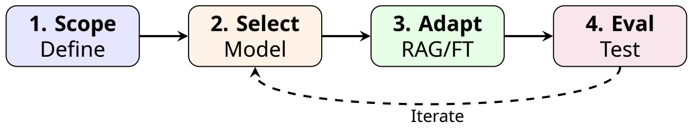
Generative AI Project Cheat Sheet

From Concept to Production: Strategy, RAG, Fine-Tuning & Ops

1 Project Lifecycle



Step 1: Scoping

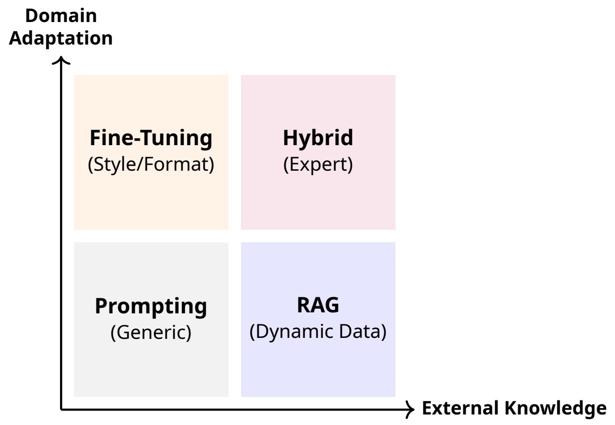
- **Task Definition**: Summarization, Q&A, Extraction, Code Gen?
- **Value**: Does it solve a real user pain point?
- **Risk**: hallucination tolerance, privacy requirements.

Step 2: Model Selection

- **Proprietary (API)**: GPT-4o, Claude 3.5. Fast start, pay-per-token.
- **Open Source (Weights)**: Llama 3, Mistral. Data privacy, control, hosting costs.
- **SLM (Small LM)**: Phi-3, Gemma. Efficient for edge/simple tasks.

2 Adaptation Strategy

How to customize the LLM for your data.



Prompt Engineering (In-Context) *First line of defense. Low cost/effort.*

- **Zero-Shot**: Direct instruction.
- **Few-Shot**: Provide examples (input → output).
- **Chain-of-Thought**: ”Think step-by-step”.

RAG (Retrieval-Augmented Gen) *Connects LLM to external, private, up-to-date data.*

- **Pros**: Reduces hallucinations, access to live data, traceable sources.
- **Cons**: Complexity of retrieval, context window limits.

Fine-Tuning (SFT) *Adapting the model’s behavior or style.*

- **Pros**: Specific format compliance, tone, smaller models perform better.
- **Cons**: Expensive, ”Catastrophic Forgetting”, static knowledge.

3 RAG Architecture

Ingestion Pipeline

1. **Load**: Extract text from PDF, HTML, DBs.
2. **Chunk**: Split text into smaller pieces (e.g., 512 tokens).
3. **Embed**: Convert chunks to vectors (OpenAI, HuggingFace).
4. **Store**: Save vectors in DB (Pinecone, Chroma, pgvector).

Retrieval

- **Semantic Search**: Cosine similarity (Dense).
- **Keyword Search**: BM25 (Sparse).
- **Hybrid Search**: Combine Dense + Sparse + Reranking.

Generation

```
context = retrieve(query)
prompt = f"""
Answer based on context: {context}
Question: {query}
"""
response = llm.generate(prompt)
```

4 Fine-Tuning (PEFT)

Why PEFT? (Parameter-Efficient) *Training all 7B+ params is too expensive. We train adapters instead.*

Techniques

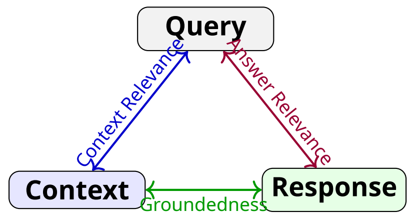
- **LoRA**: Low-Rank Adaptation. Injects small rank matrices. Fast & cheap.
- **QLoRA**: Quantized LoRA (4-bit). Run Llama-70B on 1 GPU.

Data Format (JSONL)

```
{"input": "...", "output": "..."}
{"input": "...", "output": "..."}
```

5 Evaluation

Don’t guess. Measure.

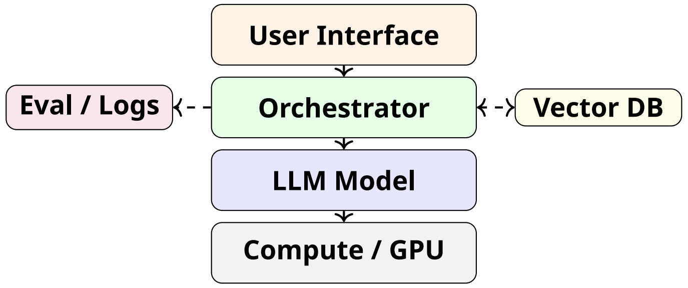


The RAG Triad (RAGAS Metrics)

- **Context Relevance**: Is the retrieved data useful?
- **Groundedness**: Is the answer supported by the context? (Hallucination check).
- **Answer Relevance**: Did it answer the user’s question?

LLM-as-a-Judge Using a stronger model (GPT-4) to grade the outputs of your system.

6 LLMOps & Deployment



Model Serving

- **vLLM**: High-throughput serving (PagedAttention).
- **TGI**: Hugging Face Text Generation Inference.
- **Ollama**: Local inference (great for dev).

Optimization

- **Quantization**: FP16 $\rightarrow$ INT8/INT4. Reduces VRAM, increases speed, slight quality loss.
- **Caching**: Cache common queries (Semantic Cache).

Guardrails Input/Output filtering for PII, toxicity, and jailbreaks (NeMo Guardrails, Guardrails AI).