



Generative AI Study Assistant using RAG

Team Name: Rahaf Kanaan, Shifaa Al-zu'bi, Thabet Zamari, Rafah Ali, Tasneem Alassaf.

After building a RAG-based chatbot for the Generative AI course, the system was able to answer questions related to the lecture materials through a conversational interface. The chatbot retrieves relevant information from the course PDFs and generates responses based on the provided context.

During development, the chatbot was run using two different language models: an API-based model (GPT-4o via GitHub Models API) and a local open-source model (flan-t5-base).

Figure 1 is a simple and user-friendly graphical interface designed to improve accessibility and usability. The GUI allows users to interact with the Generative AI Study Assistant by entering questions through a text-based input and receiving responses in real time. The interface is connected directly to the same LangChain RAG backend and does not modify the retrieval or generation logic. Its purpose is to demonstrate the practical deployment of the RAG-based system in an interactive learning environment. Below are the chatbot responses to three sample questions when using each model, shown through “Copy-Paste” process.

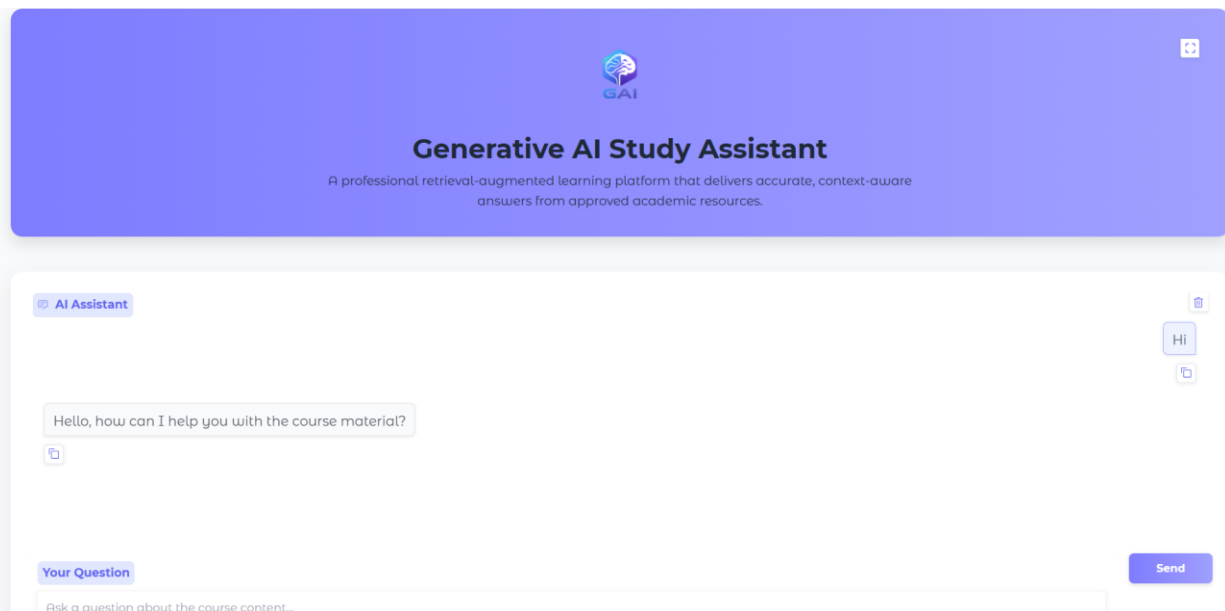


Figure 1 GUI for Generative AI Study Assistant

Question: What are the three core components of the Self-Attention mechanism within the Transformer architecture, and what is the specific role of each representation?

Answer using GPT-4o: Based on the course material, the three core components of the Self-Attention mechanism within the Transformer architecture are **Q (Query)**, **K (Key)**, and **V (Value)**. Here's a step-by-step explanation of their roles:

1. **Query (Q):**

- The Query represents the vector that asks the question, "What am I looking for?" It is used to determine the relevance of other tokens in the sequence to the current token.

2. **Key (K):**

- The Key represents the vector that provides the "identifier" or "descriptor" for each token in the sequence. It is used to match against the Query to measure how relevant or important a particular token is to the current token.

3. **Value (V):**

- The Value represents the actual content or information of the token. Once the relevance (or attention score) between the Query and Key is calculated, the Value is used to compute the weighted sum, which forms the refined contextual representation of the token.

