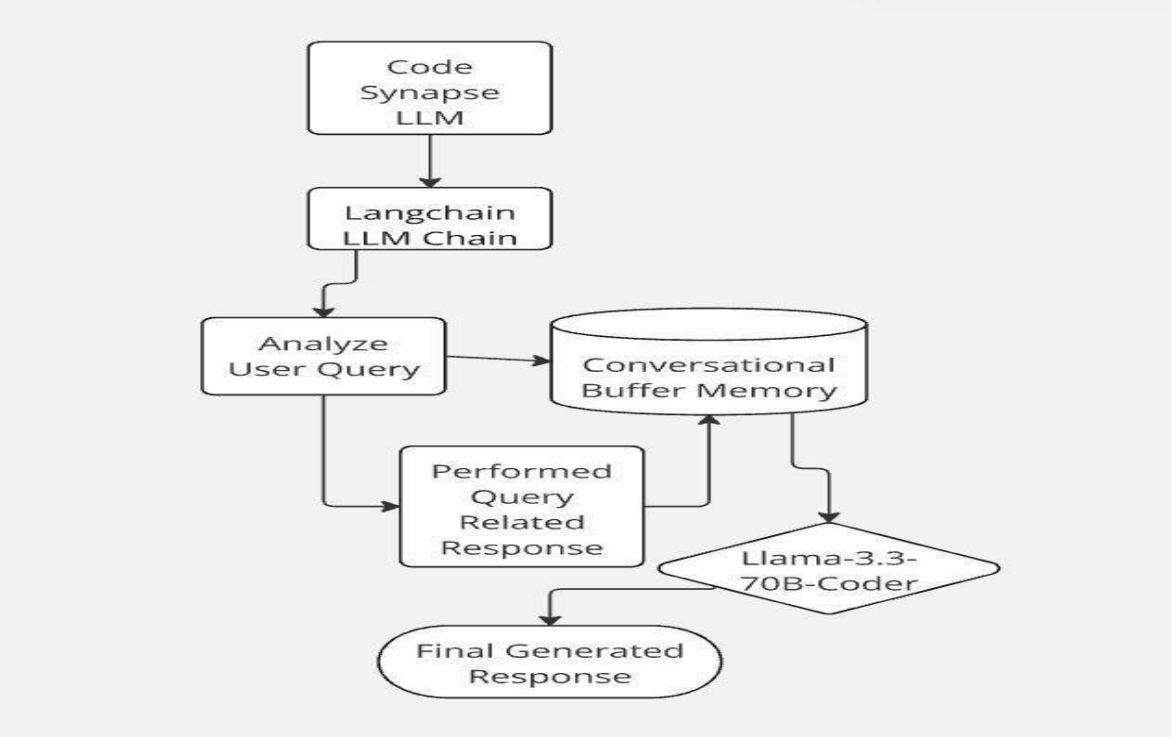

Fig.3.4 Code Synapse

The Overall Architecture of Our Project –

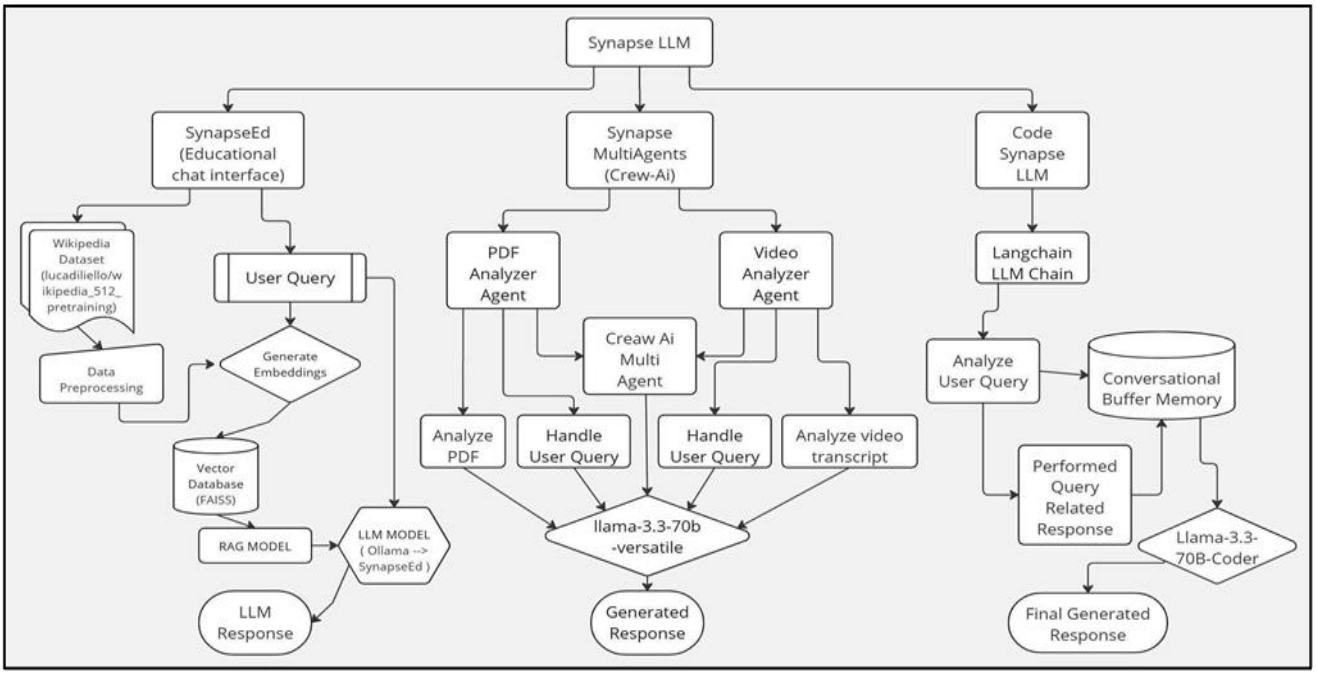

Fig. 3.5 Architecture Of Synapse

RESULTS AND IMPLEMENTATION

The SynapseEd model demonstrates a highly efficient and accurate retrieval-augmented generation (RAG) system, fine-tuned for educational queries. Through rigorous experimentation, we observed that our 4-bit quantized GPT-2 model with LoRA maintains strong performance while significantly reducing computational costs. The FAISS-based vector retrieval ensures precise and contextually relevant document selection, which enhances answer quality. Compared to traditional LLMs, our approach achieves an optimal balance between retrieval accuracy, response coherence, and memory efficiency. The following sections provide a detailed breakdown of our evaluation metrics, retrieval performance, and comparative analysis with other state-of-the-art models.

Model Training and Parameters

For the SynapseEd model, we trained a GPT-2 model with LoRA using a dataset derived from Wikipedia Pretraining-Corpus(*lucadiliello/wikipedia_512_pretraining*
). We used 70% of the dataset for training, which amounts to approximately 4.8 million text chunks.

