Zuu Crew®

AI Engineer Essentials

Assignment: Mini Project 01

Operation Ledger-Mind: The Financial Intelligence

Course Module: Weeks 01-03 (Prompt Engineering, Fine-Tuning, Advanced RAG)
Scenario: Financial Analysis of Uber Technologies (2024 Annual Report)
Weight: 10% of Total Grade

The Mission

You have just been hired as the **Lead AI Architect** for *Alpha-Yield Capital*, a quantitative hedge fund. The firm is drowning in PDF annual reports and needs an automated way to extract insights.

The Board of Directors is debating two strategies:

1. **"The Intern" (Parametric Memory):** Train a small model to "read" the report and memorize the strategy/tone.
2. **"The Librarian" (Non-Parametric Memory):** Build a search engine to retrieve exact page citations.

Your mission is to build **BOTH** systems and conduct a ruthless "Showdown" to determine which architecture wins for specific financial tasks.

Technical Requirements

Part 1: The Data Factory [15 Marks]

The Problem: Financial PDFs are messy (tables, headers, footnotes). To train a model, we need clean, structured instruction data.

Your Task: Implement a robust data generation pipeline to transform the *2024-Annual-Report.pdf* into a fine-tuning dataset.

Required Workflow:

1. **Ingestion & Cleaning:**
    - Load the documents (and clean if necessary to remove headers/footers).
2. **Chunking Strategy:**
    - Split the documents into chunks of **1500 characters**.
3. **The Generation Loop:**
   For each chunk, you must generate 10 Q/A pairs using the following pipeline:
    - **Step A (Question Generation):** Use **LLM A** to generate 10 questions based strictly on the content of the current chunk.
    - **Step B (Answer Generation):** Feed the **context (chunk)** and the **generated questions** to **LLM B** to generate the final answers.
4. **Storage & Splitting:**
    - Store the generated data in JSON or CSV format.
    - **Split the Data:** Save 80% of the pairs to train.jsonl and 20% to golden_test_set.jsonl.

Constraints:

- **Categories:** Ensure your prompts cover **Hard Facts**, **Strategic Summaries**, and **Stylistic/Creative** outputs.

Part 2: "The Intern" (Fine-Tuning) [25 Marks]

The Problem: Generic models (like Llama-3-Base) don't know Uber's specific 2024 strategy or tone.

Your Task:

- Use the **Hugging Face Ecosystem** (*transformers, peft, trl, bitsandbytes*) to fine-tune a model.
- **Base Model:** Use Llama-3-8b (or a similar instruction-tuned variant).
- **Quantization:** Implement **4-bit quantization** using BitsAndBytesConfig (NF4, double quant) to fit on T4 GPUs.
- **Adapter Config:** Configure **LoRA (Low-Rank Adaptation)** using LoraConfig (Target modules: q_proj, k_proj, v_proj, o_proj).
- **Training:** Use the SFTTrainer from the *trl library*.

Constraint: Train for min 100 steps. Save the adapters and create an inference pipeline query_intern(question).

Part 3: "The Librarian" (Advanced RAG System) [25 Marks]

The Problem: Financial documents contain specific entities (e.g., "Form 10-K", "$37B") that semantic search often misses.

Your Task:

- Build an **Advanced Hybrid RAG Pipeline**.
- **Vector Database:** You **must use Weaviate** (Embedded or Cloud).
- **Hybrid Search:** Combine Dense Vector Search + BM25 (Keyword Search).
- **Refinement:** Implement **Reciprocal Rank Fusion (RRF)** and **Cross-Encoder Reranking**.

Constraint: Create query_librarian(question).

Part 4: The Showdown (Evaluation) [20 Marks]

The Problem: Which model is better? Cost vs. Accuracy.

Your Task: Run your **Golden Test Set** through BOTH models.

1. **Implement Important Metrics:**
    - **ROUGE-L:** Measure the textual overlap between the generated answer and the ground truth.
    - **LLM-as-a-Judge:** Use a stronger model (Reasoning Model) to score "Faithfulness" and "Accuracy" on a scale of 1-5.
    - **Latency:** Measure the time taken (in milliseconds) for each system to generate a response.

2. **Compare and Contrast:** Create a results table comparing the two approaches across all metrics.

3. **BONUS (Cost Comparison):**
    - Assume you have **500 daily users** making 10 queries each.
    - Estimate the monthly cloud cost for both strategies.
    - **Hint:** Look up the pricing for relevant AWS GPU instances (e.g., g4dn.xlarge or g5.xlarge) needed to serve your models at this scale.

Main Deliverables & Submission Checklist

You must submit a **single ZIP file** containing your full project folder (all scripts, notebooks, artifacts) and your Engineering Report. Below contains the key deliverables that expects with the inclusion of your full source code.

1. Data Generation (01_data_factory.ipynb)

- Code for PDF ingestion and chunking.
- The "Master Prompt" used to generate the dataset.
- Evidence of the Train/Test split.

Output Artifacts: You must include the generated train.jsonl and golden_test_set.jsonl files in your submission.

2. LLM Finetuning (02_finetuning_intern.ipynb)

- **Setup:** BitsAndBytesConfig and LoraConfig implementation.
- **Training:** SFTTrainer loop and loss curves.
- **Inference:** Function query_intern that loads the base model + trained adapters.

3. Advanced RAG System (03_rag_librarian.ipynb)

- **Vector DB:** Weaviate (Schema & Indexing).
- **Retrieval:** Implementation of Hybrid Search (Dense Vectors + BM25).
- **Refinement:** Implementation of Rank Fusion & Cross-Encoder Reranking.
- **Inference:** Function query_librarian.

4. Proper Evaluation (04_evaluation_arena.ipynb)

- Implementation of **ROUGE-L** and **LLM-as-a-Judge** scoring.
- Latency measurement code.
- **Bonus:** The Cost Analysis calculation and markdown summary.

5. Full Source Code

- Your submission must include **all Python scripts** and **utils** generated during the project, not just the notebooks. Ensure the folder structure is preserved in your zip file.

6. Engineering Report (PDF - 1500 Words) [15 Marks]

A formal technical report detailing your process and findings.

- **Structure:**
  - **Executive Summary:** Which architecture won? (150 words)
  - **Methodology:** Explain your Prompting strategy and Hybrid RAG parameters. (500 words)
  - **The "Hallucination" Audit:** specifically analyze where the Fine-Tuned model failed on numbers. Provide examples. (500 words)
  - **Conclusion:** When would you recommend Fine-Tuning vs RAG for a Fintech client? (350 words)

⚠️ Critical Warnings

- **Colab Resources:** Use the T4 GPU runtime for Notebooks 02.
- **Weaviate:** If using Weaviate Embedded in Colab causes issues, you may use Weaviate Cloud (WCD) free tier and provide the API keys in your submission notes.