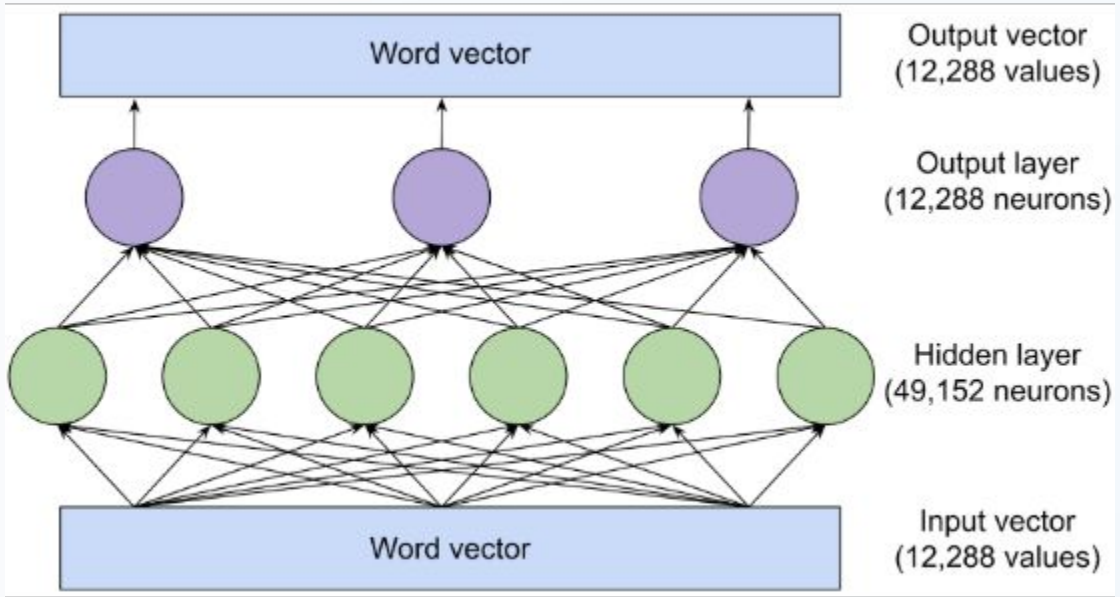
LLM Post-Training Lab: Building a Domain-Specific AI Tutor

By Hiteshwari Patel | August 2026

What is an LLM?

- At its core, a Large Language Model (LLM) is simply a massive statistical engine trained to predict the next logical token in a sequence.
- It achieves this capability by processing internet-scale text during its initial, highly intensive creation phase.
- However, raw predicting power alone is not enough. Without further refinement, it generates text but does not natively understand how to act as a helpful assistant.



The Pretraining Foundation

What the Base Model Knows

Before any post-training, the model possesses a vast understanding of general human language syntax, grammar, and broad world knowledge. It holds generalized reasoning capabilities encoded deeply within its billions of parameters.

The Computational Cost

This phase requires tens of thousands of GPUs running for months. It is the most expensive and foundational stage of AI development, which is why most practitioners download open weights rather than pretraining from scratch.

The Post-Training Gap

Pretrained Base Behavior

User Input: "How do I explain gradient descent?"

```
...is a popular question asked by computer science
students. In this article, we will list 10 textbook
definitions of optimization...
```

Result: Completes the document text instead of answering you.

Target Assistant Behavior

User Input: "How do I explain gradient descent?"

> Great question! Think of gradient descent like walking down a foggy mountain. You feel the slope with your feet and take steps downhill...

Result: Follows instructions, acts like a tutor, and explains clearly.

End-to-End Post-Training

1. Continued Pretraining

Injecting highly specific domain knowledge and vocabulary into the model.

2. Supervised Fine-Tuning

Teaching the model to behave like an assistant through instruction pairs.

3. DPO / Alignment

Aligning the model's outputs with exact human preferences and safety.

4. Evaluation

Benchmarking the final model against a fixed, held-out test set.

Continued Pretraining

What It Is & Why We Use It

- Continues causal next-token prediction on a domain-specific corpus (e.g., machine learning literature, textbooks).
- Adapts internal weights to specialized vocabulary, acronyms, and technical domain fluency.