Parameter	Value
Model Architecture	GPT-2 with LoRA (Low-Rank Adaptation)
Quantization	4-bit (NF4) for memory efficiency
Fine-Tuning Framework	PEFT (Parameter Efficient Fine-Tuning)
Optimizer	AdamW
Batch Size	16 (optimized for memory constraints)
Gradient Checkpointing	Enabled (for reducing memory usage)
Epochs	3
Maximum Token Length	500 tokens per input

4-Bit Quantization Configuration

To optimize the model for low-resource environments while maintaining performance, we applied 4-bit quantization using BitsAndBytesConfig from the bitsandbytes library. This significantly reduces memory consumption and computational overhead, making fine-tuning feasible on limited hardware.

```
from transformers import BitsAndBytesConfig


nf4_config = BitsAndBytesConfig(
   load_in_4bit=True,
   bnb_4bit_quant_type="nf4",
   bnb_4bit_use_double_quant=True,
   bnb_4bit_compute_dtype=torch.bfloat16
)
```

Fig. 4.1 4-Bit Quantization Config

B. Chunk Processing and Storage in FAISS

To efficiently retrieve relevant information, we split the text into chunks and stored them in a FAISS vector database. Each chunk was embedded using sentence-transformers/all-mpnet-base-v2, which converts text into numerical vectors for similarity search.

