

Introduction to the Customization of LLMs:

[Assem Hijazi – Experience, Knowledge & Ideas](#)

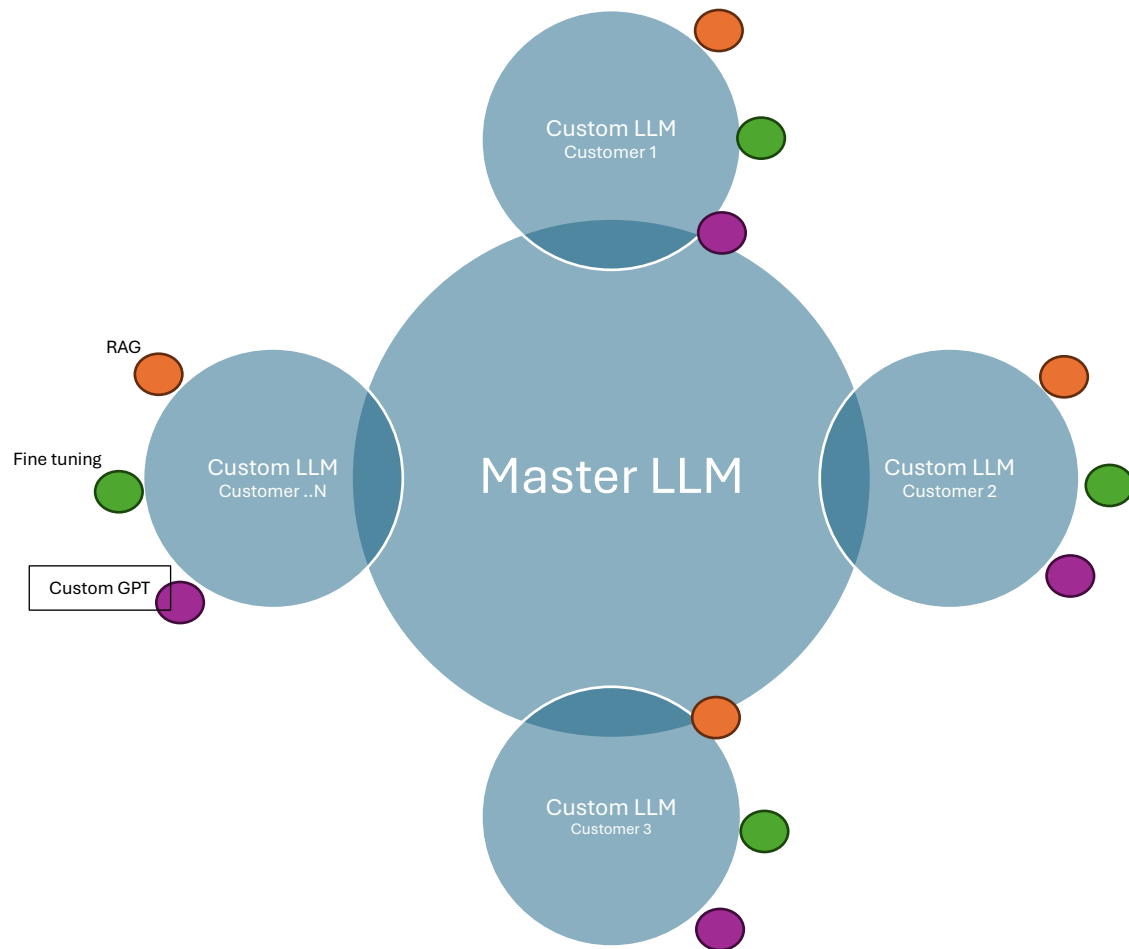
In modern large language models (LLMs) like ChatGPT, customization plays a key role. While the master LLM—the main model—comes pre-trained with a broad set of knowledge, users often need the model to focus on their specific needs, whether it's handling their own data, providing specialized information, or answering domain-specific questions. However, when users request these custom capabilities, they do so without changing the master LLM itself. Instead, the master LLM provides technologies that allow for these custom needs to be met while keeping the core model intact.

When customers or organizations need the LLM to focus on their unique data or knowledge, they can rely on the master LLM for support. The master LLM provides three main customization methods: RAG, fine-tuning, and custom GPT. These methods help users build what we call custom sub LLMs (or child LLMs), tailored to handle their specific requirements.

Think of the master LLM as a main brain with a wealth of solid, pre-trained knowledge that doesn't change “unless done by the main owner of the master LLM”. It is designed using mathematical structures, algorithms, and static training data. But it's also flexible enough to assist users in creating custom LLMs for their own needs, based on external data or specific requests.

