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

Allan Carvalho de Aguiar
Universidade Federal do Amazonas
Manaus, BR
allan.aguiar@icomp.ufam.edu.br

Beatriz Emily Silva Aguiar
Universidade Federal do Amazonas
Manaus, BR
beatriz.aguiar@icomp.ufam.edu.br

Juíle Yoshie Sarkis Hanada
Universidade Federal do Amazonas
Manaus, BR
juile.hanada@icomp.ufam.edu.br

Keywords

Fine-tuning, LLMs, Text-to-SQL

ACM Reference Format:
Allan Carvalho de Aguiar, Beatriz Emily Silva Aguiar, and Juíle Yoshie Sarkis Hanada. 2025. Análise Quantitativa do Trade-off entre Especialização e Generalização em LLMs via Fine-Tuning. In *Proceedings of ICC220 - Tópicos Especiais em Bancos de Dados - NLP (IComp).* ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introdução

Modelos de Linguagem de Grande Porte (LLMs, do inglês *Large Language Models*) tornaram-se elementos centrais em aplicações de Processamento de Linguagem Natural, devido à sua impressionante capacidade de generalização após o pré-treinamento em grandes volumes de dados [Brown et al. 2020]. No entanto, em tarefas específicas, como a tradução de linguagem natural para SQL (*Text-to-SQL*), é comum aplicar técnicas de *fine-tuning* para adaptar o modelo ao domínio-alvo e melhorar seu desempenho [Raffel et al. 2020].

Embora o *fine-tuning* permita ganhos significativos em tarefas especializadas, ele pode comprometer a performance do modelo em outras tarefas previamente dominadas. Esse fenômeno, conhecido como esquecimento tarefas específicas, como a tradução de linguagem natural catastrófico, evidencia o trade-off entre especialização e generalização. Compreender esse equilíbrio é fundamental para o uso eficiente e seguro de LLMs em contextos diversos.

Este projeto propõe uma investigação empírica desse trade-off. Os alunos irão implementar um pipeline de *fine-tuning* para a tarefa de Text-to-SQL e, com base em métricas customizadas, avaliar tanto o ganho de desempenho na tarefa-alvo quanto a possível regressão em tarefas de conhecimento geral. A análise crítica desses resultados contribuirá para uma melhor compreensão das implicações práticas da especialização de modelos.

2 Metodologia

2.1 Modelo Base

Neste projeto, foi utilizado o modelo meta-llama/Llama-3.2-3B-Instruct, um modelo de linguagem open-source da classe de 3 bilhões de parâmetros, ajustado para tarefas no estilo *instruct*. Apesar de o projeto sugerir modelos de 7–8 bilhões de parâmetros, optou-se por este modelo mais leve devido a restrições computacionais, mantendo, entretanto, a aderência ao formato de interação baseado em instruções.

O carregamento do modelo foi feito utilizando quantização em 4 bits com o uso da biblioteca `bitsandbytes`. O modelo foi carregado com o `commit hash` `0cb88a4f764b7a12671c53f0838cd831a0843b95`, garantindo reprodutibilidade. A tokenização foi realizada com o `AutoTokenizer`, e o modelo foi carregado com `AutoModelForCausalLM`, ambos da biblioteca `transformers`.

2.2 Dataset de Fine-Tuning

O conjunto de dados utilizado para o *fine-tuning* foi o `Spider Dataset`, especificamente o *training split*, conforme disponibilizado no site oficial. A formatação dos exemplos incluiu a geração de prompts no estilo LLaMA 3, com marcadores `<|begin_of_text|>`, `<|start_header_id|>`, `<|end_header_id|>` e `<|eot_id|>`. O esquema do banco de dados para cada exemplo foi construído dinamicamente por meio de uma função customizada `formatar_schema`, que gera comandos `CREATE TABLE` a partir de um dicionário de metadados `schemas_map`.

2.3 Dataset de Avaliação da Tarefa

A avaliação do desempenho do modelo na tarefa-alvo de tradução de linguagem natural para SQL foi realizada com o *development split* do Spider Dataset. Esse conjunto de dados foi mantido fora do processo de treinamento, servindo exclusivamente como benchmark de desempenho na tarefa de especialização.

