Leis de Escala para *Text-to-SQL*: Um Estudo sobre a Relação entre Tamanho e Desempenho de Modelos de Linguagem

Letícia O. Silva[1], **Paulo H. C. Silva**[1], **Fabrício A. Silva** [1]

[1]Instituto de Ciências Exatas e Tecnológicas
Universidade Federal de Viçosa - Campus Florestal (UFV-CAF)

{leticia.silva1, paulo.h.carneiro, fabricio.silva}@ufv.br

Abstract. *Although large language models achieve good results in the Text-to-SQL task, their high computational cost limits adoption by small businesses. This study evaluates the feasibility of small language models as an alternative, analyzing the trade-off between model size and performance for the open-source Qwen2.5, in variants from 0.5B to 32B parameters. Experiments were conducted on Spider benchmark and on a database with information about Brazilian companies, to evaluate the effectiveness of the approach in a real-world application context. Results show that the 3B model represents the best balance between cost and performance, but the 14B and 32B models, although more costly, offer superior performance.*

Resumo. *Embora grandes modelos de linguagem obtenham bons resultados na tarefa de Text-to-SQL, o alto custo computacional limita a adoção dos mesmos. Este estudo avalia a viabilidade de pequenos modelos como alternativa, analisando a relação entre tamanho e desempenho, utilizando o modelo Qwen2.5 nas variantes de 0.5B a 32B de parâmetros. Experimentos foram realizados no benchmark Spider e em um banco de dados com informações de empresas brasileiras, com o objetivo de analisar a eficácia da abordagem em um contexto de aplicação real. Os resultados indicam que o modelo de 3B oferece o melhor equilíbrio entre custo e desempenho, enquanto os de 14B e 32B, embora mais caros, apresentam desempenho superior.*

1. Introdução

O acesso a dados armazenados em bancos de dados relacionais é fundamental para diversos sistemas modernos, como plataformas de inteligência de negócios e ferramentas de gestão de relacionamento com clientes [Mohammadjafari et al. 2024]. A tarefa de *Text-to-SQL*, que consiste em traduzir linguagem natural para consultas SQL, tem se destacado como uma solução promissora para ampliar o acesso a esses dados, especialmente por usuários sem domínio da linguagem SQL, como gestores e pessoas de negócio. Por meio dessa abordagem, esses usuários podem interagir com os dados utilizando linguagem natural, como ilustra a Figura 1, o que facilita a exploração e o uso de informações em diferentes contextos.

Figura 1. Exemplo da tarefa de *Text-to-SQL*.

Grandes modelos de linguagem (LLMs), uma das inovações mais relevantes em Inteligência Artificial (IA) [Miranda and Campelo 2024], têm demonstrado resultados promissores nessa tarefa. No entanto, esses modelos exigem recursos computacionais elevados, o que implica altos custos financeiros e dificulta sua adoção em cenários mais restritos. Dessa forma, embora soluções baseadas em LLMs dominem o estado da arte, suas limitações evidenciam a necessidade de buscar abordagens alternativas mais acessíveis, como o uso de pequenos modelos de linguagem (SLMs).

Este trabalho, motivado pela demanda de uma empresa, tem como objetivo avaliar a relação entre o tamanho dos modelos e a acurácia de execução (EX) das consultas SQL geradas, a fim de identificar oportunidades para o uso de soluções baseadas em SLMs. Embora seja esperado que modelos com maior número de parâmetros apresentem melhor desempenho, modelos menores podem alcançar resultados satisfatórios quando se considera o *trade-off* entre custo computacional e desempenho. Para investigar essa hipótese, foi selecionado um modelo de linguagem *open-source*, o Qwen2.5, pré-treinado e de propósito geral, disponível em diferentes tamanhos, o que permite uma análise comparativa justa do desempenho na tarefa de *Text-to-SQL*. Para enriquecer o conjunto experimental, também investigamos o desempenho ao aplicar estratégias de *schema-linking* e de realizar *fine-tuning* supervisionado. Os experimentos foram realizados no conjunto de dados Spider [Yu et al. 2018], amplamente utilizado na literatura, e também em um banco de dados estruturado a partir de informações de empresas brasileiras disponibilizadas pela Receita Federal, utilizado como estudo de caso aplicado.

