Quick Answer: When to Fine-Tune Large Language Models

Published 28 September 2023 - ID G00799244 - 7 min read

By Analyst(s): Leinar Ramos, Tong Zhang, Ben Yan, Afraz Jaffri

Initiatives: Artificial Intelligence; Analytics, BI and Data Science Solutions; Generative AI Resource Center

Although fine-tuning can be used to customize large language models to obtain better performance in a given task or domain, it is not always the optimal approach. IT leaders responsible for AI must use a structured framework to decide when to fine-tune LLMs versus using alternative techniques.

Additional Perspectives

- Summary Translation: Quick Answer: When to Fine-Tune Large Language Models (09 November 2023)

Quick Answer

When should we fine-tune large language models?

- Consider fine-tuning when your goal is to align large language models (LLMs) to specific tasks, improving their performance and better matching the output style and format required. LLMs can also be fine-tuned to increase their knowledge on domain-specific areas. However, creating multi-task, industry-specific models requires more advanced LLM training than what is available in standard fine-tuning market offerings.

- Only fine-tune when prompt engineering and retrieval augmented generation (RAG) do not work. Experiment first with these alternatives, and consider fine-tuning if they are not enough to obtain the performance required for your use case.

- Only fine-tune LLMs if you have adequate training data that is not sensitive and does not have user access restrictions. Do not fine-tune LLMs on data that is frequently changing or on data that will be used only rarely.

More Detail

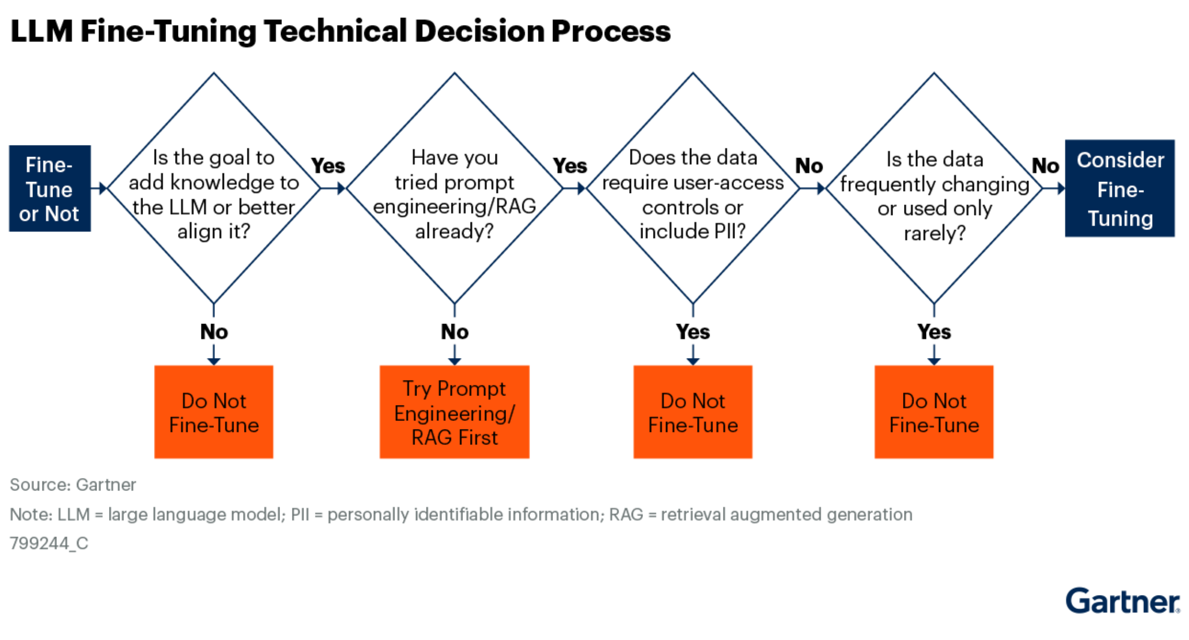
Consider Fine-Tuning When Your Goal Is to Align LLMs to Specific Tasks or Add Domain-Specific Knowledge

Many organizations are looking to create customized large language models that contain their own data and are optimized for specific use cases. Fine-tuning involves taking a pretrained LLM as a starting point and further training it on a new dataset.

LLM fine-tuning is emerging as a viable option, with many vendors offering fine-tuning via their APIs and user interfaces. [1] Similarly, highly customizable LLM open-source models are now available (see Quick Answer: What Are the Pros and Cons of Open-Source Generative AI Models?). However, it is difficult for organizations to know when to consider these new fine-tuning capabilities.