It's important to note that when the master LLM helps build custom LLMs, it does so without altering its own core knowledge base or thinking. Instead, it supports these child LLMs by using its existing capabilities to interpret new data, answer specific queries, or integrate user-provided information. It's as if the master brain is lending its thought process to assist in handling specialized tasks, but it doesn't change its own thinking—it's more like an assistant that helps others with custom tasks while keeping its own knowledge intact.

This approach allows users to build small, specialized LLMs for their own purposes, such as answering questions or managing specific types of data. These child LLMs work together with the master LLM to offer precise, personalized responses based on the user's needs.



Master LLM and Custom Sub LLMs: How They Work Together

The Master LLM (represented as a large central circle) is the core model, like ChatGPT's master LLM, which contains general-purpose knowledge trained across a wide variety of topics and domains. This master LLM remains unchanged and serves as the base for all interactions, ensuring that the model can respond across many different areas without being specifically fine-tuned for one.

Around the master LLM are customer-specific sub LLMs (represented by smaller connected circles). Each customer-specific LLM is tailored for a particular user, business, or organization. These custom sub LLMs do not modify the master LLM, but rather, they are created to address specific needs through three key methods RAG (Retrieval-Augmented Generation), Fine-Tuning and Custom GPT.

Each customer-specific sub LLM acts as a child LLM, integrating seamlessly with the master LLM while maintaining its own specialization. This flexible architecture allows customers to create powerful, tailored solutions for their own unique needs, without impacting the broader model's general knowledge and capabilities.

The main definitions

1. Retrieval-Augmented Generation (RAG):

- What it is: RAG retrieves real-time data from external sources, like databases or documents, to enhance the LLM's responses. However, it does not modify the master LLM itself. Instead, it

creates a custom-specific sub LLM or child LLM that pulls external data in real-time to provide accurate, up-to-date answers.

- **How it works:** The master LLM processes the initial query, and then the RAG component retrieves relevant data from external sources. This data is combined with the pre-existing knowledge of the master LLM to generate a response. The real-time data retrieval allows for dynamic responses without permanently altering the core LLM.
- **In simple terms:** RAG is like adding a "live search engine" to the LLM that helps it access and use real-time information for specific queries, while creating a custom sub LLM to handle the specific needs of that task.
- **Key point:** RAG does not update the master LLM, but instead creates a child LLM that integrates real-time data retrieval.
- **Example-Healthcare Application:** A custom sub LLM for a medical institution retrieves real-time patient records and the latest medical research papers to generate responses. The custom sub LLM created with RAG doesn't change the master LLM but accesses relevant external data for more accurate answers

2. Fine-Tuning:

- **What it is:** Fine-tuning involves training a model on specific datasets to make it more specialized for certain tasks. However, it does not modify the master LLM. Instead, it creates a custom-specific sub LLM (or child LLM) that is specialized in the area it was fine-tuned on.
- **How it works:** Developers provide specialized data to fine-tune a model on a specific domain or topic. The resulting custom sub LLM can now handle queries within that specialized area but doesn't alter the core model.
- **In simple terms:** Fine-tuning is like giving the LLM a focused training course that creates a new, specialized child LLM for specific tasks, but without changing the master LLM.
- **Key point:** Fine-tuning creates a custom sub LLM for specific tasks and does not modify the master LLM.
- **Example-Legal Application:** A law firm fine-tunes a model on a dataset of past legal cases to create a custom sub LLM that specializes in legal queries. This fine-tuned model does not update the master LLM, but it becomes an expert in handling specific legal queries.

3. Custom GPT:

- **What it is:** Custom GPT allows users to define specific behaviors, rules, and responses for the LLM without changing its underlying architecture or training. This creates a custom-specific sub LLM that adheres to those user-defined rules and documents.
- **How it works:** Users upload custom documents or define rules that guide the model's responses, creating a custom GPT model that operates based on the user's needs, but again, it doesn't modify the master LLM.
- **In simple terms:** Custom GPT is like setting up a custom rulebook that the LLM follows for specific queries, but it doesn't alter the core model or its original knowledge.
- **Key point:** Custom GPT creates a custom sub LLM with user-defined rules, separate from the master LLM.

- **Example-Retail Company:** A retail company uses custom GPT to create a custom sub LLM that answers customer queries based on company-specific return policies and product information. This model uses static documents but doesn't change the master LLM.

Comparison Table: 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

Comparison Table: 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.

Comparison Table: 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

Master LLM vs. Sub LLMs (or Child LLMs):

- **RAG, fine-tuning, and custom GPT** do not update the **main LLM** (which we can refer to as the **master LLM**). Instead, they create **sub LLMs** (or **child LLMs**) that work in tandem with the master LLM to provide more tailored or domain-specific functionality.
- These **sub LLMs** are isolated, domain-specific versions of the master LLM that are customized for individual organizations or purposes, but they do not affect or change the core model (master LLM).
- By creating these **sub LLMs**, developers or organizations can leverage the general capabilities of the master LLM while adding their own specialized layers on top for specific use cases, without impacting the broader knowledge of the model.