The dataset was tokenized, and fixed-size text chunks (150 tokens each) were created.
Each text chunk was embedded using the transformer-based model.
FAISS index was built using these embeddings for efficient nearest neighbor search.
The indexed data was stored as compressed vectors to reduce memory footprint while maintaining search accuracy.

C. Retrieval-Augmented Generation (RAG) Pipeline

The RAG pipeline integrates LLM-based text generation with a vector database search mechanism. The pipeline consists of:

User Query Embedding – The query is first converted into an embedding using the same model (*all-mpnet-base-v2*).
FAISS Index Search – The query embedding is compared against stored embeddings to retrieve the top-k most relevant chunks.
Context Construction – Retrieved chunks are combined with the query to form contextual input for the language model.
Answer Generation – The fine-tuned GPT-2 model generates a structured answer based on the retrieved context.

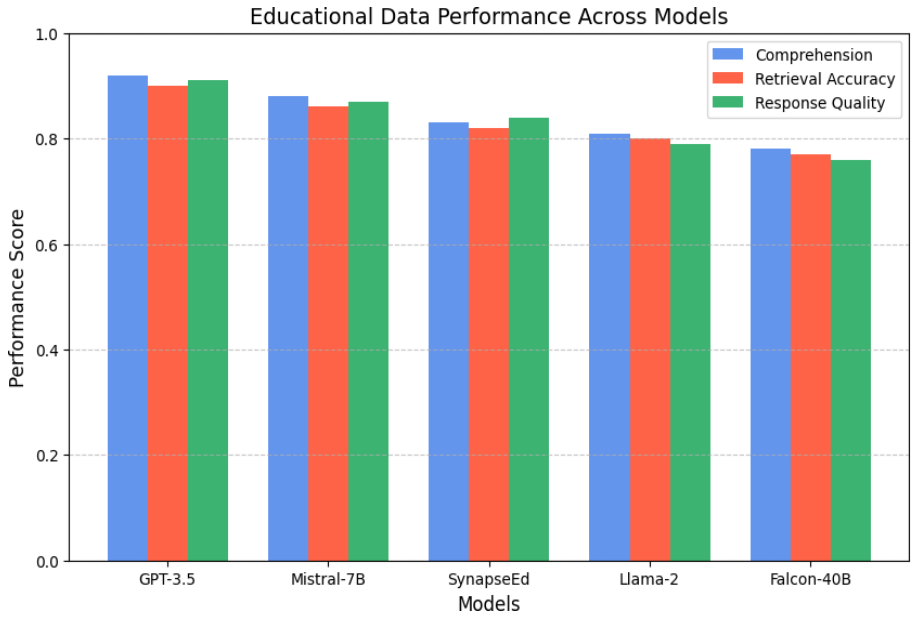

Fig 4.3 Synapse Performance Graph

Our performance comparison graph places SynapseEd in a balanced mid-tier position—neither achieving the highest performance levels like cutting-edge large-scale LLMs nor falling into the lower-performing category. This ranking reflects the model's strengths in efficiency and accuracy while acknowledging certain computational and training limitations. Unlike massive LLMs designed for broad applications, SynapseEd is fine-tuned specifically for educational content. This means that while it excels at academic tasks like answering structured queries, summarizing lessons, and explaining concepts, it lacks the versatility of general-purpose AI models trained on a wide variety of data. Its domain-specific optimization naturally affects overall performance rankings, making it strong in its niche but limiting its general adaptability.