Whether or not to use fine-tuning is a use-case-by-use-case decision. See Figure 1 for a simple process to decide whether to consider fine-tuning or not for a given use case, which summarizes the criteria discussed in this research.

Figure 1: LLM Fine-Tuning Technical Decision Process


LLM Fine-Tuning Technical Decision Process
Fine-Tune or Not
Is the goal to add knowledge to the LLM or better align it?
Yes
No
Do Not Fine-Tune
Have you tried prompt engineering/RAG already?
Yes
No
Try Prompt Engineering/ RAG First
Does the data require user-access controls or include PII?
No
Yes
Do Not Fine-Tune
Is the data frequently changing or used only rarely?
No
Yes
Do Not Fine-Tune
Consider Fine-Tuning
Source: Gartner
Note: LLM = large language model; PII = personally identifiable information; RAG = retrieval augmented generation
799244_C
Gartner

The first step in Figure 1 is to start with the intended goal of fine-tuning. Organizations should consider LLM fine-tuning if they are trying to achieve at least one of the following goals:

- **Task-based alignment:** LLMs are general-purpose models that can exhibit many different capabilities. Fine-tuning can be used to better align LLMs' underlying capabilities with the specific task at hand. LLMs are not only used as conversational assistants or for question-answering use cases. They can also be used for tasks such as classification. In these scenarios, detailed instructions and examples can be used to fine-tuned LLMs and improve task performance.

- **Style-based alignment:** Similarly, fine-tuning can be effective at aligning an LLM to match the specific communication style and tone required, to better moderate conversations, and to output the desired format more consistently. Examples of prompts and completions that exhibit the right style and format are typically used to fine-tune the LLM in this case.

- **Include additional knowledge:** LLMs can be fine-tuned with additional data to increase their knowledge on domain-specific areas that were not in the original LLM training data. However, fine-tuning is more useful for performing light updates to the LLM knowledge. This contrasts with an approach for creating domain-specific or company-specific general LLMs that are trained with extensive collections of corporate documents. The latter would require substantially more effort and investment.

Current LLM fine-tuning market offerings are geared toward improving task-specific performance or performing light updates with additional domain knowledge. The creation of high-performing, multi-task, industry-specific models requires more advanced forms of LLM training and much more investment and expertise.

Only Fine-Tune When Other Alternatives Do Not Work

Fine-tuning is not the only alternative to create custom applications that align LLMs and incorporate new domain-specific knowledge, so be sure to understand the alternatives when deciding whether or not to use fine-tuning for each use case.

There are two main alternatives to fine-tuning (see How to Choose an Approach for Deploying Generative AI):

