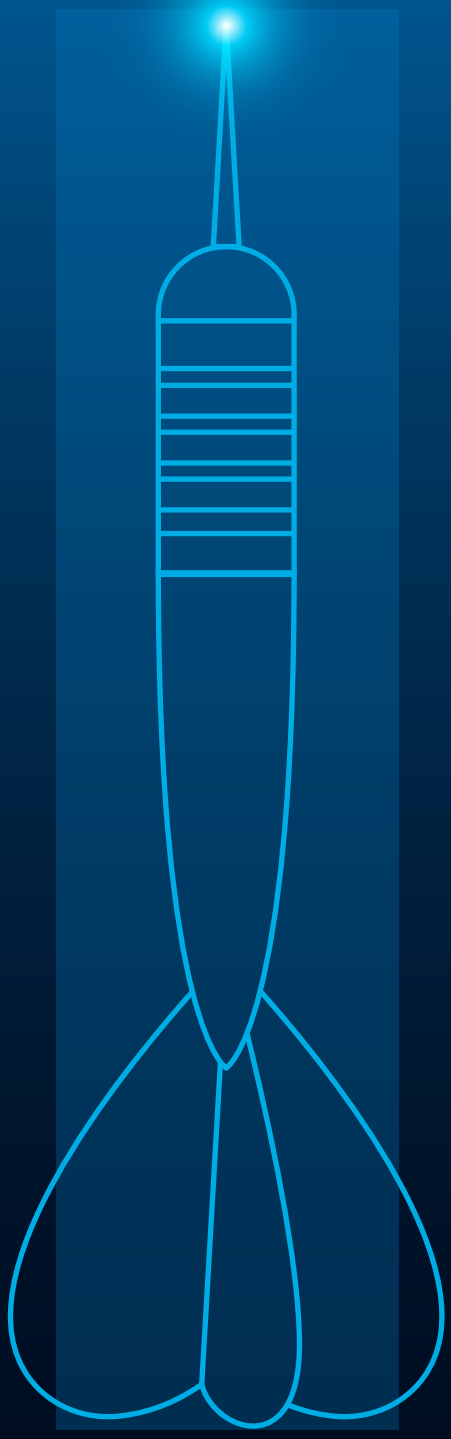
Infosys topaz

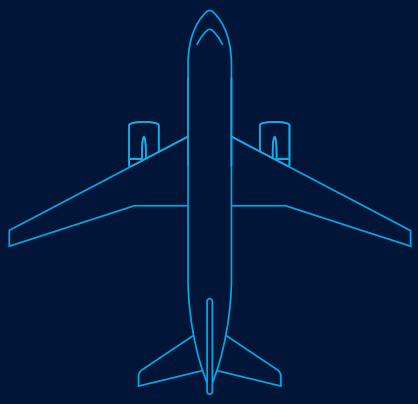
VIEW POINT



FINE-TUNED LARGE LANGUAGE MODELS FOR SPECIALIZED TASKS

Fine-tuned LLMs for specialized tasks are highly effective and can be developed at a fraction of the cost and time that conventional approaches take.

Infosys®
Navigate your next



There are two different approaches to using large language models (LLMs). One is to scale up the model size and increase the performance of general-purpose models that can handle various tasks. Large companies and AI startups are competing via this approach to build the biggest and most efficacious models such as GPT-4 with over a trillion parameters. The other approach is to scale down the model size and fine-tune open access models for specific domains and tasks. Despite its current lack of popularity, Infosys has used this approach successfully and believes enterprises will follow suit for customized and cost-effective solutions with the requisite data privacy and security.

Both scale-up and scale-down approaches have their respective advantages and objectives. They both work with a base of closed-access models and open-access models.

