Task 3 – Dataset Preparation for Fine-Tuning

Submitted by: Utkarsh Srivastava

Project: Business QA Bot using OpenAI & Pinecone

Introduction

High-quality datasets are the cornerstone of successful fine-tuning for any AI language model. Preparing a dataset for fine-tuning a Retrieval-Augmented Generation (RAG) or LLM-based QA system requires strategic design, refinement, and validation steps to ensure it enhances the model's performance and domain relevance.

This document discusses essential techniques for dataset development and refinement, followed by a comparison of popular fine-tuning approaches with a recommended method for business-focused QA tasks.

Part 1: Techniques for Developing and Refining High-Quality Fine-Tuning Datasets

1.1 Data Collection & Curation

- **Source Identification**: Choose relevant and authoritative sources such as internal knowledge bases, business documentation, product manuals, support chats, and FAQs.
- **Diversity of Questions**: Include a variety of query types—factual, descriptive, procedural, and edge cases.
- **Balance**: Ensure representation across all topics and services offered by the business.

1.2 Data Formatting

- Use **JSONL format**: Each example should be a line-separated JSON object with prompt and completion fields.

{"prompt": "What are the subscription plans?", "completion": "Our service starts at $49/month with a 7-day free trial."}

- Maintain consistent structure across examples.
- Strip special characters, redundant tokens, and personal identifiable information (PII).

1.3 Data Augmentation

- Use **question rephrasing** techniques with models like GPT-3 to generate multiple versions of a query.
- Introduce **synthetic examples** based on unseen scenarios to improve generalization.
- Add **negative samples** to help the model learn to say "I don't know" when uncertain.

1.4 Quality Validation

- Perform **manual review** of examples for factual correctness and tone.
- Use GPT-4 to rate dataset examples for clarity and completeness.
- Run small-scale fine-tuning and evaluate model outputs for hallucinations or bias.

1.5 Dataset Size and Tokenization

- Ensure dataset is not too small (<500 examples may underperform) or too large (may overfit or become costly).
- Use tokenizers (like OpenAI's tiktoken) to check prompt-completion lengths and stay within limits (e.g., 2048 tokens for older models, 4096+ for GPT-3.5/4).

Part 2: Comparison of Fine-Tuning Approaches

2.1 Full Fine-Tuning

- **Description**: Updates all model parameters using a custom dataset.
- **Pros**: Tailors the model deeply to your domain.
- **Cons**: High compute cost; risk of catastrophic forgetting.

2.2 Instruction Fine-Tuning

- **Description**: Trains the model to follow domain-specific instructions or prompts.
- **Pros**: Effective for improving task-following behavior.
- **Cons**: Still computationally intensive.

2.3 Parameter-Efficient Fine-Tuning (PEFT)

- **Description**: Updates a small subset of model parameters (e.g., using LoRA, Prefix Tuning).
- **Pros**: Lower cost, faster training, supports frequent updates.
- **Cons**: May not achieve full performance on deeply technical domains.

2.4 Embedding Fine-Tuning (for Retrieval)

- **Description**: Fine-tunes only the embedding model used in the retriever, not the generator.
- **Pros**: Improves retrieval precision while keeping generation intact.
- **Cons**: Limited impact on answer quality without good generation.

2.5 Preference: Parameter-Efficient Fine-Tuning (PEFT)

For our use case (Business QA Bot), **PEFT**—specifically **LoRA-based fine-tuning**—is ideal due to:

- Low compute requirements
- Fast retraining on new business content
- Retains general capabilities of base model

- Easily updatable with new datasets

Conclusion

Effective fine-tuning of language models depends critically on well-structured and validated datasets. Techniques like thoughtful data collection, augmentation, and validation improve model performance and reduce hallucinations. Among available methods, **Parameter-Efficient Fine-Tuning (PEFT)** stands out as a practical, scalable, and cost-effective approach for fine-tuning domain-specific QA bots like ours.

References

- OpenAI Fine-Tuning Guide: https://platform.openai.com/docs/guides/fine-tuning
- Hugging Face LoRA: https://huggingface.co/blog/peft
- LangChain Dataset Tools: https://docs.langchain.com/docs/modules/data_connection
- Stanford Alpaca: https://github.com/tatsu-lab/stanford_alpaca

---