Important Distinction:

- **Fine-tuning** and **RAG** should not be seen as methods to update the master LLM itself. Instead, they are ways to build specialized, custom versions (sub LLMs) of the master LLM.
- These **sub LLMs** work alongside the master LLM but are tailored to meet specific domain needs, whether through specialized training (fine-tuning) or real-time data access (RAG).
- The master LLM remains **untouched**, ensuring its broad applicability across many domains, while the sub LLMs cater to individual, niche requirements.

How Fine-Tuning and RAG Create Sub LLMs:

- **Fine-Tuning:** When fine-tuning is done, a **sub LLM** is created with specialized knowledge. This sub LLM is connected to the master LLM but only engages for domain-specific questions. It doesn't update the master LLM's parameters.
- **RAG:** When RAG is used, the model retrieves **external data** in real-time, but again, the master LLM's core knowledge is not updated. The retrieval process augments the response without permanently changing the model.

Why the Master LLM Needs Strong Domain Knowledge:

- For these sub LLMs to be effective, the **master LLM** needs to have a **solid foundational knowledge** of the domain or subdomain. If the master LLM lacks understanding in a particular area, the sub LLM might struggle, especially if it encounters entirely new terms or concepts that the master LLM can't relate to.
- The **master LLM** provides the essential framework, and without that foundational knowledge, the sub LLM will face limitations, even if it's fine-tuned or uses RAG for real-time data.

Summary:

- **Master LLM:** The core model that remains unchanged and provides a broad understanding of many domains.
- **Sub LLMs (Child LLMs):** Created through fine-tuning or RAG, they are specialized versions of the master LLM that cater to specific use cases but don't modify the core model.
- **Fine-Tuning and RAG** allow organizations or individuals to have their **own custom sub LLMs** without altering the master LLM.
- It's essential for the master LLM to have foundational domain knowledge, or else the sub LLMs may struggle with new or niche terms.

This approach ensures that organizations can build specific tools and models for their needs without compromising the integrity of the master LLM, while still leveraging its overall capabilities.

A **custom sub LLM** handles **very niche data** by focusing on a **specific, small area** of knowledge, using the fine-tuning or RAG mechanisms to provide accurate responses in that narrow context. Here's how it works:

1. Fine-Tuning for Niche Data:

- **Customization through Data:** When fine-tuning is applied, the custom sub LLM is trained on **highly specialized data** related to a particular topic or niche. This allows the sub LLM to learn **specific patterns, terminology, and relationships** that are unique to that niche.
- **Precision over Breadth:** Unlike the master LLM, which covers a wide range of topics, the fine-tuned sub LLM narrows its focus to provide **accurate and context-specific responses** within its niche. It has deeper knowledge of the niche but lacks the broader understanding of the entire domain.
- **Example:** Imagine fine-tuning a sub LLM for a **rare medical condition**. The custom sub LLM will become highly skilled at answering questions related to that condition, understanding specific terms, treatments, and patient care details that might not be well-covered in the master LLM.

2. RAG for Niche Data:

- **Real-Time Retrieval:** RAG allows a custom sub LLM to handle very niche data by **retrieving real-time information** from highly specific sources (e.g., private databases, niche research documents, or up-to-date reports) when responding to a query.
- **Combining External Knowledge:** While the master LLM might not have the necessary depth for niche topics, RAG enables the sub LLM to pull relevant external data from **trusted, niche sources**, enhancing its responses by combining pre-trained knowledge with fresh, real-time insights.
- **Example:** A custom sub LLM in **investment strategies** might use RAG to pull the latest stock market data from a private financial database. Even though the master LLM may not have real-time stock prices, the RAG-enabled sub LLM can pull this data dynamically and offer precise, timely advice for niche financial markets.

3. Handling Unique Terminology and Context:

- **Terminology Mastery:** For very niche data, custom sub LLMs are trained to understand the **specific terminology** that is unique to that area. This could involve uncommon words, abbreviations, or technical jargon that the master LLM might not recognize or understand well.
- **Contextual Relevance:** Fine-tuning ensures that the sub LLM interprets queries in the context of that niche. It learns how terms are used in specific scenarios and becomes adept at recognizing subtle differences in meaning that are important for that niche.

4. Handling Limited Data in Niche Areas:

- **Overcoming Data Limitations:** Niche areas often have **limited available data**, which can make traditional training difficult. However, custom sub LLMs can overcome this limitation by fine-tuning on **focused, high-quality datasets** that are specific to the niche, allowing the model to generalize knowledge even from smaller data sources.
- **RAG Complements Limited Data:** If there isn't enough data to fine-tune the model effectively, RAG can complement the sub LLM by providing access to **external, real-time niche data**, ensuring that the LLM's knowledge is supplemented as needed.

