Task 3 – Dataset Preparation for Fine-Tuning

Objective: To ensure high-quality performance from an AI model, especially in domain-specific tasks, it is essential to prepare, curate, and refine datasets effectively for fine-tuning. This task outlines best practices in dataset development and compares fine-tuning techniques to guide optimal model custom

1. Techniques for High-Quality Dataset Preparation

a. Data Collection and Relevance Filtering

- Gather data from domain-relevant sources: FAQs, manuals, chat logs, support tickets.
- Use keyword filtering and domain classifiers to exclude irrelevant or noisy entries.

b. Text Normalization and Cleaning

- Remove duplicates, non-UTF8 characters, and formatting artifacts.
- Normalize punctuation, casing, and spacing to reduce noise.

c. Annotation and Labeling

- Use subject matter experts (SMEs) or trained annotators to label intents, entities, or answers.
- Maintain inter-annotator agreement and audit trails.

d. Data Structuring (JSONL Format)

- Follow the standard format expected by the fine-tuning API (e.g., OpenAI: {"prompt": "...", "completion": "..."}).
- Maintain consistent context windows and avoid ambiguous completions.

e. Quality Assurance and Review Loops

- Random sampling for manual inspection.
- Run baseline model predictions on the dataset to detect inconsistencies.

f. Dataset Augmentation

- Use paraphrasing models or back-translation to enrich training data diversity.
- Simulate edge cases and adversarial examples for robustness.

2. Comparison of Fine-Tuning Approaches

Approach	Description	Pros	Cons
Instruction Tuning	Train model to follow structured instructions.	Improves task adherence and clarity	Requires diverse prompt formulations
Supervised Fine-Tuning	Provide specific input-output pairs.	High control and precision	Risk of overfitting on small datasets
RLHF	Reinforcement Learning from Human Feedback.	Aligns with human preferences	Resource-intensive and complex to manage
LoRA / PEFT	Parameter-efficient tuning of small model layers.	Low compute cost, fast adaptation	Limited flexibility for deep changes
Adapters	Plugin-like tuning modules for specific tasks.	Modular, reusable tuning	Adds complexity to deployment

3. Preferred Method: LoRA (Low-Rank Adaptation)

Why LoRA?

- It allows tuning large language models using limited data and compute.
- Ideal for domain-specific adaptation without needing to retrain the full model.
- Integrates well with HuggingFace Transformers and LangChain pipelines.

Use Case Fit: For a business QA bot, LoRA enables incremental domain updates as new data emerges, ensuring continued relevance and adaptability

Conclusion: Effective dataset preparation is foundational to fine-tuning success. Techniques like cleaning, structured formatting, expert labeling, and QA review ensure high-quality inputs. Among various fine-tuning strategies, LoRA stands out for its efficiency, modularity, and practicality in real-world deployments