Dessa forma, avaliamos diferentes estratégias de geração de consultas utilizando um modelo *open-source* em seis variações de tamanho: 0.5B, 1.5B, 3B, 7B, 14B e 32B. Com isso, contribuímos com diversos resultados experimentais que servem como critérios para a escolha do tamanho de modelo a ser utilizado, além de motivar novos estudos sobre o uso de pequenos modelos de linguagem como uma alternativa mais acessível. De modo geral, apesar de modelos maiores (14B e 32B) apresentarem resultados superiores, o modelo de 3B representa o melhor ponto de equilíbrio entre custo computacional e desempenho, com resultados expressivos especialmente em consultas *easy* e *medium*.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados; a Seção 3 descreve a metodologia utilizada; a Seção 4 aborda o ambiente de testes; a Seção 5 apresenta os resultados obtidos; por fim, a conclusão e os trabalhos futuros estão na Seção 6.

2. Trabalhos Relacionados

A tarefa de conversão de linguagem natural para SQL (*Text-to-SQL*) tem sido amplamente estudada na área de Processamento de Linguagem Natural, com avanços significativos ao longo dos anos [Volvovsky et al. 2024], especialmente com o advento dos grandes modelos de linguagem [Liu et al. 2024].

Parte considerável das pesquisas recentes tem se concentrado no uso de LLMs, explorando técnicas como *prompt engineering* e *in-context learning* [Dong et al. 2023, Gao et al. 2024], além da decomposição da tarefa em subtarefas como *schema-linking* e *self-correction* [Wang et al. 2025]. Apesar dos avanços, soluções baseadas em grandes modelos estão relacionadas a uma alta demanda computacional, o que limita sua adoção em cenários com restrições de recursos. Além disso, muitas dessas abordagens são base-

adas em modelos proprietários, os quais, além do custo, apresentam limitações quanto à transparência e privacidade dos dados [Hong et al. 2024].

Modelos menores, especialmente aqueles com até 3B de parâmetros, ainda são pouco explorados. O trabalho DTS-SQL [Pourreza and Rafiei 2024] propõe uma arquitetura em duas etapas: um modelo realiza o *schema-linking*, enquanto outro gera a consulta SQL. Embora utilize *fine-tuning* supervisionado com modelos de 7B (considerados "pequenos" no contexto do estudo), a abordagem requer dois modelos distintos, sem avaliar se um único modelo de tamanho intermediário, que ocupe espaço de memória equivalente, poderia alcançar desempenho similar ou superior com menor complexidade operacional.

O CodeS [Li et al. 2024a] apresenta uma família de modelos pré-treinados, com tamanhos variando de 1B a 15B de parâmetros, voltados para a tarefa de *Text-to-SQL*. Embora sejam de código aberto e incluam variantes menores, os modelos são pré-treinados especificamente para essa tarefa, estratégia que demanda muitos recursos computacionais e dificulta a replicação. Além disso, os autores propõem adotar estratégia para adaptar o modelo ao domínio do banco de dados, o que compromete a generalização da solução.

[Oliveira et al. 2024] investigam a influência do tamanho dos modelos na tarefa de *Text-to-SQL*. No entanto, a comparação é feita entre modelos de diferentes famílias e de diferentes versões, o que dificulta isolar o efeito do tamanho. Além disso, a estratégia de *schema-linking* é utilizada apenas para enriquecer o contexto com informações sobre tabelas e colunas relevantes, e não como uma estratégia para filtrar informações irrelevantes ao modelo. Os experimentos são restritos a um único estudo de caso, cujo banco de dados e conjunto de teste não são disponibilizados publicamente, comprometendo a reprodutibilidade e a comparabilidade das estratégias adotadas com outros trabalhos.