Training data, model weights, architecture	Open	Closed
Examples	Falcon, BLOOM, Whisper	OpenAI GPT 4.0, Anthropic
Pricing	Free	Pay-per-use
Skillsets	Programming (high), machine learning (high)	Programming (high), prompt engineering (high)
Time to build use cases	High	Medium to low
Infrastructure	High	Minimum
License	Non-commercial use/Apache 2.0	Only API calls
Type of AI roadmap	Narrow AI	Artificial general intelligence (AGI)
Use cases	Business-driven	Consumer-driven
Purpose	Task or domain-specific	General purpose
Internal tools (e.g., MLOps)	Very important	Not important
Dominant usage pattern	Fine-tuned for specific needs with private data	Retrieval augmented generation (RAG)

Figure 1: Open-access models versus closed-access models

Source: Infosys

Big and powerful models, often proprietary, are good for retrieval-augmented generation (RAG) and are used in business applications such as dialogue systems, semantic search, question answering, and summarization. They do not require any model adaptation. However, for specialized tasks where customization and cost are important, fine-tuning of open access models is the path to success. Open-access models are best suited for auto-completion tasks such as code completion and machine translation. These models are much more efficient and effective when they are fine-tuned with instructions using supervised learning or through extended pre-training with self-supervised learning. Figure 2 compares RAG and other model optimization methods.

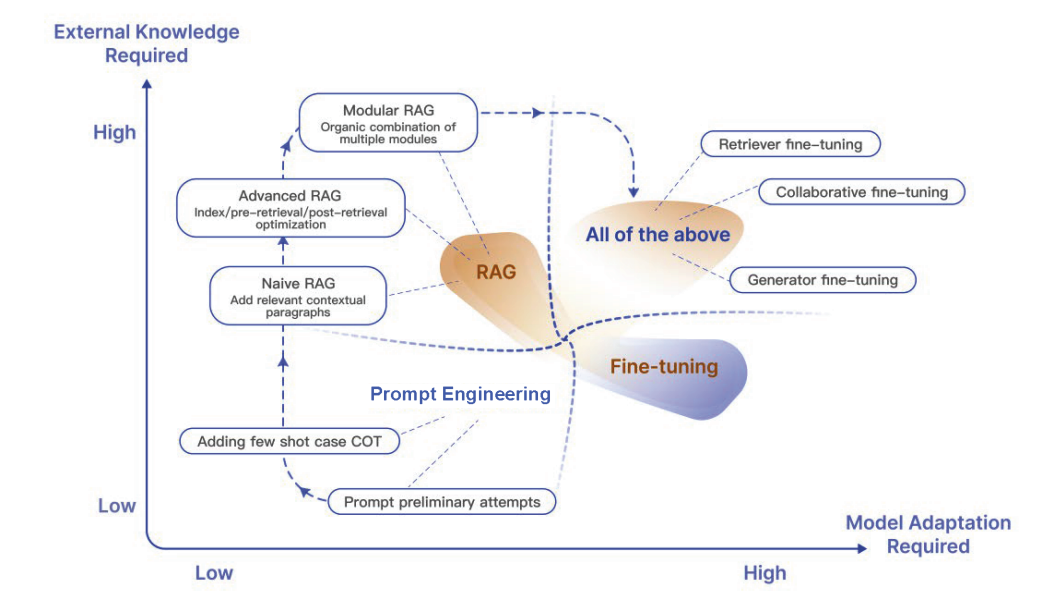


Figure 2: Fine-tuning versus RAG Source: Retrieval Augmentation Generation Ushers in a New Generation of Advanced AI

Creating LLMs is difficult, expensive, and time consuming. They can take months to train on hundreds or thousands of the most advanced GPUs and then need as many GPUs to deploy them. LLMs also pose significant challenges when it comes to performing inference with them, requiring multiple GPUs at the datacenter level, and backend techniques such as weight streaming to enable them.

Hence, enterprises are turning to smaller open-source LLMs customized for specific tasks. For example, with developers increasingly relying on code assistance, models such as StarCoderBase [1] or other advanced code models are being used as a basis to fine-tune LLMs with organization-specific code and data. This results in better productivity gains compared to generic code assistance. A specialized fine-tuned large language model built at Infosys is 20% more accurate than generalized code assistants.