In summary, the Self-Attention mechanism uses **Q**, **K**, and **V** to allow each token in a sequence to attend to all other tokens (including itself) and calculate a refined representation that captures long-range dependencies efficiently. This process is fundamental to both the Encoder and Decoder blocks in the Transformer architecture.

Answer using flan-t5-base: Token indices sequence length is longer than the specified maximum sequence length for this model (1677 > 512). Running this sequence through the model will result in indexing errors

Answer:

Encoder Block (for comprehension) and Decoder Block (for generation) within the full Transformer model. Differentiate the roles and internal components of the Encoder Block (for comprehension) and the Decoder Block (for generation) within the full Transformer model. Differentiate between the three main Transformer architectural variants (Encoder-Only, Decoder-Only, and Encoder-Decoder). Allows a token to attend to all other tokens in the same sequence (and itself) to calculate a refined contextual representation. It enables long-range dependencies to be captured directly and efficiently. Used in both the Encoder and Decoder blocks. Multi-Head Attention: The process is run multiple times in parallel using different, independently learned **Q**, **K**, and **V**.

Question: I am developing two AI tools: one for translating technical manuals from English to Arabic and another for sentiment analysis to identify the tone of customer reviews.

Which specific Transformer variants (Encoder-Only, Decoder-Only, or Encoder-Decoder) would you recommend for each tool? Justify your choice based on their primary goals.

Answer using GPT-4o: Based on the provided course material, here is the recommendation for the two AI tools you are developing:

Tool 1: Translating Technical Manuals from English to Arabic

****Recommended Transformer Variant:**** ****Encoder-Decoder****

****Justification:****

- The task of translation requires the model to fully understand the source text (English) before generating the target text (Arabic).
- The ****Encoder**** processes the entire input text to create a comprehensive representation of its meaning, while the ****Decoder**** generates the translated output based on this representation.
- This approach is ideal for translation tasks because it ensures that the meaning and context of the source language are accurately captured and transformed into the target language.

Tool 2: Sentiment Analysis to Identify the Tone of Customer Reviews

****Recommended Transformer Variant:**** ****Encoder-Only****

****Justification:****

- Sentiment analysis involves identifying the tone or sentiment of a given text, which requires a thorough understanding of the entire input text.
- The ****Encoder-Only**** model is suitable for this task because it focuses on comprehensively analyzing the input text without generating new text.
- Since sentiment analysis does not involve producing a new output (like translation or summarization), the Encoder-Only model is efficient and effective for this purpose.

Summary of Recommendations:

1. **Translation Tool:** Use **Encoder-Decoder** because it requires reading the source text and generating a target text.
2. **Sentiment Analysis Tool:** Use **Encoder-Only** because it requires understanding the input text without generating new text.

These recommendations align with the rationale provided in the course material.

[illegible]

Question: If an organization needs to adapt a massive Large Language Model (LLM) to a niche domain but has a strictly limited compute budget, should they use Full Fine-Tuning or Parameter-Efficient Fine-Tuning (PEFT) techniques like LoRA? Explain your recommendation focusing on GPU resources and storage efficiency.

Answer using GPT-4o: Based on the course material, the organization should use **Parameter-Efficient Fine-Tuning (PEFT)** techniques like LoRA for adapting a massive Large Language Model (LLM) to a niche domain under a strictly limited compute budget. Here's the step-by-step reasoning:

1. **Characteristics of Full Fine-Tuning**:

- Full fine-tuning involves updating **all the parameters** (weights) of the model based on new data.
- Modern LLMs are extremely large, with billions of parameters, making full fine-tuning computationally expensive.
- It requires **large GPU resources**, which would not align with the organization's limited compute budget.
- While full fine-tuning provides maximum flexibility, it comes with the **highest cost** in terms of computational power and storage.

2. **Characteristics of PEFT**:

- PEFT focuses on **freezing the majority of the model's pre-trained weights** and updating only a **small subset of parameters** (small trainable modules).
- This approach is **computationally efficient** because fewer parameters are updated, requiring **less computing power**.
- PEFT is **storage-efficient**, as the task-specific modules take up minimal storage space.
- Training is faster due to the reduced number of parameters being updated, making it suitable for scenarios with limited resources.

