Universidade Federal do Amazonas - Instituto de Computação
ICC220 - Tópicos Especiais em Bancos de Dados (para CC e ES) - 2025/01
PPGINF528 - Tópicos Especiais em Recuperação de Informação - 2025/01
Professores: André Carvalho e Altigran da Silva

Quarto Trabalho Prático
Entrega: 23/06/2025 até 23:59
Execução: Individual para alunos de pós-graduação Em trios para alunos de graduação

Análise Quantitativa do Trade-off entre Especialização e Generalização em LLMs via Fine-Tuning

1. Objetivo

O objetivo central deste projeto é a avaliação empírica e sistemática do processo de *fine-tuning* em Modelos de Linguagem de Grande Porte (LLMs). Os alunos irão implementar, treinar e avaliar um LLM para a tarefa de Text-to-SQL. A análise quantificará o ganho de desempenho na tarefa-alvo e, simultaneamente, medirá a degradação de performance em tarefas de conhecimento geral. O projeto exige a implementação de métricas de avaliação customizadas e uma análise crítica dos trade-offs inerentes à especialização de modelos.

2. Problema

A especialização de LLMs via *fine-tuning* é uma técnica proposta para otimizar o desempenho em domínios específicos. Contudo, este processo de otimização focado pode comprometer a robustez do modelo em tarefas que não pertencem ao domínio de treinamento, um fenômeno conhecido como "esquecimento catastrófico" ou "regressão de capacidade". Esse trabalho consiste em projetar e executar um pipeline experimental que permita medir com precisão ambas as facetas deste fenômeno: o ganho de especialização e a perda de generalização.

3. Materiais e Configuração

- **Modelo Base:** Deve ser utilizado um modelo open-source da classe de 7-8 bilhões de parâmetros em sua versão *instruct/chat*. Modelos sugeridos: `meta-llama/Llama-3-8B-Instruct`, `mistralai/Mistral-7B-Instruct-v0.2`. A versão exata do checkpoint utilizado deve ser documentada para garantir a reprodutibilidade.

- **Dataset de Fine-Tuning:** Spider Dataset, disponível no site oficial. Utilizar exclusivamente o *training split*. Os dados devem ser pré-processados para o formato exigido pelo framework de treinamento escolhido (e.g., formato de chat com `[INST]`, `[SYS]`).
- **Dataset de Avaliação de Tarefa:** Spider *development split*. Este conjunto de dados não deve ser visto pelo modelo durante o treinamento.
- **Dataset de Avaliação de Generalização:** MMLU (Massive Multitask Language Understanding), disponível no Hugging Face Hub. Deve ser criada uma suíte de avaliação composta por **exatamente 150 questões**, divididas igualmente em 3 categorias:
    1. **STEM:** 50 questões de uma subcategoria (e.g., `computer_science`).
    2. **Humanidades:** 50 questões de uma subcategoria (e.g., `philosophy`).
    3. **Ciências Sociais:** 50 questões de uma subcategoria (e.g., `economics`).
- **Framework de Avaliação:** `DeepEval` (versão `0.21.x` ou superior).
- **Frameworks de Treinamento:** `Hugging Face TRL`, `Axolotl` ou similar.

4. Procedimento Experimental Detalhado

Fase 1: Estabelecimento do Baseline de Desempenho

1.1. **Prompt Engineering:** Construir um prompt de *few-shot* para a tarefa Text-to-SQL. O prompt deve conter 3 exemplos representativos (par `linguagem natural` -> `consulta SQL`) extraídos do *training split* do Spider. Este template de prompt deve ser fixo e utilizado em todas as avaliações de baseline.

1.2. **Execução da Avaliação:** Submeter o modelo base (não treinado) ao `Spider dev split`, utilizando o template de prompt definido.

1.3. **Coleta de Dados:** Registrar a consulta SQL gerada para cada entrada. A métrica a ser reportada nesta fase é a contagem bruta de sucesso/falha baseada em execução manual ou em um script preliminar.

Fase 2: Execução do Fine-Tuning

2.1. **PEFT (Parameter-Efficient Fine-Tuning):** Implementar o fine-tuning utilizando a técnica LoRA (Low-Rank Adaptation).

