Task 3

YardStick

Dharmanshu Singh | July 05, 2025

Introduction

Effective fine-tuning requires meticulously prepared datasets and strategic methodology selection. This report details practical techniques for dataset development and objectively compares fine-tuning approaches, drawing from implementation experience in Tasks 1-2.

Part 1: Dataset Preparation Techniques

1. Strategic Data Sourcing

- Primary Sources: User interaction logs from production systems, domain-specific documents
- Synthetic Expansion: Controlled generation using seed data

```
from langchain_core.prompts import ChatPromptTemplate
synth_prompt = ChatPromptTemplate.from_template(
    "Generate policy-compliant variations of: {seed_question}"
)
```

2. Data Refinement Process

- Cleaning Protocol: Remove non-text elements, standardize terminology, resolve ambiguous references
- Annotation Framework: Tiered validation system

3. Quality Assurance Measures

Checkpoint	Method	Quality Metric
Context Consistency	Cosine similarity analysis	Threshold: >0.75
Completeness	Required field validation	100% coverage
Ambiguity Resolution	Triple-blind annotation	Kappa score: >0.80

4. Augmentation Strategy

- Controlled Paraphrasing: Preserves core semantics while altering structure
- Context-Bound Expansion: Augment within verified policy boundaries

Task 3

YardStick

Dharmanshu Singh | July 05, 2025

```
def generate_variants(question):
    # Preserves core semantics while altering structure
    return [
        f"Explain {question}",
        f"Details about {question}",
        f"{question} - please clarify"
    ]
```

Part 2: Fine-Tuning Methodology Comparison

Approach Analysis

Method	Computational Efficiency	Data Requirements	Stability	Implementation Complexity
Full Fine-Tuning	Low	Extensive	Moderate risk	High
LoRA	High	Moderate	High stability	Medium
Adapter Layers	Medium	Moderate	High	High
Prompt Tuning	Very High	Minimal	Very High	Low
QLoRA	Very High	Minimal	High	Medium

Recommended Approach: LoRA (Low-Rank Adaptation)

Rationale for Selection:
1. Operational Efficiency: Modifies only 0.1-0.5% of parameters with 3× faster convergence
2. Knowledge Preservation: Maintains base model capabilities while adapting to domain specifics
3. Resource Optimization: Suitable for iterative development cycles

```
# LoRA configuration for business policy adaptation
peft_config = LoraConfig(
    r=8,
    lora_alpha=32,
    target_modules=["q_proj", "v_proj"],
    lora_dropout=0.05,
    bias="none"
)
```

Task 3

YardStick

Dharmanshu Singh | July 05, 2025

Conclusion

Dataset Preparation Insights:
- Quality emerges from structured refinement, not volume
- Context-bound augmentation ensures semantic integrity
- Tiered validation prevents error propagation

Fine-Tuning Recommendation:
LoRA provides optimal balance between adaptability and stability for policy-based QA systems.

Implementation Roadmap:
1. Curate 1,500 verified QA pairs from Task 1-2 systems
2. Establish continuous data quality monitoring
3. Execute phased LoRA deployment (departmental policies first)