What It Is NOT

- It is NOT instruction tuning. The model still generates raw continuous text.
- It will NOT know how to pause and answer user questions directly yet.

Supervised Fine-Tuning

Learning to Act Like an Assistant

Supervised Fine-Tuning (SFT) is the critical transformation stage in post-training.

CORE METHODOLOGY

Instruction → Response

We train the model strictly on structured input-output pairs to align its response patterns.

The Behavioral Shift

Instead of rambling or completing a prompt with another question, SFT forces the model to learn the behavioral pattern of answering the user's prompt directly, concisely, and helpfully.

This targeted behavioral alignment is precisely where our model transitions to become a specialized **"Tutor"**.

Low-Rank Adaptation (LoRA)

The Parameter Problem

Updating all 3 billion parameters of Llama-3.2 requires immense memory overhead. Full fine-tuning is entirely unfeasible on consumer or free-tier hardware.

The LoRA Solution

We freeze the base model and inject small, trainable low-rank matrices. We train <1% of parameters, resulting in adapters that are mere megabytes in size.

The Mathematical Intuition

Instead of learning a full dense weight update, we decompose the update into two smaller matrices:

$$W = W_0 + \Delta W = W_0 + BA$$

Where W0 remains fixed, drastically cutting GPU VRAM requirements.

QLoRA

4-Bit

Quantized Base Model

Unlocking Free-Tier Fine-Tuning

QLoRA combines the parameter efficiency of LoRA with the extreme memory savings of 4-bit NormalFloat quantization for the frozen base weights.

This breakthrough is exactly what makes fine-tuning a powerful 3B+ parameter model entirely possible on a free, single Colab GPU instance without hitting Out-Of-Memory errors.

Preference Alignment

SFT Output: "Plausible"

Prompt: Explain matrix multiplication.

SFT Answer: Matrix multiplication is a binary operation that produces a matrix from two matrices. For matrix A and B, the element c_ij is computed as...
(Factually accurate, but dry, dense, and not ideal for a tutor.)

Aligned Output: "Preferred"

Prompt: Explain matrix multiplication.

Aligned Answer: Imagine you are tracking inventory costs. Matrix multiplication is just a structured way to multiply prices by quantities across multiple items at once...
(Engaging, accessible, and vastly superior for our specific use case.)

RLHF vs. DPO

Property	Classic RLHF (PPO)	DPO (Direct Preference Opt.)
Reward Model	Requires a separate trained model	None required (Implicit in loss)
Training Pipeline	Complex (Rollouts, Actor, Critic)	Simple (Classification-style loss)
Hardware Cost	Very High (3+ models in VRAM)	Low (One reference, one policy)
Stability	Prone to reward hacking, unstable	Stable, acts like SFT extension

Llama-3.2 Project Setup

- **Base Architecture:** We selected Meta's Llama-3.2-3B-Instruct loaded in 4-bit precision via BitsAndBytes.
- **Continued Pretraining Data:** A custom unstructured text corpus consisting of cutting-edge LLM post-training whitepapers and documentation.
- **SFT Dataset:** Curated synthetic multi-turn dialogues where the "Tutor" answers LLMs questions concisely and accurately.
- **DPO Preference Data:** Triples containing a prompt, a chosen insightful answer, and a rejected hallucinated or overly verbose answer.

Evaluation Methodology

- To ensure a fair comparison, we use a fixed, held-out set of 100 complex domain questions.
- We evaluate three checkpoints: The Base Model, the SFT Checkpoint, and the final DPO Checkpoint.
- Because we use LoRA, we only load the heavy base model into VRAM once. We simply swap the megabyte-sized adapter layers dynamically during inference to generate answers for all three stages.

Takeaways & What's Next

- **Scaling to RLHF:** While DPO is efficient, experimenting with standard PPO and a robust Reward Model could push our performance ceiling even higher.
- **LLM-as-a-Judge:** Implementing automated evaluation loops using a larger model (e.g., Llama-3-70B) to score our iterative training runs without manual labeling.
- **Deployment:** Merging our LoRA adapters back into the base weights, applying final hardware quantization, and serving the Tutor via a live chat interface.
- **Try it yourself:** All scripts, datasets, and the Colab notebook will be uploaded to GitHub for the community to replicate.