Promptly - Model Development Report

---

1. Overview

This report documents the model development process for *Promptly*, an AI-powered document-based Q&A system. The model development focuses on loading preprocessed embeddings from the data pipeline, preparing for model training, fine-tuning large language models (LLMs), model validation, sensitivity analysis, bias detection, and pipeline automation for continuous improvement.

---

2. Model Development Process

2.1 Loading Data from the Data Pipeline

Our data pipeline automates the ingestion, processing, and storage of documents. This involves the following steps:

Ingestion: We use Airflow DAGs to trigger the document ingestion process.

PII Detection and Redaction: Sensitive information is identified and redacted using custom PII detection techniques.

Semantic Chunking and Embedding: Documents are semantically chunked using Markdown chunking methods, and embeddings are generated using the Nomic embed-text-v1.5 model.

Storage: Cleaned documents and their embeddings are stored in Supabase, which is used for both RAG and as a database.

Finetuning Data: To finetune our model for now we run a script to ingest dummy conversations and their relevant context chunks in Supabase. These conversation are then fetched to finetune our LLM model.

2.2 Model Training & Selection

We trained multiple LLM models like `Qwen/Qwen2.5-0.5B-Instruct, Llama 3.1 8B` but we chose `Qwen/Qwen2.5-0.5B-Instruct` to balance the accuracy and managing resource constraint.

- **Embeddings:** Generated with `nomic-embed-text-v1.5`.
- **Training:** Conducted using Leave-One-Out Cross-Validation due to limited dataset size.
- **Model Selection:** The best model was chosen based on accuracy and stability across validation runs.
- **Model Upload:** After finetuning the best model is uploaded to huggingface.

2.3 Model Validation

Validation metrics included:

- **ROUGE Metrics (ROUGE-1, ROUGE-2, ROUGE-L):** Used to evaluate the quality of model-generated responses by comparing them against ground truth outputs. This helps quantify improvements in content overlap and linguistic accuracy.
- **Baseline vs. Fine-Tuned Model Comparison:** We evaluate the fine-tuned model's performance relative to the pre-trained model to assess the impact of fine-tuning.
- **Quantitative and Qualitative Analysis:**
    - Quantitative: ROUGE scores for each test sample and their averages are recorded for performance benchmarking.
    - Qualitative: Sample outputs (query, ground truth, model responses) are collected for manual inspection of model behavior.
- **Model Checkpoint Evaluation:** We use the best-performing model checkpoint for validation to ensure optimal performance.
- **Reproducibility:** We ensure consistent evaluation through controlled decoding parameters and comprehensive logging with Weights & Biases (wandb) and MLflow.

2.4 Bias Detection

We implemented bias detection by measuring user dominance in the training data.

- **Method:** Aggregating chunk contributions by `upload_user_id` using Supabase join queries.
- **Threshold:** If any single user contributes more than 50% of document chunks, retraining is blocked.

- **Implementation:** Integrated into Airflow retraining DAG using a branch operator. Future plans include slicing-based bias detection by document category.

2.5 CI/CD Pipeline Integration

Our model development is automated using Airflow and containerized processes:

- **Data Pipeline DAGs:** Handle document ingestion, PII redaction, and chunk embedding.
- **Model Pipeline DAGs**: Automate fine-tuning, evaluation, and model registry updates.
- **GitHub Actions:** Trigger fine-tuning jobs, deploy docker images to the container registry, and monitor performance logs.

---

3. Hyperparameter Tuning

Hyperparameters tuned:

- **LoRA Parameters:** Tuned rank and alpha values for efficient training.
- **Batch Size:** Explored sizes from 4 to 32, selecting 16 for stability and speed.
- **Learning Rate:** Fine-tuned between 1e-5 and 5e-6 for optimal convergence.

