RAFT: A new way to teach LLMs to be better at RAG

By Cedric Vidal

Published Mar 15 2024 10:19 AM 74.7K Views

RAFT: A new way to teach LLMs to be better at RAG

“Retrieval-Augmented Fine-Tuning” combines the benefits of Retrieval-Augmented Generation and Fine-Tuning for better domain adaptation

By Cedric Vidal, Principal AI Advocate, Microsoft
And Suraj Subramanian, AI Advocate, Meta

Gorilla Student passing exam (Generated using DALL-E 3)

Introduction

One of the most impactful applications of generative AI for businesses is to create natural language interfaces that have access to existing knowledge. This means answering questions about specific domains such as banking, legal and medical. There are currently two main ways to do this. First: domain-specific Fine-tuning (DSF), which means training an existing base model on a set of documents that represent the domain specific knowledge. Second: RAG (Retrieval Augmented Generation), which involves storing those documents in a vector database and (at query time) finding documents based on their semantic similarity with the question and bringing them into the context of the LLM for in context learning.

In this article, we will look at the limitations of those two approaches and how a team of UC Berkeley researchers, Tianjun Zhang and Shishir G. Patil, may have just discovered a better approach. The team previously known for Gorilla LLM :gorilla: presents this new approach in their RAFT paper (Retrieval Augmented Fine Tuning) showing how they used Meta Llama 2 and Azure AI Studio to conduct their research and implement their approach.

The Berkeley team also published a blog post about the paper explaining what those advantages and disadvantages are and how the RAFT approach produces more effective results. The RAFT paper implementation is available in their Github repository.

Let’s start by giving an overview of how the RAFT approach works.

Understanding the RAFT method

In conventional RAG, when a query is posed to a model, it retrieves a few documents from an index that are likely to contain the answer. It uses these documents as the context to generate an answer to the user’s query.

With fine-tuning, the model answers queries like a student writing a ***closed-book exam.*** With RAG, this scenario resembles an ***open-book exam***, where the student has full access to a textbook to find the answers. Open-book exams are easier to solve than closed-book exams, which explains the efficacy and popularity of RAG.

Both approaches have limitations. With fine-tuning, the model is not only limited to what it has been trained on, but it is also subject to approximation and hallucination. With RAG, the model is grounded but documents are retrieved merely on their semantic proximity with the query. The model doesn't know which documents are truly relevant or are just red herrings. These "distractor" documents may be pulled into the model's context even when they are not good sources for a well-reasoned answer.

Tianjun and Shishir were looking to improve these deficiencies of RAG. They hypothesized that a student who studies the textbooks before the open-book exam was likely to perform better than a student who studies the textbook. Translating that back to LLMs, if a model "studied" the documents beforehand, could that improve its RAG performance? Their approach – Retrieval Augmented Fine Tuning – attempts to get the model to study or adapt to a domain before it is used in a RAG setup.

Using Meta Llama 2 7B language model, they first prepare a synthetic dataset where each data sample consists of:

- A question,
- A set of documents to refer to (including documents containing relevant information and documents that do not contain any relevant information to answer the question and therefore can safely be ignored),
- An answer generated from the documents,
- A Chain-of-Thought explanation including excerpts from the relevant documents (generated by a general purpose LLM such as GPT-4, or Llama 2 70B)

This dataset is used to fine-tune the Llama 2 7B model using standard supervised training. The model is now better adapted to the domain; it not only aligns its tone and voice to the domain dataset but is also better at extracting the useful bits of information from the retrieved context. The addition of Chain-of-Thought reasoning prevents overfitting and improves training robustness.

RAFT sits in the middle-ground between RAG and domain-specific SFT. It simultaneously primes the LLM on domain knowledge and style (a la DSF), while improving the quality of generated answers from the retrieved context. Since pretrained models like Llama 2 are trained on a diverse set of domains, techniques like RAFT can make it better suited for niche areas like healthcare or legal datasets.

The RAFT team answers questions

Cedric and Suraj had the opportunity to sit down with Tianjun and Shishir and ask them a few questions about their work on RAFT.

Question: Why did you choose Llama 2 7B?

Answer: We chose Llama 2 7B because we focus on RAG tasks, where the task requires a combination of the model's ability to reason, understand language, have lower-latency inference, and be easily adaptable to diverse settings. Llama 2 7B fit the bill well- it's a good base model for a lot of the general-knowledge, question-answering tasks, with encouraging math skills, and the ability to parse reasonably long documents due to its

4096k pre-training. Llama 2 7B is also a perfect model for training on 4 A100-40G GPUs and serving on a single GPU. Thereby in the pareto curve or performance, ease-of-deployment, and with the right licensing, the Llama 2 model is quite apt for the RAFT task. With the help of Microsoft AI studio, we are happy to explore Llama 2 13b or 70b as well.