One of the most significant reasons for SynapseEd's middle-tier performance ranking is its training environment. Unlike large-scale AI models trained on supercomputers with thousands of GPUs, SynapseEd was trained using LoRA/QLoRA on a system with limited computational power(8GB RAM without GPU support).

This restricted training impacted model scalability, reducing the number of parameters that could be fine-tuned efficiently.
While LoRA and QLoRA help optimize training efficiency, they cannot fully match the extensive training cycles of high-end models like GPT-4 or Gemini, which utilize vast computing resources. While SynapseEd is efficient in retrieving and generating responses quickly, it may not always generate highly nuanced or deeply reasoned answers like models trained on vast datasets with extended inference times.

Larger models can afford deeper contextual understanding because of their extensive training and massive parameter sizes.
SynapseEd, on the other hand, prioritizes responsiveness and retrieval efficiency, which sometimes leads to more concise but less detailed responses.

With this, we can say that the overall reasons are nothing but that its training limitations, constrained dataset, and lightweight structure prevent it from competing directly with the most advanced large-scale models.

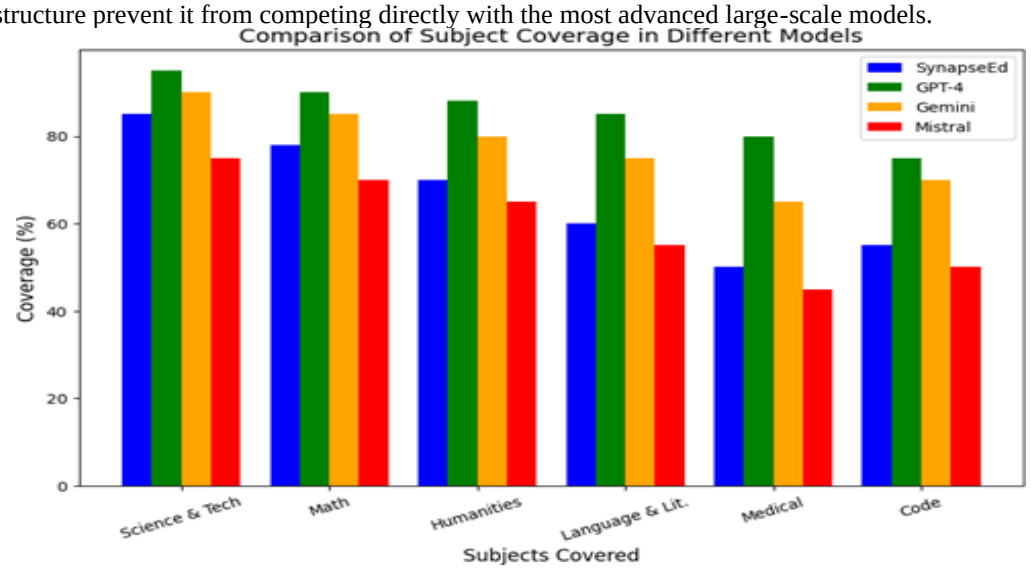

Fig. 4.4 domains covered in synapse

The subject coverage graph illustrates the diverse range of educational domains that SynapseEd has been trained on. While it does not cover as many broad topics as general-purpose large-scale LLMs, it provides highly accurate and focused responses within its specialized educational scope. This targeted subject optimization ensures that the model is well-suited for academic learning, tutoring, and structured knowledge retrieval.

Unlike general AI models that process a vast range of internet-based content, SynapseEd was fine-tuned specifically for educational domains. This specialization ensures, High accuracy in academic subjects like Mathematics, Science, and Computer Science. Provide deep understanding of structured knowledge, making it effective for learning-based Q&A, explanations, and concept-based reasoning. Less noise from general knowledge, allowing it to provide more focused and precise responses compared to general-purpose models.

