Fine-Tuning and Small Language Models

MGMT 675: Generative AI for Finance

Kerry Back

Beyond RAG

- RAG injects knowledge at query time without changing the model
- But what if you need the model itself to behave differently?
- Two approaches, in order of increasing effort and control:
    1. **Fine-tuning** — adjust a pre-trained model's weights
    2. **Training a small language model** — build from scratch on your data

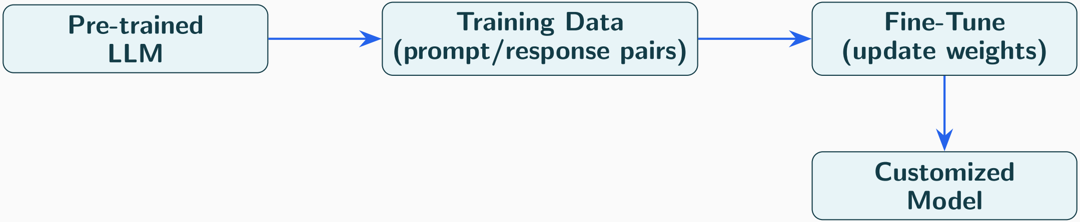
Fine-Tuning

What is Fine-Tuning?

Fine-tuning = take a pre-trained LLM and continue training it on a smaller, task-specific dataset. The model's weights are updated to reflect new patterns.

- Starts from a capable base model (e.g., GPT-4, Llama, Mistral)
- Additional training on curated examples teaches style, format, or domain knowledge
- The result is a customized model that "just knows" your domain

How Fine-Tuning Works



- Training data: hundreds to thousands of example prompt/response pairs
- Often uses parameter-efficient methods (LoRA) — only a small fraction of weights are updated

Fine-Tuning in Finance

- **Report generation** — train the model to produce reports in your firm's style and format
- **Sentiment analysis** — fine-tune on labeled financial text (earnings calls, news)
- **Classification** — categorize transactions, flag compliance issues
- **Code generation** — specialize for your internal tools, databases, or APIs
- **Client communication** — match your firm's tone and terminology

Fine-Tuning: Strengths and Limitations

Strengths

- Domain knowledge baked into the model
- Consistent style and format without long prompts
- Often faster inference (no retrieval step)
- Can improve accuracy on specific tasks

Limitations

- Requires curated training data
- Model can "forget" general capabilities (catastrophic forgetting)
- Expensive to retrain as data changes
- Can hallucinate confidently on topics outside training

Training Small Language Models

What is a Small Language Model?

A **small language model (SLM)** is a language model trained from scratch (or heavily adapted) on a focused corpus. It trades general capability for efficiency and domain specificity.

- Typically 1–10 billion parameters (vs. hundreds of billions for frontier LLMs)
- Trained on domain-specific data: financial filings, legal documents, medical records, etc.
- Can run on modest hardware — even a single GPU or CPU

Why Train a Small Model?

- **Privacy** — data never leaves your infrastructure
- **Cost** — much cheaper to run than large cloud-hosted models
- **Speed** — low latency for real-time applications
- **Control** — full ownership of the model and its behavior
- **Specialization** — a small model trained on your data can outperform a general LLM on your specific tasks

SLMs in Finance

- **BloombergGPT** — 50B parameter model trained on financial data (news, filings, Bloomberg terminal data)
- **FinGPT** — open-source financial LLM for sentiment analysis, forecasting
- **Internal models** — banks training proprietary models on transaction data, risk reports, internal communications
- **On-device models** — running locally for privacy-sensitive tasks (client data, trading signals)

SLMs: Strengths and Limitations

Strengths

- Full data privacy and control
- Low inference cost at scale
- Can excel at narrow tasks
- No vendor lock-in

Limitations

- Requires significant ML expertise
- Large training data requirements
- Limited general reasoning ability
- Ongoing maintenance burden

Hands-On: Training a Tiny Language Model

Why Train One Yourself?

- Training a model yourself — even a tiny one — builds intuition for how LLMs work
- Key concepts become concrete: tokenization, attention, loss, overfitting
- A model with <1M parameters can train on your laptop's CPU in minutes

Belcak et al. (2025) argue that small, specialized models are more economical and often sufficient for agentic AI tasks that are performed repetitively.

https://arxiv.org/abs/2506.02153

Resources: Karpathy's Zero to Hero

- Andrej Karpathy's video lecture series builds a GPT from an empty file
- The key lecture: *Let's build GPT: from scratch, in code, spelled out* (2 hrs)
- Covers attention, positional encoding, layer normalization, training loops
- Course page: `https://karpathy.ai/zero-to-hero.html`

We will try two exercises:

1. **nanoGPT on Shakespeare** — a quick demo you can run in minutes
2. **GPT from scratch in a notebook** — a deeper, step-by-step walkthrough

Exercise 1: nanoGPT on Shakespeare

Train a ∼0.8M-parameter character-level GPT on Shakespeare's complete works using Karpathy's **nanoGPT** (https://github.com/karpathy/nanoGPT).

- Prepare data, then train on CPU:

```
python data/shakespeare_char/prepare.py
python train.py config/train_shakespeare_char.py \
  --device=cpu --compile=False --eval_iters=20 \
  --block_size=64 --batch_size=12 --n_layer=4  \
  --n_head=4 --n_embd=128 --max_iters=2000     \
  --lr_decay_iters=2000 --dropout=0.0
```

- Trains in a few minutes on a laptop CPU
- The model generates pseudo-Shakespearean text

Exercise 2: GPT from Scratch in a Single Notebook

Walk through a complete GPT implementation in one Jupyter notebook:
https://github.com/kevinpdev/gpt-from-scratch

- A single notebook (`llm-from-scratch.ipynb`) covers end to end:
  - **Tokenization** — converting text to token IDs
  - **Positional encoding** — telling the model about word order
  - **Self-attention** — the core mechanism of transformers
  - **Transformer decoder blocks** — multi-head attention + feedforward
  - **Training loop** — pretraining and supervised fine-tuning
  - **Inference** — generating new text from the trained model
- Runs on CPU — no GPU required

Putting It All Together

Comparison

	RAG	Fine-Tuning	Train SLM
Effort to set up	Low	Medium	High
Data requirements	Documents	Labeled examples	Large corpus
Model weights change?	No	Yes	Yes
Data freshness	Real-time	Retrain needed	Retrain needed
Privacy	Data sent to LLM	Data used in training	Fully private
Best for	Fact lookup	Style & format	Full control

These Approaches Are Complementary

- RAG + fine-tuning: a fine-tuned model that also retrieves current documents
- SLM + RAG: a private, domain-specific model augmented with a vector database
- The right choice depends on your data, budget, privacy needs, and use case
- Start simple (RAG), add complexity only when needed

"The best approach is usually the simplest one that meets your requirements."