Question: What recommendations do you have for people trying to fine-tune Llama? Any best practices you learnt on the field with fine-tuning LLMs?
Answer: Fine-tuning Llama is usually a complex task involving data collection, data cleaning and actual fine-tuning. In terms of data, we recommend collecting diverse questions with respect to your domain and constructing chain-of-thought (CoT) answers (also talked about in our RAFT paper). We also recommend you store intermediate checkpoints, which would then help with early stopping. It is also critical to have the fine-tuning learning rate set to at least a magnitude lower than what was used for pre-training. Other than this, the usual best-practices of 16-bit precision, not training for more than 3 epochs, using large-batch sizes are also recommended.

Question: Should the fine-tuning be applied to each domain? Or is the fine-tuned model better at RAG on multiple domains in general?
Answer: The fine-tuned model's performance is dependent on the domain (documents it is trained on) for knowledge but can generalize across domains for behavior to a certain extent. There is a slight tradeoff between accuracy vs. generalization. Usually fine-tuning for a domain is a good practice, but fine-tuning for a limited set of enterprise docs may bring better performance since the knowledge is strictly narrower.

Question: What did you think about the Azure AI Studio Fine-tuning system?
Answer: The Azure AI fine-tuning system is very user-friendly, from training data uploading, to hyperparameter selection, to deploying the trained models, everything is easy to use.

Question: What are the benefits of AI Studio Fine-tuning?
Answer: The biggest benefit is that you do not need to worry about GPUs; Do not need to handle training platforms; Do not need to worry about model deployment; One click, easy to use and the performance is great!

Question: What do you think could be improved in AI Studio Fine-tuning?
Answer: As a researcher, it would be interesting if the developer can find additional peak or insights into the exact fine-tuning recipe happening inside the system (e.g., if it is Lora or full-parameter fine-tuning, how many GPUs has been used, what's the different hyperparameters for LoRA, etc)!

Question: What do you think AI Studio Fine-tuning changes for the industry?
Answer: This could enable the easy fine-tuning and deployment of LLMs for enterprises, greatly enabling the deployments of custom models for different enterprises.