3. **Recommendation**:

- Given the organization's ****strictly limited compute budget****, PEFT techniques like LoRA are the ideal choice because they minimize GPU resource requirements and storage needs.

- PEFT is particularly useful for adapting models to niche domains efficiently, as it balances performance improvements with resource constraints.

In conclusion, **Parameter-Efficient Fine-Tuning (PEFT)** is the recommended approach for the organization due to its low computational cost, storage efficiency, and faster training time, which align with the constraints of limited resources.

[illegible]

modernnnLarge Language Models are huge and havenbillions of parameters. This works well but the problem is that modernnnLarge Language Models are huge and havenbillions of parameters. This works well but the problem is that modernnnLarge Language Models are huge and havenbillions of parameters. This works well but the problem is that modernnnLarge Language Models are huge and havenbillions of parameters.

From the chatbot responses to the three questions, it is clear that the choice of language model significantly influenced the quality and clarity of the generated answers. When using GPT-4o, the chatbot consistently produced well-structured, detailed, and coherent explanations that directly addressed the questions and demonstrated a strong ability to reason and organize information based on the course material.

In contrast, the responses generated using the local open-source model (flan-t5-base) were less stable in quality. While some answers contained relevant information, others suffered from repetition, formatting issues, or incomplete explanations, especially for more complex or multi-part questions. These observations suggest that although both models were successfully integrated into the RAG-based chatbot, API-based models provided clearer and more reliable responses in this implementation, while the local model remained useful as a lightweight and offline-friendly alternative. Figure 2 shown how the chatbot interacts with user in the friendly way.

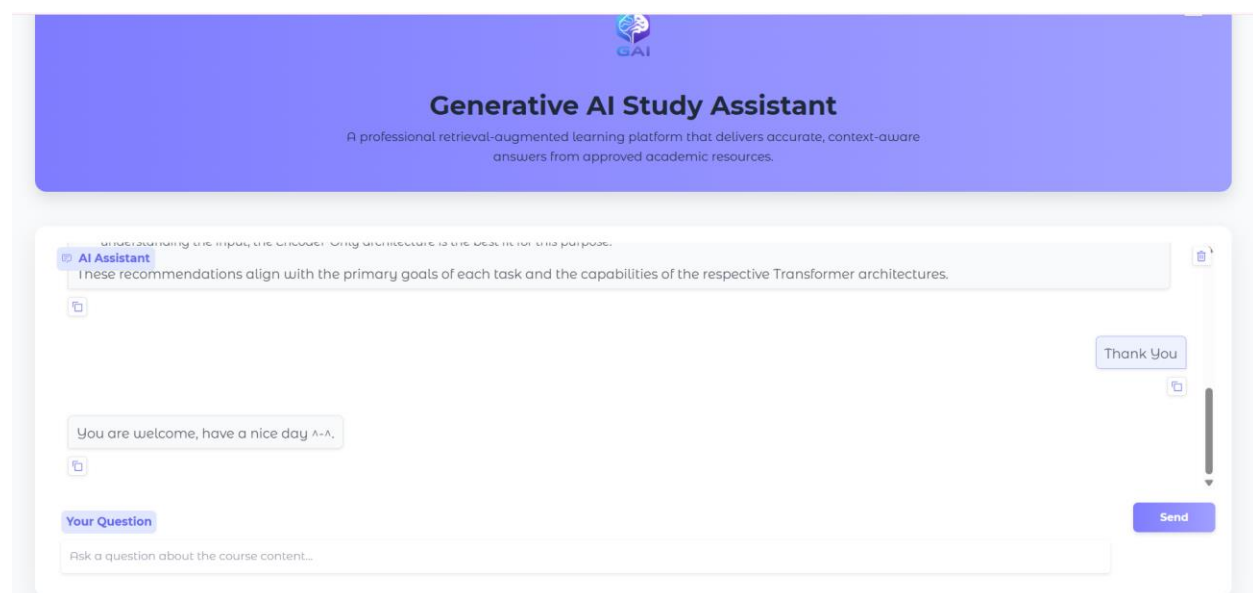


Figure 2 Chatbot Friendly interaction

GoogleColab-Link:

<https://colab.research.google.com/drive/1E-epswJmLOC-U95PgrE1jBdx-VIUvs2R?usp=sharing>