Why, When and How to Fine-Tune a Custom Embedding Model

Victoria Slocum, Leonie Monigatti, August 5th, 2025

Just fine-tune your embeddings" they said.

"It'll fix your RAG system" they said.

They were wrong. Here's what actually works:

After working with countless retrieval systems, I've noticed a pattern: teams often jump straight to fine-tuning when their vector search underperforms. But that's like replacing your car engine when you might just need better tires.

𝗙𝗶𝗿𝘀𝘁, 𝗱𝗲𝗯𝘂𝗴 𝗯𝗲𝗳𝗼𝗿𝗲 𝘆𝗼𝘂 𝗳𝗶𝗻𝗲-𝘁𝘂𝗻𝗲:

Before spending time and compute on fine-tuning, ask yourself:

• Do many queries need exact keyword matches? → Try hybrid search first

• Are your chunks oddly split or lacking context? → Experiment with different chunking techniques like late chunking

• Is the model missing general semantic relationships? → Try a larger model or one with more dimensions

• Is it only failing on your specific domain terminology? → NOW we're talking fine-tuning territory

𝗪𝗵𝗲𝗻 𝗳𝗶𝗻𝗲-𝘁𝘂𝗻𝗶𝗻𝗴 𝗺𝗮𝗸𝗲𝘀 𝘀𝗲𝗻𝘀𝗲:

Fine-tuning shines when off-the-shelf models can't grasp your domain-specific language. Pre-trained models learn from Wikipedia and web crawls - they don't know your company's product names or industry jargon.

The payoff can be substantial:

• Better retrieval = better RAG performance

• Smaller fine-tuned models can outperform larger general ones

• Lower costs and latency for domain-specific tasks

𝗧𝗵𝗲 𝘁𝗲𝗰𝗵𝗻𝗶𝗰𝗮𝗹 𝗱𝗲𝗲𝗽-𝗱𝗶𝘃𝗲:

Fine-tuning embedding models isn't like fine-tuning LLMs. It's all about adjusting distances in vector space using contrastive learning.

Three main approaches:

1. 𝗠𝘂𝗹𝘁𝗶𝗽𝗹𝗲 𝗡𝗲𝗴𝗮𝘁𝗶𝘃𝗲𝘀 𝗥𝗮𝗻𝗸𝗶𝗻𝗴 𝗟𝗼𝘀𝘀: Just needs query-context pairs. Treats other examples in the batch as negatives - elegant and popular

2. 𝗧𝗿𝗶𝗽𝗹𝗲𝘁 𝗟𝗼𝘀𝘀: Requires (anchor, positive, negative) triplets. Great for precise control but finding good hard negatives is tricky

3. 𝗖𝗼𝘀𝗶𝗻𝗲 𝗘𝗺𝗯𝗲𝗱𝗱𝗶𝗻𝗴 𝗟𝗼𝘀𝘀: Uses similarity scores between sentence pairs. Perfect when you have gradients of similarity

𝗣𝗿𝗮𝗰𝘁𝗶𝗰𝗮𝗹 𝗰𝗼𝗻𝘀𝗶𝗱𝗲𝗿𝗮𝘁𝗶𝗼𝗻𝘀:

• Start with 1,000-5,000 high-quality samples for narrow domains

• Plan for 10,000+ for complex specialized terminology

• Good news: fine-tuning can run on consumer GPUs or free Google Colab for smaller models

• Always evaluate against a baseline - use metrics like MRR, Recall@k, or NDCG

𝗣𝗿𝗼 𝘁𝗶𝗽: The MTEB leaderboard is your friend for finding base models, but remember - leaderboard performance doesn't always translate to your specific use case.

The bottom line? Fine-tuning is powerful but it's not a magic bullet. Sometimes your retrieval problems need a different solution entirely. Debug systematically, and when you do fine-tune, start small and iterate.

Check out the full technical blog - it includes code examples for both Hugging Face and AWS SageMaker integrations: [1]

# References

[1] <https://weaviate.io/blog/fine-tune-embedding-model>, Leonie Monigatti, Aug 5th, 2025

[2] [Just fine-tune your embeddings, Victoria Slocum, X post, Aug 13, 2025](https://x.com/victorialslocum/status/1955595439241044274)