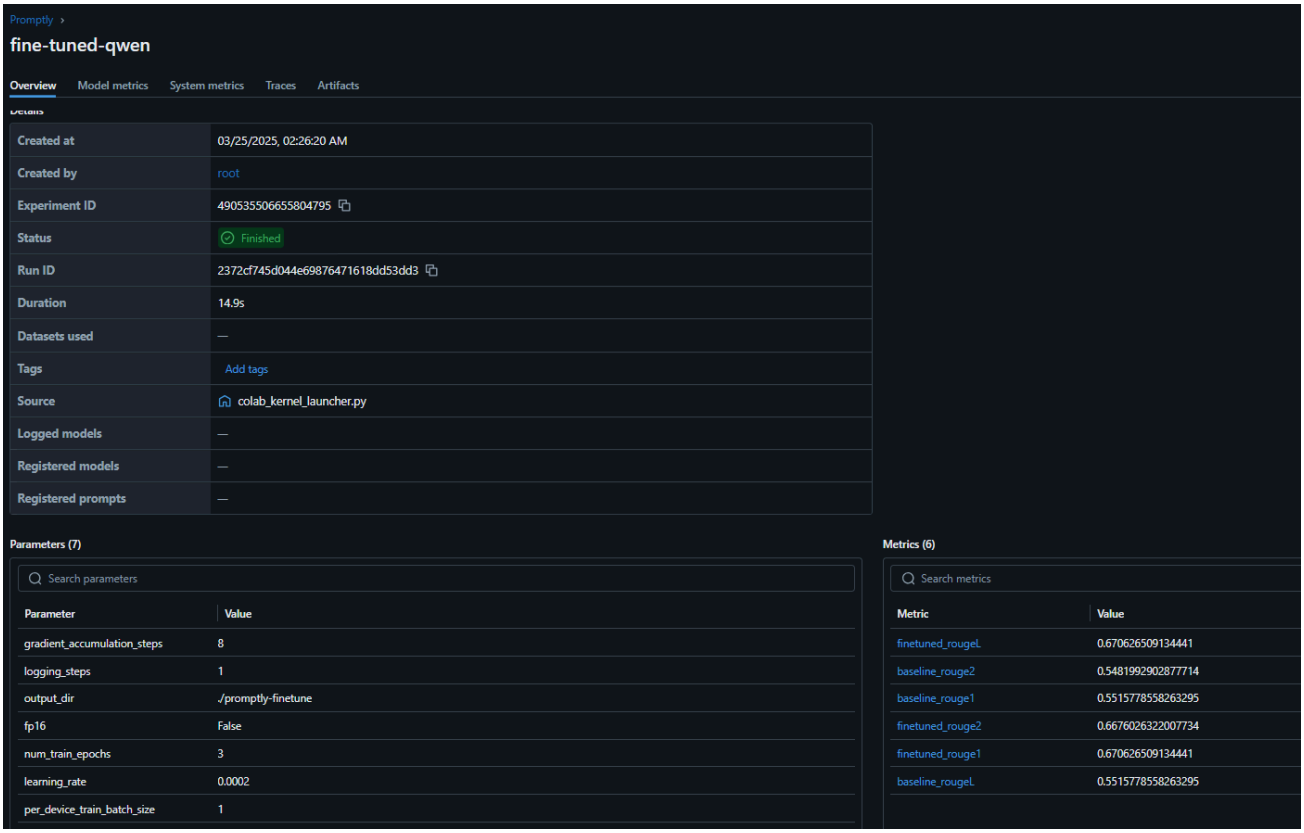
Hyperparameter decisions are logged and tracked for reproducibility.

