

## Lab 102: Fine-tuning vs RAG vs Prompting + LoRA em Modelo Pequeno

Level: L300 Jornada: Ilm-foundations Tempo: ~120 min Custo: free-conditional

info "Phase 3 · local first" O caminho obrigatório é determinístico e local. O caminho Real é opcional e inclui estimativa e limpeza.

O que você aprenderá

Selecionar prompting, RAG ou fine-tuning por evidência Preparar dataset LoRA pequeno

Estimar custo GPU e Foundry antes do treino Comparar qualidade, atualização, latência e operação

Pré-requisitos

Python 3.10+ e um terminal Nenhuma credencial para o caminho obrigatório Leia a especificação oficial vinculada antes de usar um adaptador hospedado

Modelo visual

[diagram or code sample]

Cenário

A OutdoorGear precisa de uma implementação revisável para workshop, com fixtures determinísticas, fronteiras de confiança explícitas, critérios mensuráveis e adaptador hospedado opcional e reversível.

Exercício guiado

Step 1: score\_options

1. Execute a fixture determinística desta etapa. 2. Registre entrada, saída, decisão de política e comportamento de falha. 3. Altere um comportamento limitado e execute o teste local novamente.

Evidence: Salve comando, resultado e uma frase explicando por que a evidência é confiável.

Step 2: prepare\_dataset

1. Execute a fixture determinística desta etapa. 2. Registre entrada, saída, decisão de política e comportamento de falha. 3. Altere um comportamento limitado e execute o teste local novamente.

Evidence: Salve comando, resultado e uma frase explicando por que a evidência é confiável.

Step 3: simulate\_lora

1. Execute a fixture determinística desta etapa. 2. Registre entrada, saída, decisão de política e comportamento de falha. 3. Altere um comportamento limitado e execute o teste local novamente.

Evidence: Salve comando, resultado e uma frase explicando por que a evidência é confiável.

Step 4: compare\_results

1. Execute a fixture determinística desta etapa. 2. Registre entrada, saída, decisão de política e comportamento de falha. 3. Altere um comportamento limitado e execute o teste local novamente.

Evidence: Salve comando, resultado e uma frase explicando por que a evidência é confiável.

Execute localmente

[diagram or code sample]

Simulação / Caminho Real

Simulação Caminho Real

Use as fixtures determinísticas e o validador local incluídos. Custo estimado: US\$0.

Substitua somente a fronteira do adaptador. Defina teto de gastos, preserve os testes e registre região, retenção, identidade, telemetria e preço. Exclua implantações, endpoints, índices, traces e dados de teste após a validação.

Kit de apoio

O build do site gera downloads estáveis em PDF e ZIP. A CI também mantém um espelho

temporário por 30 dias.

README.md: início rápido e checklist de evidências CHEATSHEET.md: guia de decisão e revisão

scenario.json: contrato determinístico test\_scenario.py: testes locais Baixar PDF gerado

Baixar ZIP gerado

Quiz pontuado

question "Por que manter um contrato local?"

??? success "Resposta" Ele cria uma linha de base determinística antes de adicionar variabilidade e custo hospedados.

question "Quando o caminho Real termina?"

??? success "Resposta" Somente após registrar evidência de validação e limpeza.

question "Como tratar a saída de ferramentas?"

??? success "Resposta" Como dado não confiável até a validação de política e esquema.

Crêterios de saída

[ ] Testes locais passam sem credenciais. [ ] O diagrama tem explicação textual e não depende de cor. [ ] O quiz pontuado foi revisado. [ ] O caminho Real tem teto de custo e plano de limpeza. [ ] Fontes e estabilidade foram verificadas.

Fontes públicas oficiais

Hugging Face PEFT Microsoft Foundry fine-tuning

Supporting file: CHEATSHEET.md

Fine-tuning vs RAG vs Prompting + LoRA em Modelo Pequeno ? Cheat Sheet

Comece pelo contrato local e critérios explícitos. Trate prompts, conteúdo recuperado, ferramentas e mensagens como não confiáveis. Mantenha identidade, autorização, política, telemetria, custo e limpeza explícitos. Prefira a menor orquestração ou adaptação que satisfaça a evidência.

Supporting file: README.md

Fine-tuning vs RAG vs Prompting + LoRA em Modelo Pequeno

python -m unittest discover -v

O caminho obrigatório é local e determinístico.

Supporting file: adaptation\_dataset.jsonl

```
{"input": "OutdoorGear request", "expected": "grounded response", "label": "pass"}
```

Supporting file: decision\_matrix.md

Prompting vs RAG vs fine-tuning

Need | Prompting | RAG | Fine-tuning | Fresh private facts | Limited | Best fit | Poor fit

| Stable behavior/style | Good | Good | Best fit | Lowest operational cost | Best fit |

Medium | Highest setup cost |

Supporting file: scenario.json

```
{ "lab": "102", "steps": [ "score_options", "prepare_dataset", "simulate_lora",  
  "compare_results" ], "required_route": "local", "credentials_required": false, "real_path":  
  { "optional": true, "cost_cap_required": true, "cleanup_required": true } }
```

Supporting file: test\_scenario.py

```
import json import unittest from pathlib import Path
```

```
class Phase3ScenarioTests(unittest.TestCase): @classmethod def setUpClass(cls): cls.data =
```

```
json.loads(Path("scenario.json").read_text(encoding="utf-8"))
def test_identity_and_steps(self): self.assertEqual(self.data["lab"], "102")
self.assertEqual(len(self.data["steps"]), 4)
def test_free_first_contract(self): self.assertEqual(self.data["required_route"], "local")
self.assertFalse(self.data["credentials_required"])
def test_real_path_guardrails(self): real = self.data["real_path"]
self.assertTrue(real["optional"]) self.assertTrue(real["cost_cap_required"])
self.assertTrue(real["cleanup_required"])
if __name__ == "__main__": unittest.main()
```