2.4 Dataset de Avaliação de Generalização

Para avaliar a capacidade de generalização do modelo após o *fine-tuning*, foi utilizada uma suíte de avaliação composta por 150 questões do benchmark MMLU (Massive Multitask Language Understanding), acessível via Hugging Face Hub. A construção da suíte foi realizada com base nas seguintes etapas:

- Seleção de três subcategorias: `college_computer_science` (STEM), `philosophy` (Humanidades) e `econometrics` (Ciências Sociais);
- Para cada subcategoria, foram amostradas aleatoriamente 50 questões a partir do split `test`, com `seed = 42` para garantir reprodutibilidade;
- Os subconjuntos foram combinados e embaralhados para formar o dataset final, estruturado como um `DatasetDict` contendo a chave `"evaluation"`.

IComp, June 23, 2025, Manaus, AM

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/XXXXXXX.XXXXXXX

Essa abordagem garante uma cobertura balanceada de domínios e uma amostra representativa para a análise de regressão de capacidade.

2.5 Procedimento Experimental

2.5.1 Fase 1: Estabelecimento do Baseline de Desempenho.

(1) **Construção dos prompts**: Para cada exemplo, o esquema do banco de dados correspondente foi formatado em uma string contendo declarações `CREATE TABLE`, utilizando uma função que constrói esse esquema a partir dos metadados originais do Spider Dataset. Esse esquema formatado é incorporado ao prompt junto com a pergunta.
(2) **Execução da Avaliação**: O modelo base (sem fine-tuning) foi avaliado em 50 exemplos do *dev split* do Spider Dataset, utilizando os prompts que incluem o esquema formatado e a pergunta correspondente.
(3) **Análise dos Resultados**: Para cada exemplo, foi registrada a pergunta, a query correta, a query gerada. Esses resultados foram salvos em um arquivo CSV para análise. Para medir a exatidão funcional das queries geradas, foi implementada uma verificação automática baseada na execução SQL. A métrica final considera como sucesso os casos em que os resultados coincidem.
(4) **Consolidação dos Resultados**: O arquivo CSV contendo as queries geradas foi carregado e submetido à verificação de execução descrita anteriormente. Para cada exemplo, foi registrada a classificação como *Sucesso* ou *Falha*, com base na comparação dos resultados da query gerada e da query correta. Essa informação foi utilizada para o cálculo da taxa bruta de acertos (*execution accuracy*).

2.5.2 Fase 2: Execução do Fine-Tuning.

(1) **PEFT (Parameter-Efficient Fine-Tuning)**: O modelo foi ajustado com a técnica **LoRA** (Low-Rank Adaptation).
(2) **Configuração Documentada**: A configuração usada foi:
   - Rank (r): 16
   - Alpha: 32
   - Dropout: 0,05
   - Módulos-alvo: `q_proj`, `k_proj`, `v_proj`, `o_proj`
(3) **Experimentação de Hiperparâmetros**: Foram testadas ao menos duas configurações distintas de hiperparâmetros (ex.: variação da taxa de aprendizado e número de épocas) com o objetivo de observar os impactos no desempenho do modelo.

Table 1: Configurações utilizadas nos experimentos de fine-tuning com LoRA

Parâmetro	V1	V2
num_train_epochs	1	2
per_device_train_batch_size	2	2
gradient_accumulation_steps	2	2
learning_rate	2×10^{-4}	2×10^{-4}
optim	paged_adamw_32bit	paged_adamw_32bit
logging_steps	25	25
save_strategy	epoch	epoch
fp16	True	True
push_to_hub	False	False
report_to	"none"	"none"

2.5.3 Fase 3: Avaliação na Tarefa-Alvo com Métrica Customizada.

(1) **Implementação da Métrica**: Foi criada uma métrica chamada *Execution Accuracy* estendendo a classe `BaseMetric` da biblioteca DeepEval. A função `measure` estabelece conexão com um banco SQLite contendo a base do Spider, executa a consulta gerada e compara os resultados, retornando 1.0 para igualdade e 0.0 para diferença.
(2) **Avaliação Automatizada**: A métrica foi incorporada a um teste `pytest` e aplicada ao modelo ajustado sobre o *dev split* do Spider Dataset.