Nosso trabalho contribui com a análise de diferentes estratégias em modelos da mesma família e versão para investigar o impacto do tamanho do modelo na tarefa de *Text-to-SQL*. Com isso, oferece uma base experimental para auxiliar em decisões sobre o uso de modelos conforme as restrições e objetivos de aplicação. Além disso, avaliamos as abordagens tanto em um *benchmark* amplamente utilizado quanto em um estudo de caso aplicado. O banco de dados, conjunto de dados de teste e código-fonte estão disponíveis[1] publicamente, promovendo transparência e replicabilidade. Por fim, nossos experimentos incluem análises segmentadas por complexidade das consultas, o que permite insights mais aprofundados sobre o desempenho dos modelos em diferentes cenários.

3. Metodologia

Para investigar o impacto do tamanho do modelo na tarefa de *Text-to-SQL*, realizamos experimentos utilizando a família de modelos pré-treinados de propósito geral e *open-source* Qwen2.5 na versão `Instruct`, disponível no Hugging Face[2], nas variações de 0.5B, 1.5B, 3B, 7B, 14B e 32B de parâmetros. A escolha pelo Qwen2.5 se deve não apenas ao fato de ser um modelo *open-source*, o que favorece a reprodutibilidade dos experimentos, mas também por estar disponível em diferentes tamanhos na mesma família e versão, característica essencial para garantir uma análise comparativa consistente dos efeitos da escala. A Tabela 1 consolida os cenários investigados, detalhados a seguir.

[1]https://github.com/lleticiasilvaa/LeisDeEscala-Text2SQL

[2]huggingface.co/Qwen

Tabela 1. Notações das abordagens e variações utilizadas nos experimentos

Notação	Descrição
`BASELINE`	Geração da consulta SQL pelo modelo de propósito geral com esquema completo.
`SFT`	Modelo após *fine-Tuning* supervisionado em 1 *epoch* no conjunto de dados de treino do Spider.
`VEX`	Geração da consulta SQL com esquema completo e valores de exemplos.
`SQL-SQL`	Geração inicial da consulta SQL com esquema completo (com exemplos), seguida da filtragem do esquema com base na consulta e geração de uma nova consulta SQL com esquema reduzido (também com exemplos).
`SL-SQL`	Identificação de tabelas e colunas relevantes (*schema-linking*) em formato JSON, seguida da geração da consulta SQL com base no esquema filtrado (com exemplos).
`tabelas`	Esquema reduzido apenas com base nas tabelas relevantes identificadas.
`tabelas&colunas`	Esquema reduzido com base nas tabelas e colunas relevantes identificadas.

Os experimentos iniciais consistem em fornecer um *prompt* que instrui o modelo a gerar uma consulta SQL a partir de uma consulta em linguagem natural, juntamente com o esquema completo do banco de dados correspondente, conforme ilustrado na Figura 2. Essa configuração experimental é adotada como nossa abordagem `Baseline`. Visando viabilizar o uso desses modelos em contextos com recursos limitados e contribuir com a democratização de soluções para a tarefa de *Text-to-SQL*, os modelos foram carregados com quantização de 4 bits, utilizando a biblioteca `bitsandbytes`. Embora esses experimentos já permitam observar a relação entre o tamanho do modelo e seu desempenho, também avaliamos estratégias complementares com o objetivo de verificar se promovem ganhos adicionais de desempenho e, em caso positivo, em que medida esses ganhos variam conforme o tamanho do modelo.

```
System
Given a user question and the schema of a database, your task is to generate an
SQL query that accurately answers the question based on the provided schema.

User
# Schema:
"'sql
{DATABASE SCHEMA}
"'
# Question:{NATURAL LANGUAGE QUERY}
```