2.2. **Configuração e Documentação:** A configuração do LoRA deve ser explicitamente documentada, incluindo: o rank (`r`), `alpha`, `dropout` e os módulos alvo (`target_modules`, e.g., `q_proj`, `v_proj`).

2.3. **Experimentação de Hiperparâmetros:** É **obrigatório** testar no mínimo duas configurações distintas de hiperparâmetros de treinamento. Sugestão: varie a taxa de aprendizado (`learning_rate`) ou o número de épocas (`num_train_epochs`). O objetivo é observar o impacto dessas variações no resultado final.

Fase 3: Avaliação de Desempenho na Tarefa-Alvo com Métrica Customizada

3.1. **Implementação da Métrica:**Desenvolver uma métrica customizada em `DeepEval` para "Execution Accuracy". A classe da métrica deve seguir a seguinte estrutura:

- Herdar de `deepeval.metrics.BaseMetric`.
- Implementar o método `measure(self, test_case: LLMTestCase) -> float`.
- A lógica interna do `measure` deve:
  - a. Estabelecer uma conexão com um banco de dados `sqlite` contendo a base de dados de teste do Spider.
  - b. Executar a consulta SQL contida em `test_case.actual_output`dentro de uma transação segura (para lidar com erros de sintaxe via `try-except`).
  - c. Executar a consulta *ground truth*contida em `test_case.expected_output`.
  - d. Comparar os conjuntos de resultados. A comparação deve ser insensível à ordem das linhas. e. Retornar `1.0` para sucesso (resultados idênticos) e `0.0` para falha.

3.2. **Avaliação Automatizada:**Integrar a métrica customizada em um teste `pytest` e avaliar o(s) modelo(s) fine-tuned no `Spider dev split`.

Fase 4: Análise Quantitativa de Regressão de Capacidade

4.1. **Metodologia de Avaliação MMLU:** Avaliar o modelo base e o(s) modelo(s) fine-tuned na suíte de 150 questões do MMLU. A avaliação deve ser feita em modo *4-shot*(conforme o benchmark padrão) para cada uma das quatro opções de múltipla escolha.

4.2. **Cálculo de Acurácia:** A métrica será a acurácia de resposta correta.

4.3. **Análise de Regressão:** Calcular a variação percentual de acurácia entre o modelo base e o modelo fine-tuned. Esta análise deve ser reportada de forma agregada e também por categoria (STEM, Humanidades, Ciências Sociais) para identificar se a regressão de capacidade afeta domínios de forma heterogênea.

5. Requisitos para os Entregáveis

1. **Repositório de Código (GitHub):** O repositório deve ter uma estrutura profissional e organizada:
   - `/scripts`: Contém os scripts de treinamento, avaliação e pré-processamento.
   - `/custom_metrics`: Contém o código da métrica de `ExecutionAccuracy`.
   - `requirements.txt`: Arquivo listando todas as dependências com suas versões fixadas.
   - `README.md`: Documentação detalhada sobre como configurar o ambiente e reproduzir todos os resultados apresentados no relatório.
2. **Relatório Técnico (PDF, máx. 10 páginas, formato IEEE/ACM):**

- **Metodologia:** Esta seção deve conter uma descrição pormenorizada do pipeline de dados, da configuração exata do LoRA (tabela de hiperparâmetros), e da arquitetura de software da métrica `ExecutionAccuracy`.
- **Resultados:** Os resultados devem ser apresentados com clareza estatística (e.g., médias, desvio padrão). É fortemente recomendada a inclusão de uma breve **análise de erros**, examinando 2-3 exemplos onde o modelo fine-tuned falhou na tarefa-alvo.
- **Discussão:** A análise deve ser aprofundada, respondendo a perguntas como: A magnitude do ganho na tarefa de Text-to-SQL justifica a perda de capacidade geral? Quais fatores (hiperparâmetros, arquitetura do modelo) parecem influenciar mais este trade-off? Quais são as implicações práticas destes achados para o desenvolvimento de LLMs comerciais especializados?

6. Sobre Reprodutibilidade

Toda a análise experimental deve ser computacionalmente reprodutível. É obrigatório fixar as sementes (`seeds`) para todas as operações estocásticas (e.g., inicialização de pesos, amostragem de dados, divisões de treino/teste) para garantir que os resultados possam ser verificados de forma independente.