5. Narrowed Focus for Increased Accuracy:

- **Focused Responses:** By limiting its scope to a small area, the sub LLM can deliver **highly accurate responses** within its niche, as it doesn't need to manage the complexities of a broader domain. It specializes in what it's trained on or retrieves, making it reliable for niche queries.
- **Trade-off in Breadth:** While it excels in providing deep, specialized knowledge, the custom sub LLM may not perform well outside of its niche. Its narrowed training limits its understanding of other areas.

Example: Handling Niche Data in Different Fields:

1. **Legal Field:** A custom sub LLM fine-tuned on a specific area of law, such as **intellectual property law**, becomes highly proficient in answering legal queries about patents, trademarks, and copyrights but might not be as accurate when asked about other legal areas like family or criminal law.
2. **Medical Field:** A custom sub LLM fine-tuned for **rare diseases** can answer detailed questions about treatment protocols, symptoms, and research studies for those diseases. However, it may not handle broader healthcare questions (like general wellness) as effectively.
3. **Financial Markets:** A custom sub LLM focused on **niche investment strategies** could provide detailed responses about specific investment products or techniques in emerging markets. Through RAG, it might pull the latest stock prices, trends, or market analysis, giving real-time advice that the master LLM couldn't provide.

Summary:

- **Fine-Tuning** allows the custom sub LLM to learn deeply about a niche area, mastering its terminology and context.
- **RAG** allows the custom sub LLM to pull real-time data from niche sources, enhancing its accuracy with fresh, external information.
- **Handling Niche Data** requires focusing the sub LLM on specific areas, ensuring precise answers while sacrificing breadth of knowledge in other domains.

This combination makes custom sub LLMs very effective at handling specialized, niche queries, even when the master LLM lacks the depth to do so.

1. Master LLM's Domain Knowledge Assumption:

- When we say that the **master LLM** needs strong **domain knowledge**, it means that for the custom sub LLM (child LLM) to be truly effective, the master LLM should have at least a **foundational understanding** of the relevant domain or subdomain.
- **Why?** If the master LLM is **completely unfamiliar** with the domain or subdomain, even fine-tuning or RAG might struggle because the core language model won't be able to form accurate relationships between words or concepts in that niche. The custom sub LLM will still be reliant on the broader understanding provided by the master LLM's layers and knowledge.

2. Handling Niche Focus without Strong Domain Knowledge:

- **Fine-tuning** or **RAG** can compensate for **specific gaps** in the master LLM's knowledge but only to a certain extent.
 - If the master LLM lacks **basic understanding** of a domain, fine-tuning can give it a **boost** in that niche, but the quality may not be as strong as if the model had been pre-trained with foundational domain knowledge.
 - Similarly, **RAG** can retrieve real-time information from external sources, but without the master LLM's solid grounding in the domain, the relationships between niche terms could still be misunderstood.

3. Limitations of Custom Sub LLMs without a Strong Domain Base:

- **General Understanding vs. Specific Mastery:** The master LLM's general understanding of a domain helps form basic connections between words and concepts that are essential even for very niche areas. If the master LLM doesn't have this foundation, the custom sub LLM may misinterpret certain relationships, especially in complex or technical fields.
- **Example:** In a specialized medical field, if the master LLM lacks any understanding of general medical terms, a sub LLM fine-tuned on a narrow set of rare disease data might have difficulty forming accurate interpretations. It might struggle to connect niche medical terms properly without the broader medical context that the master LLM should ideally provide.

4. Can Fine-Tuning Work without Strong Domain Knowledge?

- **It Depends on the Niche:** If the custom sub LLM's focus is **extremely narrow** and the master LLM doesn't need to know much about the broader domain (e.g., fine-tuning on customer-specific data that is self-contained), it might still perform reasonably well without deep domain knowledge. However, this is **not ideal** for highly specialized or technical fields.
- **For Broader Niche Areas:** In cases where the niche area is connected to a broader domain (e.g., a specific legal field or specialized medical knowledge), the master LLM's understanding of the overarching domain is important for the sub LLM to make accurate connections and provide quality answers.

Conclusion:

- **Niche Focus Requires Domain Support:** While fine-tuning and RAG can enable niche-focused custom sub LLMs, they generally rely on the **master LLM** having at least a **basic understanding** of the domain. The better the master LLM's domain knowledge, the more effective and accurate the custom sub LLM will be.
- **If Domain Knowledge Is Missing:** Fine-tuning or RAG can partially bridge gaps, but there are limitations. A custom sub LLM might misinterpret relationships between niche concepts if the master LLM doesn't have a good grounding in the broader domain.