Figura 2. *Prompt* para abordagem `BASELINE`

Considerando o objetivo de avaliar o desempenho de modelos de propósito geral em uma tarefa específica, a primeira estratégia avaliada foi a de *fine-tuning* supervisionado (`SFT`), que envolve o treinamento adicional do modelo em dados rotulados específicos de uma tarefa, aproveitando o pré-treinamento para se especializar em domínios específicos, como *Text-to-SQL* [Zhang et al. 2024]. Essa abordagem é significativamente mais acessível do que treinar um modelo do zero, mas ainda assim, o *fine-tuning* completo demanda recursos computacionais elevados. Para contornar essa limitação, existem técnicas de otimização como PEFT [Xu et al. 2023] e QLoRA [Dettmers et al. 2023] que permitem ajustar apenas alguns parâmetros do modelo quantizado. Considerando essas técnicas, realizamos o *fine-tuning* supervisionado dos modelos com até 14B de parâmetros, devido à limitações de memória de GPU, utilizando o conjunto de treino do *benchmark* Spider [Yu et al. 2018], que contém 8659 exemplos distribuídos em 200 bancos de dados de diferentes domínios.

Como essa estratégia envolve uma etapa custosa e demorada, também avaliamos alternativas baseadas em engenharia de *prompt* utilizando os modelos de 3B, 7B, 14B e

32B de parâmetros. A segunda estratégia (`VEX`), consistiu em enriquecer o esquema do banco de dados com exemplos de valores armazenados. No entanto, como as consultas geralmente envolvem apenas uma parte das tabelas e colunas, fornecer o esquema completo significa fornecer informações irrelevantes, podendo prejudicar o desempenho do modelo. Nesse contexto, a terceira e última estratégia avaliada foi a de *schema-linking*, cujo objetivo é identificar as tabelas e colunas relevantes à consulta. Essa estratégia, que pode ser considerada um subtarefa da tarefa de *Text-to-SQL*, possibilita filtrar informações irrelevantes do banco de dados, seja removendo tabelas inteiras (`tabelas`), ou realizando uma filtragem mais detalhada, que considera tanto tabelas quanto colunas (`tabelas&colunas`).

Adotamos duas abordagens para realizar o *schema-linking*. A primeira consiste em solicitar ao modelo a geração de uma consulta SQL inicial e a partir dessa primeira consulta, extraímos as tabelas e colunas utilizadas para filtrar o esquema, gerando então uma segunda consulta. Essa abordagem foi inspirada em [Li et al. 2024b] e [Yang et al. 2024], com a principal diferença sendo o uso de um *prompt* distinto na etapa inicial. Como o modelo executa a tarefa de geração de consulta SQL duas vezes, adotamos a nomenclatura `SQL-SQL` para essa abordagem.

Na segunda abordagem, a identificação das tabelas e colunas relevantes é atribuída ao próprio modelo, de forma que: primeiro, o modelo realiza o *schema-linking* a partir da consulta em linguagem natural e retorna um JSON contendo as tabelas e colunas relevantes. Com essas informações, o esquema é filtrado e passado ao modelo para gerar a consulta SQL. Essa estratégia foi baseada em [Pourreza and Rafiei 2024] mas sem a realização de *fine-tuning* em nenhuma das etapas e utilizando o mesmo modelo para ambas. Como o modelo executa primeiro o *schema-linking* e depois a geração da consulta SQL, utilizamos a nomenclatura `SL-SQL`.

4. Ambiente de Testes

Os experimentos foram realizados utilizando uma GPU Nvidia A10 com 24 GB de VRAM. Todos os testes foram conduzidos usando *greedy search*, onde o próximo *token* na saída é sempre aquele com a maior probabilidade. A temperatura foi definida como zero e o limite máximo de novos *tokens* como 300.