2.5.4 Fase 4: Análise Quantitativa de Regressão de Capacidade.

(1) **Metodologia MMLU**: Os modelos base e ajustados foram avaliados com a suíte de 150 questões do MMLU em modo 4-shot. A suíte foi construída manualmente com o carregamento e embaralhamento controlado dos exemplos.
(2) **Cálculo de Acurácia**: A métrica considerada foi a acurácia de múltipla escolha.
(3) **Análise de Regressão**: A regressão de capacidade foi medida como a variação percentual da acurácia entre os modelos, agregada por domínio (STEM, Humanidades e Ciências Sociais).

3 Resultados

3.1 Resumo

Categoria	Acurácia Base (%)	Acurácia Fine-Tuned (%)	Variação Percentual (%)
GERAL	39.73	37.67	-5.17
STEM (Computer Science)	36.73	38.78	+5.56
Social Sciences (Econometrics)	32.65	30.61	-6.25
Humanities (Philosophy)	50.00	43.75	-12.50

Table 2: Comparação de desempenho por categoria no benchmark MMLU

O modelo ganhou desempenho na categoria **STEM (Computer Science)**, o que é consistente com o objetivo do fine-tuning voltado para Text-to-SQL. Foi observada uma regressão de capacidade nas áreas de **Humanities** e **Social Sciences**, o que pode indicar um caso típico de esquecimento catastrófico. A acurácia geral apresentou uma pequena queda de aproximadamente 5,17%, considerada um trade-off aceitável frente ao ganho substancial na tarefa-alvo.

Métrica	Baseline (%)	Fine-tuned (%)	Desvio Padrão	Variância
Execution Accuracy (SQL)	52.00	76.00	0.5047	0.2547
Acurácia MMLU	39.73	37.67	0.4910	0.2411

Table 3: Resumo estatístico das avaliações SQL e MMLU

O fine-tuning resultou em um ganho expressivo na Execution Accuracy (SQL), aumentando de 52.00% para 76.00%. Isso demonstra um avanço notável na especialização do modelo na tarefa de Text-to-SQL.

Observou-se uma leve regressão de capacidade geral na avaliação MMLU, com redução de 39.73% para 37.67% na acurácia geral. Apesar dessa queda, a magnitude da regressão pode ser considerada pequena e aceitável frente ao ganho substancial na tarefa-alvo.

A diminuição do desvio padrão e variância nas avaliações SQL sugere que o modelo fine-tuned apresentou respostas mais consistentes. Já no MMLU, a variabilidade das respostas permaneceu praticamente inalterada.

3.2 Análise de Erros - SQL

Exemplo 1
Pergunta: Find the number of concerts happened in the stadium with the highest capacity.
Query Correta: `SELECT count(*) FROM concert WHERE stadium_id = (SELECT stadium_id FROM stadium ORDER BY capacity DESC LIMIT 1)`
Query Gerada: `SELECT count(*) FROM stadium WHERE Capacity = (SELECT max(Capacity) FROM stadium)`
Problema: O modelo selecionou a tabela errada e não utilizou o contexto correto.

Exemplo 2
Pergunta: What are the names and locations of the stadiums that had concerts that occurred in both 2014 and 2015?
Query Correta: Consulta com `INTERSECT` correta entre os anos.
Query Gerada: A consulta montou JOINs redundantes e não completou corretamente a segunda parte da interseção.
Problema: Erro na construção de consultas com múltiplos passos e interseção de resultados.

Exemplo 3
Pergunta: Show the name and the release year of the song by the youngest singer.
Query Correta: `SELECT song_name , song_release_year FROM singer ORDER BY age LIMIT 1`
Query Gerada: `SELECT Name, Song_release_year FROM singer ORDER BY Age ASC LIMIT 1`
Problema: O modelo trocou o nome da coluna `song_name` por `Name`, causando falha de execução.

3.3 Análise de Erros - MMLU

Exemplo 1
Pergunta: Epicurus claims that all other virtues spring from:
Resposta Correta: 0
Resposta Gerada: 1
Categoria: Philosophy

Exemplo 2
Pergunta: Which of the following statements is TRUE concerning the standard regression model?
Resposta Correta: 0
Resposta Gerada: 3
Categoria: Econometrics

Exemplo 3
Pergunta: Which of the following are alternative names for the independent variable (usually denoted by x) in linear regression analysis?
(i) The regressor
(ii) The regressand
(iii) The causal variable
(iv) The effect variable
Resposta Correta: 1
Resposta Gerada: 3
Categoria: Econometrics

4 Discussão

4.1 O ganho na tarefa de Text-to-SQL justifica a perda de capacidade geral?

A partir dos resultados obtidos, é evidente que o fine-tuning com LoRA trouxe ganhos substanciais de especialização na tarefa de Text-to-SQL. A *Execution Accuracy* aumentou de 52% para 76%, representando uma melhora de 24 pontos percentuais. Esse crescimento reflete um modelo muito mais capacitado para resolver consultas SQL diretamente a partir de linguagem natural.