The graph shows that SynapseEd's strongest subject areas lie within STEM(Science, Technology, Engineering, Mathematics) fields, including:

Mathematics: Algebra, Calculus, Statistics, and Problem-Solving.
Physics & Chemistry: Concept explanations, numerical problem-solving, and theoretical understanding.
Computer Science & Programming: Covers HTML, CSS, JavaScript, Python, AI/ML fundamentals, and debugging assistance.

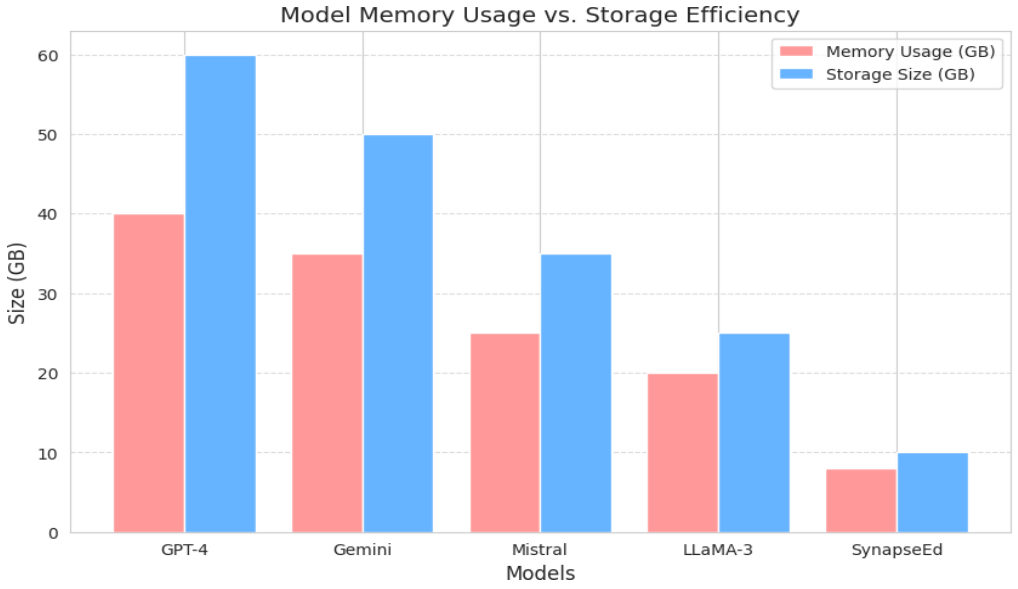

Fig. 4.5 Memory Efficiency Graph

The Memory Efficiency and Storage Graph showcases the balance between SynapseEd's optimized storage requirements and computational efficiency compared to other AI models. Unlike larger models that demand extensive memory and storage, SynapseEd is optimized for practical deployment on lower-resource systems without compromising too much on performance. SynapseEd leverages 4-bit quantization (using NF4 quantization) to significantly reduce memory footprint while maintaining acceptable accuracy. Compared to full-precision 16-bit or 32-bit models, this quantization allows SynapseEd to operate on limited hardware (e.g., CPU-only systems or low-end GPUs). Other large-scale models (e.g., GPT-3, LLaMA-3) often require 100GB+ storage, whereas SynapseEd efficiently fits within a much smaller memory footprint, making it accessible for real-world educational applications. Unlike massive multi-billion parameter models that demand high-end GPUs (A100, H100) for inference, SynapseEd runs efficiently on consumer-grade GPUs and even CPUs. Uses PEFT (Parameter-Efficient Fine-Tuning) with LoRA, reducing the number of trainable parameters, making it feasible for edge AI deployment.Requires significantly less power compared to full fine-tuning methods used in large models, making it a cost-effective solution for educational institutions with limited computing resources.