4.1. Conjunto de Dados

Foram utilizados dois conjuntos de dados para a avaliação dos modelos. O primeiro é o *split* de desenvolvimento do *benchmark* Spider [Yu et al. 2018], composto por 1034 exemplos distribuídos em 200 bancos de dados de domínios variados. O segundo é o conjunto `CNPJ`, com 200 exemplos de consultas em linguagem natural projetadas para conduzir um estudo de caso aplicado, voltado a uma demanda prática de negócios: a filtragem e segmentação de empresas com base em consultas em linguagem natural. O banco de dados correspondente foi construído a partir de dados públicos[3] do Cadastro Nacional da Pessoa Jurídica (CNPJ), disponibilizados pela Receita Federal do Brasil, os quais oferecem informações detalhadas sobre empresas registradas no país. A extração dos dados foi realizada com base em scripts disponíveis no GitHub[4]. Após a extração, o

[3]dados.gov.br/cnpj

[4]https://github.com/aphonsoar/Receita_Federal_do_Brasil_-_Dados_Publicos_CNPJ

esquema do banco foi reorganizado: tabelas e atributos foram renomeados e traduzidos para o inglês, novas tabelas foram criadas para melhorar a estrutura relacional, e chaves foram adicionadas para garantir a integridade referencial.

4.2. Métricas

Para avaliar os modelos na tarefa *Text-to-SQL* utilizamos a métrica de acurácia de execução (EX), que mede a proporção de exemplos para os quais os resultados de execução das consultas SQL geradas pelo modelo produz o mesmo resultado que a consulta SQL de referência, em relação ao número total de consultas [Wang et al. 2025]. Essa métrica foi calculada utilizando o *Test Suite Accuracy*[5], proposto em [Zhong et al. 2020] que, além da métrica geral (*all*), calcula também as métricas segmentadas por nível de dificuldade da consulta (*easy*, *medium*, *hard* e *extra hard*), classificando as consultas com base na taxonomia definida no Spider [Yu et al. 2018]. A Tabela 2 mostra a distribuição das consultas por nível de dificuldade em cada conjunto de dados.

Tabela 2. Distribuição por Nível de Dificuldade nos conjuntos Spider-Dev e CNPJ

Conjunto	*easy*	*medium*	*hard*	*extra hard*	Total
Spider-Dev	248 (23,98%)	446 (43,13%)	174 (16,83%)	166 (16,05%)	1034
CNPJ	35 (17,5%)	61 (30,5%)	24 (12%)	80 (40%)	200

Além da avaliação de desempenho, também consideramos o custo computacional. Avaliamos o uso de memória de cada modelo e calculamos a quantidade média de *tokens* necessários para compor o *prompt*, considerando diferentes estratégias de *schema* (completo, reduzido por tabelas e reduzido por tabelas e colunas), sempre com um limite de até 300 *tokens* gerados na resposta. Com isso, estimamos o custo de inferência com base nos valores de cobrança por milhão de *tokens* disponíveis na plataforma *Together.ai*[6].

4.3. *Fine-Tuning* Supervisionado

A estratégia de *fine-tuning* supervisionado foi implementada com o `SFTTrainer` da biblioteca `trl`, em conjunto com a técnica QLoRA [Dettmers et al. 2023]. Para o ajuste dos parâmetros com LoRA [Hu et al. 2022], adotamos as seguintes configurações: rank (r) igual a 64, α (`lora_alpha`) igual a 128, e dropout (`lora_dropout`) de 0.1. Essas configurações foram aplicadas a todos os módulos lineares do modelo, definidos nos `target_modules`. O comprimento máximo de sequência (`max_seq_length`) foi definido como 2048.

O treinamento foi realizado ao longo de uma única *epoch* no conjunto de dados de treino do Spider [Yu et al. 2018]. Utilizamos o otimizador `paged_adamw_8bit`, com taxa de aprendizado (*learning rate*) de $1e^{-4}$, decaimento de peso (`weight_decay`) de 0.001 e norma máxima de gradiente (`max_grad_norm`) de 0.3. A taxa de aprendizado foi ajustada conforme uma curva cosseno, com razão de aquecimento linear de 0.03 (`warmup_ratio`). O tamanho de lote efetivo empregado foi de 16, e os dados de treinamento foram agrupados conforme o comprimento das sequências (`group_by_length`).