---

4. Experiment Tracking & Results

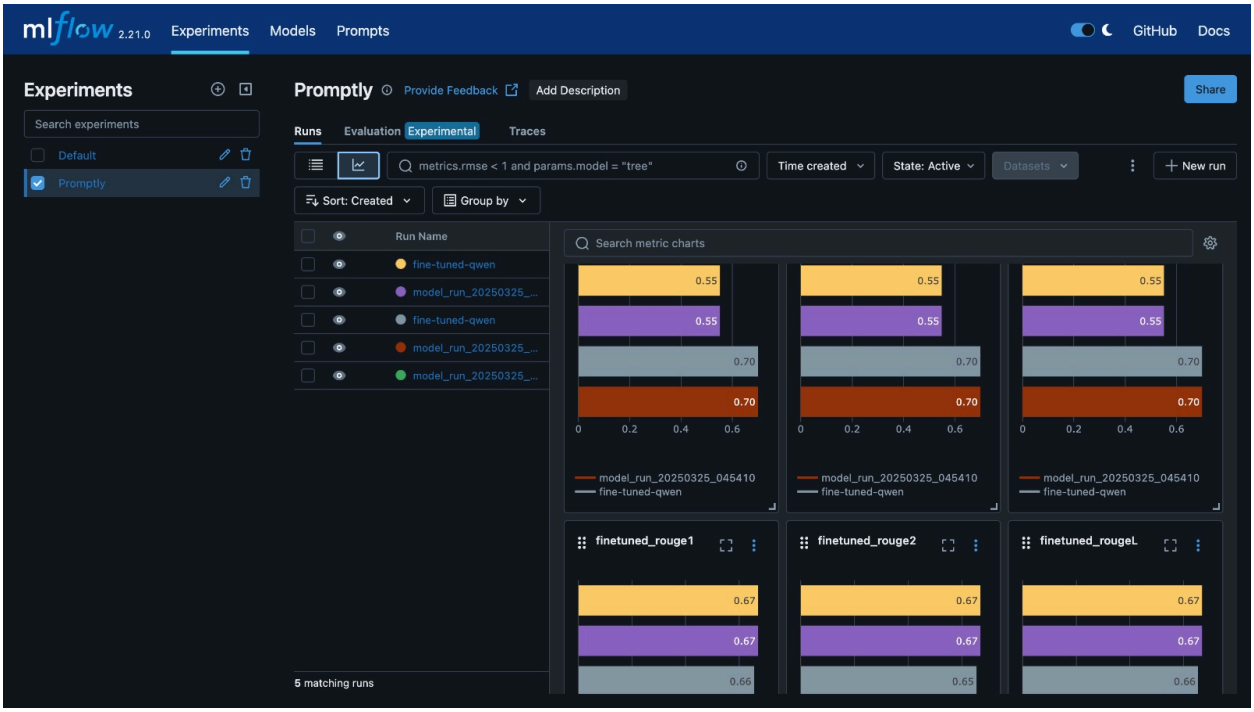
- Used MLflow for experiment tracking

- **MLFlow Tracking:**



5. Model Sensitivity Analysis

- **Feature Importance:** Not applicable (dense embedding vectors).
- **Hyperparameter Sensitivity:** Minor fluctuations were noted when tuning our LLM models' hyperparameter like learning rate, LORA and etc.



---

6. Model Bias Detection (Using Slicing Techniques)

- Current implementation: User dominance bias detection via join-based analysis of `document_chunks` and `documents` in Supabase.
- Future roadmap:
    - Slice performance analysis by document categories.
    - Visual bias reports via Fairlearn and TFMA once larger datasets are available.
    - Planned bias mitigation through re-sampling and balanced synthetic data generation.

---

7. CI/CD Pipeline Automation for Model Development

- **Airflow Orchestration:** DAG with integrated bias detection gating.
- **Trigger Setup:** Planned GitHub Actions integration.

- **Validation & Metrics Logging:** Automatic metrics computation post-training.
- **Automated Bias Detection:** Included in the DAG branch logic.
- **Model Registry Push:** Deploying model endpoint docker image to GCP Artifact Registry.
- **Rollback Mechanism:** If the new model underperforms, revert to the last stable model pickle.

---

8. Code Implementation Summary

- **Data Loading:** `load_data.py` integrated with Supabase and versioned data sources.
- **Training & Selection:** `train_model.py` handles training, validation, and saving of best models.
- **Bias Detection:** `bias_detection.py` fully integrated with Supabase and Airflow.
- **Inference Service:** FastAPI-based inference service (`inference_service.py`) using FAISS and fallback LLMs.
- **Front-end:** Streamlit app for end-user queries.
- **Containerization:** Dockerfile (WIP) for reproducibility and deployment.
- **Monitoring:** Planned W&B and Prometheus integration for long-term tracking.

---

9. Model Fine-Tuning and Synthetic Data Generation Pipeline

9.1 Fine-Tuning Qwen2.5-0.5B-Instruct

- **Base Model:** `Qwen/Qwen2.5-0.5B-Instruct`
- **Fine-Tuning Framework:** LoRA adaptation (`r=16`, `lora_alpha=32`)
- **Dataset:** 69 conversational examples from the Supabase conversations table.
- **Training Config:**
    - Batch size: 1 (gradient accumulation steps = 8)
    - Learning rate: 2e-4
    - Epochs: 3
    - Tracked on W&B
- **Deployment:** Hosted on Hugging Face Hub: [huggingface/promptly-tuned]

9.2 Synthetic Data Generation Pipeline

- **Workflow:**
    - Ingest and redact documents
    - Semantic chunking and embedding generation
    - Generate queries and responses using LLaMA 3.2 8B
    - Store conversations in Supabase for future fine-tuning
- **Technology Stack:**
    - Data Pipeline: Airflow
    - Storage: Supabase
    - Embeddings: Nomic
    - Synthetic Generation: LLaMA 3.2 8B

9.3 Fine-Tuning Results

Example ID	Baseline ROUGE-L	Fine-Tuned ROUGE-L
0	0.5244	0.5029
1	0.8397	0.8730
2	0.7629	0.7957
3	0.4828	0.7955
4	0.7143	0.8434
5	0.5656	0.8681
6	0.2342	0.3700
7	0.8163	0.9023
Average	**0.6175**	**0.7438**

9.4 Future Improvements

- Expand the dataset with real user queries
- Add multi-modal document support
- Improve conversational diversity in synthetic data

---

10. Conclusion

The Promptly project has successfully established a robust end-to-end model pipeline integrating data ingestion, embeddings, classification, fine-tuning, and inference. Bias detection, model selection, and planned CI/CD automation ensure sustainable model development. Future improvements will focus on scaling, enhancing fairness detection, and automating deployment with GCP services.