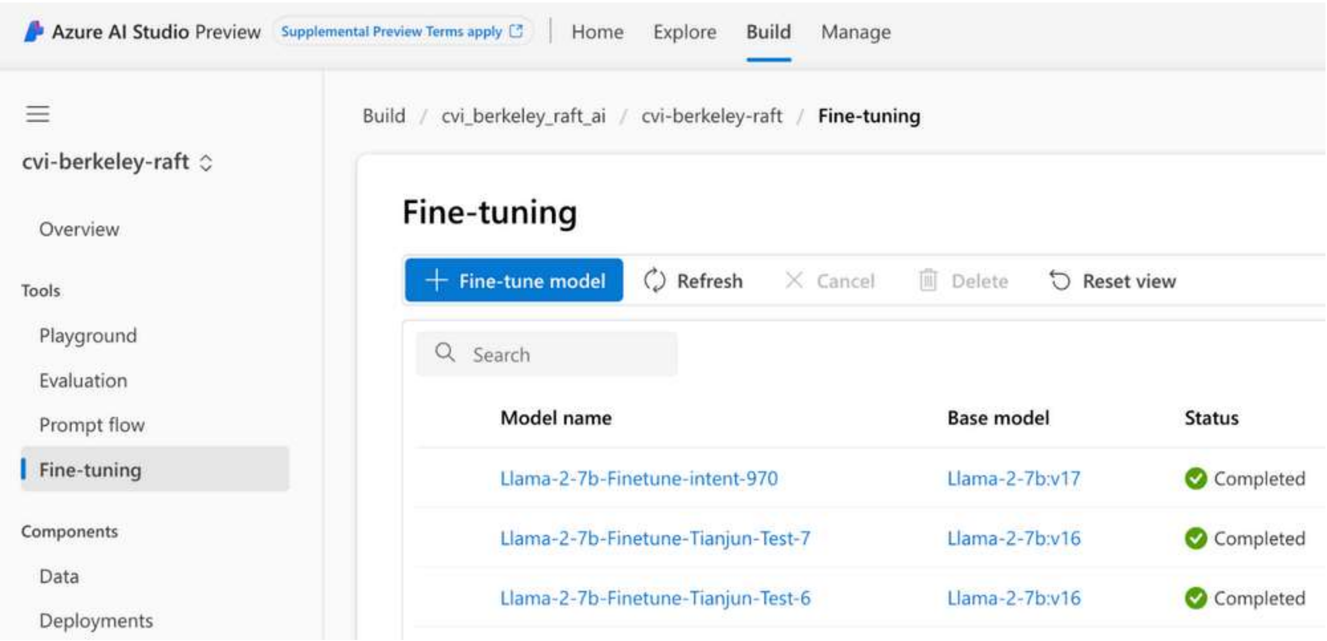
Azure AI Studio Fine-tuning

The Berkeley team (aka.ms/raft-repo) to fine-tune Meta Llama 2 for their RAFT paper using MaaS (Model as a Service) in Azure AI Studio.

MS Learn has a tutorial explaining how to Fine-tune a Llama 2 model in Azure AI Studio.

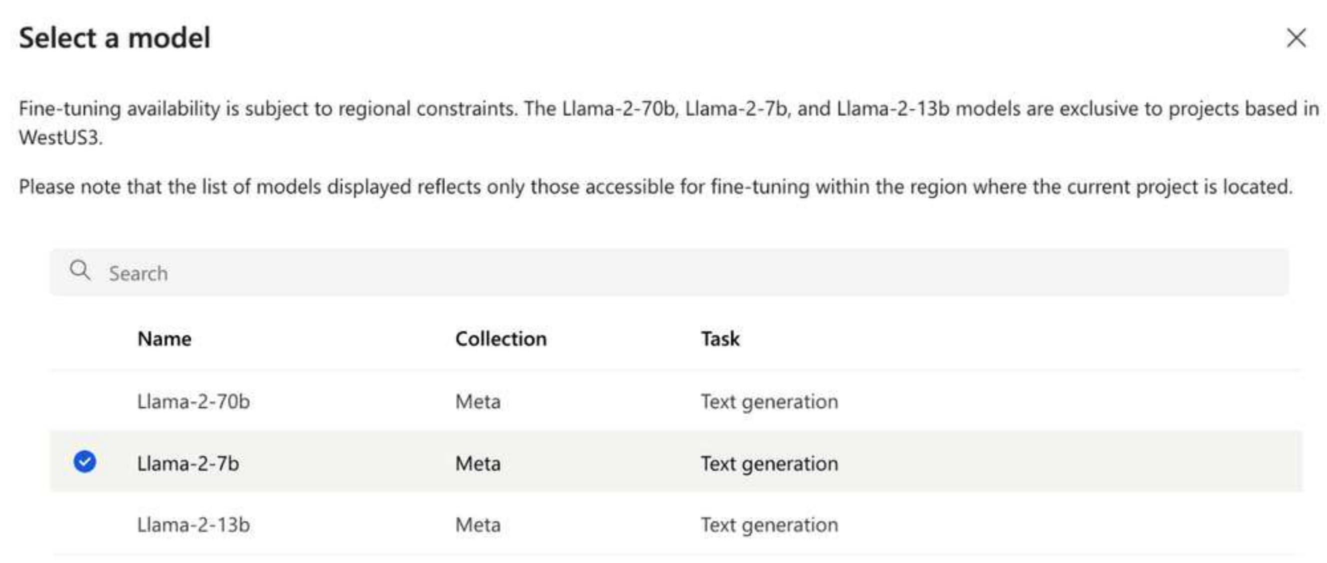
So far, fine-tuning has been reserved for ML engineers with excellent understanding of the latest advancements in generative AI, Python, ML frameworks, GPUs and Cloud infrastructure. Azure AI Studio is a game changer: it automates all the technicalities, infrastructure and fine-tuning ML framework setup to focus on the data preparation.

All it takes is opening the AI Studio 's Fine-tuning wizard [1].



Model name	Base model	Status
Llama-2-7b-Finetune-intent-970	Llama-2-7b:v17	✓ Completed
Llama-2-7b-Finetune-Tianjun-Test-7	Llama-2-7b:v16	✓ Completed
Llama-2-7b-Finetune-Tianjun-Test-6	Llama-2-7b:v16	✓ Completed

You select which model you want to fine-tune:


Select a model

Fine-tuning availability is subject to regional constraints. The Llama-2-70b, Llama-2-7b, and Llama-2-13b models are exclusive to projects based in WestUS3.

Please note that the list of models displayed reflects only those accessible for fine-tuning within the region where the current project is located.

	Name	Collection	Task
	Llama-2-70b	Meta	Text generation
●	Llama-2-7b	Meta	Text generation
	Llama-2-13b	Meta	Text generation

You upload the training dataset JSONL file (JSON Lines format):