[5]https://github.com/taoyds/test-suite-sql-eval

[6]together.ai/pricing

5. Resultados

Nossa primeira avaliação investiga o uso de modelos de propósito geral na tarefa de *Text-to-SQL*, considerando perguntas em linguagem natural acompanhadas do esquema completo do banco de dados. O objetivo é analisar a relação entre o tamanho do modelo e seu desempenho. O modelo base utilizado foi o `Qwen2.5 Instruct`, e, considerando os recursos computacionais disponíveis, avaliamos as variantes com 0.5B, 1.5B, 3B, 7B, 14B e 32B de parâmetros com quantização de 4 bits, o que pode resultar em uma perda de desempenho devido à redução na precisão dos parâmetros, mas proporciona uma redução significativa no uso de memória.

Tabela 3. Desempenho (EX) nos conjuntos Spider-Dev e CNPJ (`BASELINE`)

	Spider-Dev					CNPJ				
	Easy	Medium	Hard	Extra Hard	All	Easy	Medium	Hard	Extra Hard	All
0.5B	37,1	22,0	11,5	13,3	22,4	51,4	41,0	12,5	5,0	25,0
1.5B	55,6	41,0	28,7	21,7	39,4	60,0	73,8	33,3	13,8	42,5
3B	74,2	62,8	45,4	27,7	57,0	91,4	86,9	62,5	35,0	64,0
7B	87,5	76,5	62,6	45,2	71,8	97,1	88,5	70,8	53,7	74,0
14B	86,7	81,2	64,4	48,8	74,5	100	82,0	66,7	52,5	71,5
32B	90,3	82,5	69,0	53,6	77,5	97,1	90,2	70,8	68,8	80,5

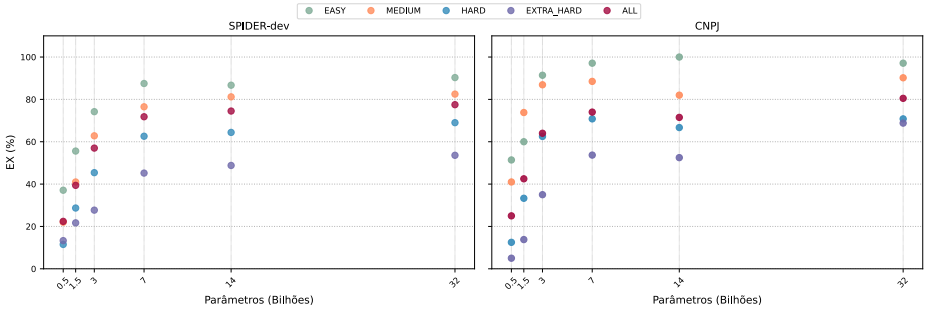

Figura 3. Desempenho dos Modelos (`BASELINE`)

Os resultados apresentados na Tabela 3 indicam uma correlação positiva entre o tamanho dos modelos `Qwen2.5 Instruct` e seu desempenho na tarefa de *Text-to-SQL*, tanto no conjunto Spider-Dev quanto no conjunto CNPJ. No entanto, como mostra o gráfico da Figura 3, essa correlação não é linear: os maiores ganhos ocorrem nas versões menores dos modelos, especialmente até 7B parâmetros. A partir desse ponto, os aumentos de desempenho tornam-se marginais. Por exemplo, no Spider-Dev, o salto do modelo de 3B para 7B resulta em um ganho de 14,8 pontos percentuais na acurácia de execução geral (EX - all), enquanto o ganho do modelo de 14B em relação ao 7B é de apenas 2,7 pontos percentuais, e do 14B para o 32B, de 3 pontos percentuais. Essa tendência se repete em todos os níveis de dificuldade (*easy*, *medium*, *hard* e *extra hard*).