Fine-tuning Large Language Models

LLMs can be fine-tuned using supervised/unsupervised fine-tuning, instruction-tuning, or reinforcement-learning with human feedback (RLHF). Based on our experience, the true potential of fine-tuning lies in self-supervised fine-tuning (extended pre-training) as shown in Figure 3. This is a niche area that the industry is struggling with for want of enough knowledge about the process, the required tools, and the choice of the right base model.

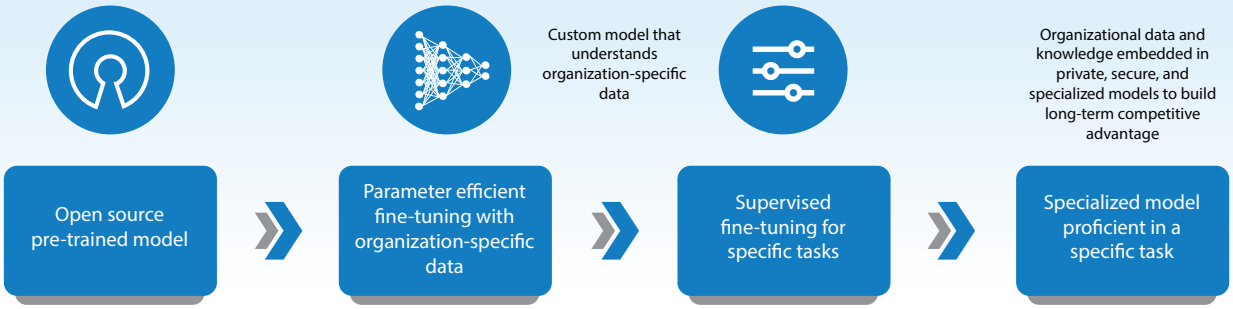


Figure 3: Illustration of fine-tuning of code model Source: Infosys

The base models for fine-tuning must be commercially permissible, and "truly open" by revealing the data they were trained on, weights, parameters, and 360 checkpoints at the very least. Without this information, adjusting the base model with the resources can be very challenging. Projects such as LLM360 Open-source Initiative [2] are a leap forward in the area as they share everything – weights, data, code, h-parameters, 360 checkpoints, exact data sequence, mixing, processing, and evaluation to enable fine-tuning and thereby cut down on unnecessary work and carbon emissions. Examples of highly successful fine-tuned models include Replit's code generation model [3], Vicuna a "decarbonized" high-performance English LLM [4], and Infosys' fine-tuned model for Finacle, one of the world's largest digital banking products.

The Future of LLMs

Generative AI is undergoing significant change in the way large language models (LLMs) work. We are moving from a situation where one LLM provides answers based on its vast knowledge base, to a scenario where multiple LLMs, each with different skill sets, work together to generate answers. This change marks the shift from a single LLM model to a flexible agent model [5] [6]. As this future develops, it is clear that conventional prompt engineering based on uniform agents will not be enough to provide reliable answers. Instead, we expect these agents to be supported by various LLMs. These supporting LLMs will be smaller in size, but carefully fine-tuned for specific tasks or domains, making them more effective. In this context, instruction tuning will become a vital factor. It will influence the behavior and answers of these agents, making sure they are skilled at dealing with the specific tasks and domains to which they are assigned. This transition will not merely increase the accuracy of AI answers but also widen the range of applications.

References

1. StarCoderBase - bigcode/starcoderbase · Hugging Face
2. LLM360 Open-source Initiative - LLM360 | Open-source LLMs for Transparency, Trust, and Collaborative Research
3. Replit's Code Model - replit/replit-code-v1_5-3b · Hugging Face
4. Vicuna - lmsys/vicuna-13b-v1.5 · Hugging Face
5. Reasoning with Language Model is Planning with World Model - [2305.14992] Reasoning with Language Model is Planning with World Model (arxiv.org)
6. Large Language Model based Multi-Agents: A Survey of Progress and Challenges - [2402.01680] Large Language Model based Multi-Agents: A Survey of Progress and Challenges (arxiv.org)

For more information, contact askus@infosys.com

Infosys®
Navigate your next