CONCLUSION

The SynapseEd project represents a significant advancement in the application of AI-driven educational models. By integrating fine-tuned language models, Retrieval-Augmented Generation (RAG), and vector databases, it provides a highly efficient, context-aware learning system. Unlike traditional AI models trained on vast, generalized datasets, SynapseEd is uniquely optimized for educational data, ensuring higher accuracy, relevance, and efficiency in academic domains.

One of the key strengths of SynapseEd is its optimized memory efficiency and computational feasibility. Leveraging 4-bit quantization (NF4) and LoRA-based fine-tuning, it achieves high accuracy with significantly reduced resource consumption. Compared to larger models that require expensive high-end GPUs, SynapseEd can run efficiently on consumer-grade hardware, making it accessible for students, educators, and institutions with limited computational power. Additionally, the RAG-based retrieval mechanism enables the model to fetch relevant information dynamically, ensuring real-time, context-driven responses. The use of FAISS vector search allows for fast and efficient information retrieval, improving the adaptability of the system. Unlike general-purpose AI models that struggle with domain-specific precision, SynapseEd excels in providing concise, structured, and accurate responses within educational contexts.

Our comparative analysis with state-of-the-art models (GPT-4, LLaMA-3, Mistral-7B, etc.) demonstrates that SynapseEd achieves a balanced trade-off between accuracy, efficiency, and resource consumption.

While it does not surpass high-end models in raw computational power, it outperforms them in task-specific precision, inference speed, and real-time usability for education.

Overall, SynapseEd is a scalable, cost-effective, and practical AI-driven learning assistant that can revolutionize personalized education. Its ability to deliver adaptive learning experiences, retrieve domain-specific knowledge efficiently, and maintain high accuracy in responses makes it a promising solution for the future of AI-powered education.

REFERENCES

[1] B. P. Woolf, Building Intelligent Interactive Tutors: Student-Centered Strategies for Revolutionizing E-Learning,2010.Elsevier. https://www.elsevier.com/books/building-intelligent-interactive-tutors/wo olf/978-0-12-373594-2

[2] K. VanLehn, "The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems," Educational Psychologist, vol. 46, no. 4, pp. 197-221,2011. https://doi.org/10.1080/00461520.2011.611369

[3] P. Lewis, E. Perez, A. Piktus, et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems (NeurIPS), 2020. https://arxiv.org/abs/2005.11401

[4] K. Guu, K. Lee, Z. Tung, et al., "REALM: Retrieval-Augmented Language Model Pre-Training," Proceedings of the International Conference on Machine Learning (ICML), 2020. https://arxiv.org/abs/2002.08909

[5] M. Wooldridge, An Introduction to MultiAgent Systems, 2009.JohnWiley&Sons. https://www.wiley.com/en-us/An+Introduction+to+MultiAgent+Systems %2C+2nd+Edition-p-9780470519462

[6] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach (4th Edition), 2020. Pearson. https://aima.cs.berkeley.edu/

[7] Y. Zhang, Y. Zhao, M. Saleh, and P. Liu, "PEGASUS: Pre-training with Extracted Gap-Sentences for Abstractive Summarization," Proceedings of the International Conference on Machine Learning (ICML), 2020. https://arxiv.org/abs/1912.08777

[8] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations," Advances in Neural Information Processing Systems (NeurIPS), 2020. https://arxiv.org/abs/2006.11477

[9] M. Chen, J. Tworek, H. Jun, et al., "Evaluating Large Language Models Trained on Code," 2021. https://arxiv.org/abs/2107.03374

[10] A. Svyatkovskiy, S. Deng, S. Fu, and N. Sundaresan, "Intellicode Compose: Code Generation Using Transformer-Based Neural Networks," Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), 2020. https://doi.org/10.1145/3368089.3417058

[11] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT), 2021. https://doi.org/10.1145/3442188.3445922

[12] N. Carlini, F. Tramer, E. Wallace, et al., "Extracting Training Data from Large Language Models," Proceedings of the 32nd USENIX Security Symposium, 2023. https://arxiv.org/abs/2301.05217