

6. Comparison Between RAG, Fine-Tuning, and Custom GPT

Assem Hijazi – Experience, Knowledge & Ideas

This article provides a focused comparison between the three core methods for customizing **Large Language Models (LLMs)**: **RAG (Retrieval-Augmented Generation)**, **Fine-Tuning**, and **Custom GPT**. Each method offers distinct advantages depending on the specific needs of an organization, and they serve different purposes in shaping a custom LLM.

RAG vs. Custom GPT

Feature	RAG (Retrieval-Augmented Generation)	Custom GPT
Purpose	Retrieves real-time data to enhance the custom sub LLM's responses	Guides the custom sub LLM's responses based on predefined rules or documents
Customization Type	Integrates external data dynamically to generate contextually accurate responses for the custom sub LLM	Customizes the custom sub LLM's behavior using user-uploaded documents or instructions
Impact on the Model	Creates a custom sub LLM that temporarily integrates real-time external data without altering its parameters	Creates a custom sub LLM that follows rules or documents without modifying its parameters
Data Source	Pulls data from real-time external databases or documents	Uses static documents or instructions uploaded by the user
Dynamic/Static Data	Dynamic: Retrieves live data with each request	Static: Operates based on predefined rules and data
Customization Level	Integrates external, real-time data into the custom sub LLM responses	Customizes behavior without altering the custom sub LLM's core structure
Knowledge Update	Doesn't permanently update the custom sub LLM knowledge; dynamically retrieves fresh data	Doesn't update the custom sub LLM knowledge, just applies predefined instructions
Implementation	Requires external data sources (APIs, databases) to pull live data	Requires user-provided documents or instructions to guide responses
Use Case Example	Retrieving live weather or stock market data to enhance the custom sub LLM's response	Using a company's internal documents to guide customer service responses from the custom sub LLM
Scope	Broad and flexible: Can pull any relevant real-time data	Narrower: Follows specific instructions based on user-provided documents
Summary	Adds real-time data to enhance the custom sub LLM's responses, perfect for dynamic, up-to-date information needs.	Relies on predefined documents and rules, ideal for users who want specific behavior but don't need real-time updates

RAG vs. Fine-Tuning

Feature	RAG (Retrieval-Augmented Generation)	Fine-Tuning
Purpose	Retrieves real-time external data to enhance the custom sub LLM responses without altering its core knowledge	Trains the custom sub LLM on specific datasets to specialize in certain tasks or domains
Customization Type	Dynamically pulls data in real-time to generate relevant answers for the custom sub LLM	Deeply customizes the custom sub LLM by training on specialized datasets
Impact on the Model	Creates a custom sub LLM that uses real-time data but doesn't change its underlying parameters	Creates a custom sub LLM that permanently updates its parameters based on the fine-tuned data
Data Source	Uses external, real-time databases or APIs	Uses static, curated datasets provided for training
Dynamic/Static Data	Dynamic: Pulls live, up-to-date data for every query	Static: Once fine-tuned, the custom sub LLM retains static knowledge based on training data
Customization Level	Adds external real-time data to enhance the custom sub LLM without altering core parameters	Adjusts the custom sub LLM for deeper, more specific tasks or domains by modifying its internal parameters
Knowledge Update	Doesn't update the custom sub LLM's core knowledge; integrates new data dynamically	Permanently updates the custom sub LLM's knowledge based on the fine-tuned dataset
Implementation	Requires external data sources (APIs, databases) to pull live data	Requires training datasets and API access for fine-tuning the custom sub LLM
Use Case Example	Pulling real-time legal updates to supplement the custom sub LLM's legal knowledge	Fine-tuning a custom sub LLM to specialize in medical research for specific diseases
Scope	Broad: Can pull any relevant real-time data	Narrower: Specializes the custom sub LLM in a specific domain or subdomain
Summary	Retrieves external, real-time data to dynamically adjust the custom sub LLM's responses without changing its core structure.	Provides permanent specialization of the custom sub LLM, training it on specific datasets for more accurate, niche task handling.

Fine-Tuning vs. Custom GPT

Feature	Fine-Tuning	Custom GPT
Purpose	Trains the model (custom sub LLM) on specific datasets to specialize in certain tasks or domains	Guides the model (custom sub LLM) responses based on predefined rules or documents, without changing its core structure
Customization Type	Deep customization: Adjusts the custom sub LLM's parameters based on specialized data	Rule-based customization: Modifies behavior using user-defined instructions, without adjusting custom sub LLM parameters
Impact on the Model	Creates a custom sub LLM (child LLM) specialized in the task or domain, with permanent updates to parameters	Creates a custom sub LLM that follows specific rules or documents, but doesn't change the underlying custom sub LLM's parameters
Data Source	Uses specific, curated datasets provided for training	Works with static documents or instructions uploaded by the user
Knowledge Update	Permanently updates the custom sub LLM's specialized knowledge based on training data	Doesn't update the custom sub LLM's knowledge, just adjusts its response behavior
Scope	Narrower and more focused: Specializes the custom sub LLM in a specific domain or subdomain	Broader scope: Can be used for many purposes depending on the documents or rules defined by the user
Flexibility	Once fine-tuned, the custom sub LLM is locked into that specialized knowledge	More flexible: Users can easily change or update the rules or documents without retraining the custom sub LLM
Data Handling	Requires training data relevant to the domain	Uses user-uploaded documents or specific rules to guide responses
Implementation	Requires API access and training data to fine-tune the custom sub LLM	Can be implemented by uploading rules or documents without the need for retraining
Use Case Example	Training a custom sub LLM to specialize in medical research for specific diseases	Customizing the custom sub LLM to provide customer service responses based on a company's internal policies
Adaptability	Adapts well to complex, specialized tasks over time	Adapts quickly to changing rules or data, but less specialization in specific tasks
Time to Implement	More time-consuming: Requires training and testing the custom sub LLM on a specific dataset	Quicker to implement: Users can upload documents or set rules without needing to train the custom sub LLM
Summary	Fine-tuning creates a highly focused custom sub LLM that specializes in a domain or task by updating its parameters.	Custom GPT allows for quick, flexible customization through rules and documents, creating a custom sub LLM that adapts without changing its core knowledge