Por outro lado, a regressão de capacidade geral, avaliada por meio do benchmark MMLU, foi de apenas 2.06 pontos percentuais (de 39.73% para 37.67%). Esse nível de perda pode ser considerado aceitável, principalmente porque a tarefa-alvo (Text-to-SQL) se tornou significativamente mais precisa e o impacto na capacidade geral não foi catastrófico, permanecendo dentro de uma faixa tolerável.

Portanto, o ganho obtido no domínio especializado justifica a pequena regressão de capacidade geral. Esse é um trade-off comum e aceitável no contexto de especialização de modelos.

4.2 Quais fatores influenciaram o trade-off?

Os principais fatores que influenciaram esse trade-off foram:

4.2.1 *Hiperparâmetros.*

- **Taxa de aprendizado:** Uma taxa de 2×10^{-4} foi utilizada. Essa configuração relativamente alta pode acelerar a especialização, mas também tende a favorecer um leve esquecimento de tarefas gerais.
- **Número de épocas:** O treinamento foi feito com apenas **1 época**, o que provavelmente limitou a regressão de capacidade geral, uma vez que o modelo não foi excessivamente exposto ao domínio específico.

4.2.2 *Arquitetura.*

- **Uso de LoRA:** A técnica de Parameter-Efficient Fine-Tuning (PEFT) via LoRA restringe as atualizações a pequenos adaptadores, preservando os pesos originais do modelo. Isso é um dos motivos pelos quais a regressão de capacidade geral foi relativamente baixa.
- **Camadas-alvo:** Os adaptadores LoRA foram inseridos em módulos críticos de atenção (*q_proj*, *k_proj*, *v_proj*, *o_proj*), que são os principais responsáveis por manipular a entrada textual no modelo. A escolha desses módulos favorece a especialização rápida, mas mantém a base geral relativamente intacta.

4.3 Implicações Práticas

Os achados deste experimento indicam que:

- É viável especializar modelos de LLM para tarefas como Text-to-SQL sem comprometer severamente sua capacidade geral.

- O uso de LoRA se mostra uma excelente estratégia para balancear custo computacional, rapidez de adaptação e preservação do conhecimento geral.
- Pequenos ajustes nos hiperparâmetros, especialmente no número de épocas ou na taxa de aprendizado, podem permitir um ajuste ainda mais fino desse equilíbrio.

4.3.1 Aplicações Comerciais. Para LLMs voltados a soluções empresariais (por exemplo, geração de relatórios, consultas a banco de dados via linguagem natural, automação de BI), a especialização via LoRA permite:

- **Implementação mais rápida** de modelos específicos para cada cliente ou setor.
- **Baixo custo de adaptação**, visto que os adaptadores LoRA são leves.
- **Redução de risco de esquecimento catastrófico**, mantendo o modelo capaz de lidar com perguntas gerais mesmo após o fine-tuning.

5 Conclusão

O modelo fine-tuned demonstrou um ganho substancial de especialização na tarefa Text-to-SQL, com um aumento de 24 pontos percentuais na execução correta de queries. A regressão de capacidade geral foi pequena, aproximadamente 2%, e pode ser considerada um trade-off aceitável.

As principais falhas ocorreram em cenários complexos de construção de queries, especialmente envolvendo:

- JOINs múltiplos e complexos.
- Referência incorreta de nomes de colunas.
- Construção incompleta ou incorreta de operações como `INTERSECT`.

No benchmark MMLU, os erros ocorreram principalmente por confusão conceitual em perguntas específicas, sem impacto crítico na estrutura global de conhecimento.

Conclusão Geral: O modelo treinado representa uma alternativa promissora para aplicações especializadas, com perdas mínimas em tarefas de conhecimento geral.

References

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research* 21, 140 (2020), 1–67.