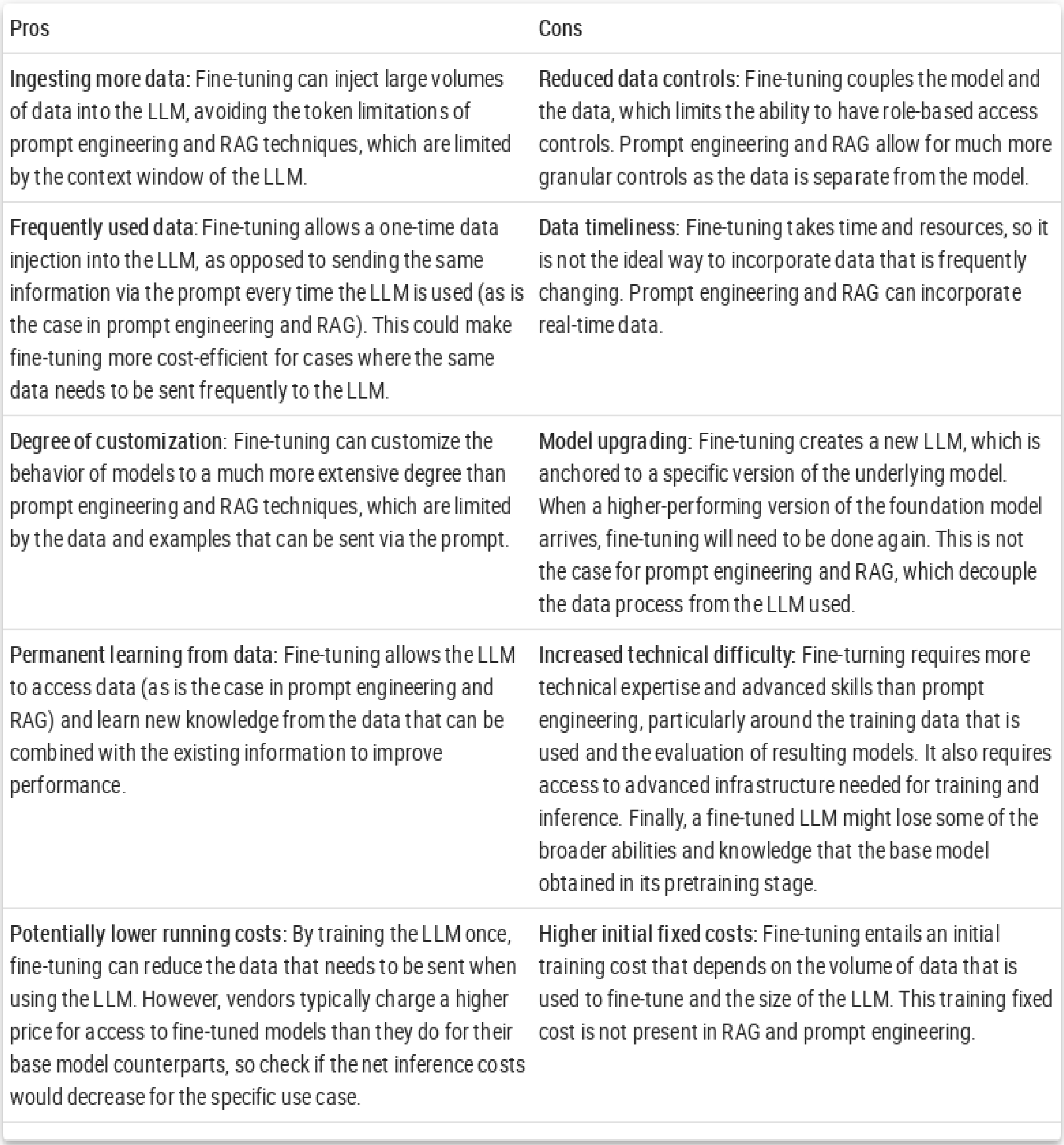
- Prompt engineering provides data to the LLMs to specify and confine the set of responses they can produce. This can produce a desired outcome without updating the model parameters (as is done with fine-tuning).

- RAG is a specific form of prompt engineering that uses search functionality to automatically find relevant data outside of LLM. It incorporates the data into the prompt for an LLM to ground the generative output with factual and new information.

Fine-tuning presents some advantages and disadvantages when compared to prompt engineering or RAG. See Table 1 for a list of the main pros and cons that should be considered.

Table 1: Fine-tuning When Compared to Prompt Engineering or RAG

(Enlarged table in Appendix)



Pros	Cons
Ingesting more data: Fine-tuning can inject large volumes of data into the LLM, avoiding the token limitations of prompt engineering and RAG techniques, which are limited by the context window of the LLM.	**Reduced data controls:** Fine-tuning couples the model and the data, which limits the ability to have role-based access controls. Prompt engineering and RAG allow for much more granular controls as the data is separate from the model.
Frequently used data: Fine-tuning allows a one-time data injection into the LLM, as opposed to sending the same information via the prompt every time the LLM is used (as is the case in prompt engineering and RAG). This could make fine-tuning more cost-efficient for cases where the same data needs to be sent frequently to the LLM.	**Data timeliness:** Fine-tuning takes time and resources, so it is not the ideal way to incorporate data that is frequently changing. Prompt engineering and RAG can incorporate real-time data.
Degree of customization: Fine-tuning can customize the behavior of models to a much more extensive degree than prompt engineering and RAG techniques, which are limited by the data and examples that can be sent via the prompt.	**Model upgrading:** Fine-tuning creates a new LLM, which is anchored to a specific version of the underlying model. When a higher-performing version of the foundation model arrives, fine-tuning will need to be done again. This is not the case for prompt engineering and RAG, which decouple the data process from the LLM used.
Permanent learning from data: Fine-tuning allows the LLM to access data (as is the case in prompt engineering and RAG) and learn new knowledge from the data that can be combined with the existing information to improve performance.	**Increased technical difficulty:** Fine-turning requires more technical expertise and advanced skills than prompt engineering, particularly around the training data that is used and the evaluation of resulting models. It also requires access to advanced infrastructure needed for training and inference. Finally, a fine-tuned LLM might lose some of the broader abilities and knowledge that the base model obtained in its pretraining stage.
Potentially lower running costs: By training the LLM once, fine-tuning can reduce the data that needs to be sent when using the LLM. However, vendors typically charge a higher price for access to fine-tuned models than they do for their base model counterparts, so check if the net inference costs would decrease for the specific use case.	**Higher initial fixed costs:** Fine-tuning entails an initial training cost that depends on the volume of data that is used to fine-tune and the size of the LLM. This training fixed cost is not present in RAG and prompt engineering.

