**Flow 1: Retrieval-Augmented Generation (RAG)**

This is the **retrieval + generation** pipeline. The GPT model doesn’t “memorize” your financial data — instead it **looks it up** at runtime.

**Steps in RAG Mode**

1. **Data Collection & Preprocessing**
   * Load financial reports (PDF, Excel, HTML → text).
   * Clean noise (headers, page numbers).
   * Split into **chunks** (100 and 400 tokens).
   * Assign IDs + metadata.
2. **Embedding & Indexing**
   * Chunks are converted into vectors with a sentence embedding model (all-MiniLM-L6-v2).
   * Store vectors in **FAISS** for dense retrieval.
   * Store text tokens in **BM25** for sparse keyword retrieval.
3. **Hybrid Retrieval**
   * User enters a query → e.g., *“What was the company’s revenue in 2023?”*
   * Preprocess query (lowercase, stopwords removed).
   * Create query embedding.
   * Retrieve top-N relevant chunks from:
     + **Dense retriever (FAISS)**
     + **Sparse retriever (BM25)**
   * Merge results (union or weighted scoring).
4. **Memory Bank Supplement**
   * Frequently asked Q/A pairs are checked.
   * If query matches → answer directly from memory.
5. **Concatenate Context + Query**
   * Combine retrieved chunks + user question → form a prompt.
6. **Generation**
   * Send prompt into GPT-2 (Small or Medium).
   * Model produces final answer.
7. **Guardrails**
   * **Input guard**: Block unsafe/irrelevant queries.
   * **Output guard**: Detect hallucinations, filter results.

➡️ **Use Case**: Best when you want the model to always pull from latest company reports without retraining.

**🔹 Flow 2: Fine-Tuning**

This is the **memorization** approach. GPT-2 is trained on your financial Q/A dataset so it learns patterns directly.

**Steps in Fine-Tuning Mode**

1. **Prepare Dataset**
   * Take ~50 handcrafted Q/A pairs from financial reports.
   * Convert to fine-tuning format:
   * Q: What was the revenue in 2023?
   * A: The company’s revenue in 2023 was $4.13 billion.
2. **Baseline Evaluation**
   * Ask GPT-2 (before fine-tuning) the 10+ test questions.
   * Measure:
     + Accuracy
     + Confidence
     + Response time
3. **Fine-Tuning**
   * Train GPT-2 Small (or Medium) on your Q/A dataset.
   * Use **Trainer API** with hyperparameters (learning rate, batch size, epochs).
   * Model learns to map questions → correct financial answers.
4. **Incremental Updates**
   * New reports arrive? Add new Q/A pairs.
   * Re-fine-tune on additional data without losing old knowledge.
5. **Chat Interaction**
   * User query → directly passed to fine-tuned GPT-2.
   * Model responds from learned weights (no retrieval step).
6. **Guardrails**
   * Same as RAG — input filtering + output checking.

➡️ **Use Case**: Best when the financial reports are small, stable (don’t change often), and you want **faster inference** without retrieval overhead.

**🔀 Switching Between Flows**

The CLI lets you pick:

* **RAG Mode** → python cli.py chat --mode rag --model gpt2-small
* **Fine-Tune Mode** → python cli.py chat --mode finetune --model gpt2-small

Both flows share:

* Guardrails
* GPT-2 backbone (Small/Medium)
* Evaluation harness

**⚖️ Quick Comparison**

| **Feature** | **RAG** | **Fine-Tuning** |
| --- | --- | --- |
| **Data freshness** | Always up-to-date (retrieves docs) | Static (requires retraining) |
| **Setup time** | Needs embeddings + index | Needs training |
| **Accuracy** | Context-aware, less memorized | Direct answers, but limited to trained Q/A |
| **Speed** | Slower (retrieval + generation) | Faster (just generation) |
| **Best for** | Large, changing financial data | Small, stable datasets |

👉 So in practice:

* Use **RAG** if your company reports change quarterly and you need **dynamic answers**.
* Use **Fine-Tuning** if you want a **lightweight, offline model** that answers fast.