Source: Gartner (September 2023)

It is possible to do prompt engineering or RAG on top of fine-tuned models, so these options are not fully mutually exclusive.

However, given the trade-offs described in Table 1, pilot and iterate with prompt engineering and RAG first, and only fine-tune if these alternatives are not enough to obtain the performance required.

Only Fine-Tune LLMs If You Have Adequate Training Data

The training data used for fine-tuning needs to closely align to the specific fine-tuning goal (task or style alignment or knowledge acquisition) and presented in the correct format for fine-tuning.

Typically, LLM fine-tuning requires creating prompt-completion pairs (for instance, question-answer pairs) that are then used to train the LLM. This dataset can be relatively small (compared to datasets needed when training AI models from scratch) but needs to be of very high quality, since the LLM will be trained to produce outputs that closely resemble completions in the training data.

Similarly, as the model's responses cannot be effectively controlled by permission or role-based access, the data must not contain any information that should not be given to an end user of the model. The data also should not be subject to frequent change or updates as there are more suitable methods available for this scenario as described in the previous section.

LLM fine-tuning is a fast-moving research field with a wide range of emerging techniques. In particular, parameter-efficient fine-tuning (PEFT) techniques are able to reduce the cost and time required to fine-tune by changing only a fraction of the LLM parameters. [2]

Regardless of the fine-tuning techniques used, the decision to fine-tune LLMs or not should follow the considerations described in this document and summarized in Figure 1, which represent a set of criteria that applies independently of the specific techniques used.

Recommended by the Authors

How to Choose an Approach for Deploying Generative AI

Quick Answer: Options for Using Your Data With Generative AI Models

Evidence

[1] GPT-3.5 Turbo fine-tuning and API updates, OpenAI.

[2] Parameter-efficient fine-tuning (PEFT) techniques include the use of Low-Rank Adaptation (LoRA) weights, prompt tuning, adapters and many other techniques. See Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning (University of Massachusetts Lowell) for a survey of existing PEFT approaches.

Table 1: Fine-tuning When Compared to Prompt Engineering or RAG

Pros	Cons
Ingesting more data: Fine-tuning can inject large volumes of data into the LLM, avoiding the token limitations of prompt engineering and RAG techniques, which are limited by the context window of the LLM.	**Reduced data controls:** Fine-tuning couples the model and the data, which limits the ability to have role-based access controls. Prompt engineering and RAG allow for much more granular controls as the data is separate from the model.
Frequently used data: Fine-tuning allows a one-time data injection into the LLM, as opposed to sending the same information via the prompt every time the LLM is used (as is the case in prompt engineering and RAG). This could make fine-tuning more cost-efficient for cases where the same data needs to be sent frequently to the LLM.	**Data timeliness**: Fine-tuning takes time and resources, so it is not the ideal way to incorporate data that is frequently changing. Prompt engineering and RAG can incorporate real-time data.
Degree of customization: Fine-tuning can customize the behavior of models to a much more extensive degree than prompt engineering and RAG techniques, which are limited by the data and examples that can be sent via the prompt.	**Model upgrading:** Fine-tuning creates a new LLM, which is anchored to a specific version of the underlying model. When a higher-performing version of the foundation model arrives, fine-tuning will need to be done again. This is not the case for prompt engineering and RAG, which decouple the data process from the LLM used.
Permanent learning from data: Fine-tuning allows the LLM to access data (as is the case in prompt engineering and RAG) and learn new knowledge from the data that can be combined with the existing information to improve performance.	**Increased technical difficulty:** Fine-turning requires more technical expertise and advanced skills than prompt engineering, particularly around the training data that is used and the evaluation of resulting models. It also requires access to advanced infrastructure needed for training and inference. Finally, a fine-tuned LLM might lose some of the broader abilities and knowledge that the base model obtained in its pretraining stage.
Potentially lower running costs: By training the LLM once, fine-tuning can	**Higher initial fixed costs:** Fine-tuning entails an initial training cost that

reduce the data that needs to be sent when using the LLM. However, vendors typically charge a higher price for access to fine-tuned models than they do for their base model counterparts, so check if the net inference costs would decrease for the specific use case.	depends on the volume of data that is used to fine-tune and the size of the LLM. This training fixed cost is not present in RAG and prompt engineering.

Source: Gartner (September 2023)