5.1. *Fine-Tuning* Supervisionado

Para esta etapa, realizamos *fine-tuning* supervisionado utilizando apenas o conjunto de treinamento do Spider, limitando a experimentação a modelos de até 14B parâmetros devido a restrições de GPU. Os resultados obtidos estão na Tabela 4.

Observamos que, de maneira geral, como mostra o gráfico da Figura 4, o *fine-tuning* trouxe ganhos de desempenho no conjunto Spider-Dev. Por exemplo, o modelo de 0.5B saltou de 22,4% para 37,3% na acurácia de execução geral, e o de 14B alcançou 81,2%, um avanço significativo em relação aos 74,5% obtidos sem *fine-tuning*. Esses ganhos são evidentes em todos os níveis de dificuldade, especialmente nos casos mais complexos o que sugere que o treinamento supervisionado foi eficaz em adaptar os modelos ao estilo e à semântica desse *benchmark*.

No entanto, o impacto no conjunto CNPJ foi bastante distinto. Os resultados não seguiram uma tendência crescente e em alguns casos houve queda de desempenho. Por exemplo, o modelo de 3B passou de 64,0% para 44,0%, e o modelo de 14B, mesmo com 100% de acerto em consultas *easy*, apresentou desempenho inferior em consultas *extra hard* (de 52,5% para 37,5%).

Esse comportamento pode ser atribuído ao fato de que o *fine-tuning* foi realizado exclusivamente com dados do conjunto de treino do Spider. Como consequência, pode ter ocorrido *overfitting*, com os modelos se especializando em padrões específicos do *benchmark* Spider, mas generalizando mal para domínios distintos como o CNPJ, que possui estrutura e distribuição de consultas consideravelmente diferentes.

Tabela 4. Desempenho (EX) nos conjuntos Spider-Dev e CNPJ dos Modelos com *Fine-Tuning* Supervisionado (`SFT`)

	Spider-Dev					CNPJ				
	Easy	Medium	Hard	Extra Hard	All	Easy	Medium	Hard	Extra Hard	All
0.5B	62,5	36,3	27,0	13,3	37,3	82,9	18,0	8,3	0	21,0
1.5B	87,5	73,5	54,0	38,0	67,.9	71,4	65,6	16,7	11,3	39,0
3B	78,6	61,4	43,7	29,5	57,4	94,3	65,6	33,3	8,7	44,0
7B	87,5	76,0	66,1	47,6	72,5	94,3	83,6	54,2	48,7	68,0
14B	91,5	87,4	70,7	60,2	81,2	100	83,6	62,5	37,5	65,5

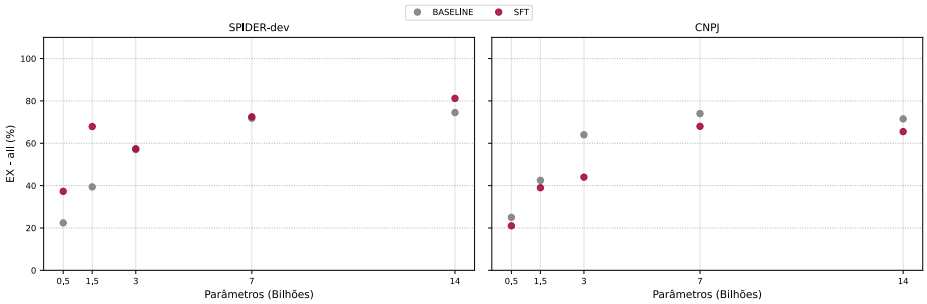

Figura 4. Desempenho dos Modelos - `SFT` - Métricas Gerais (*all*)

5.2. Adicionar exemplos de valores

Visto que o uso de *fine-tuning* supervisionado com os modelos Qwen2.5 resultou em ganhos apenas no *benchmark* Spider, alteramos para uma abordagem de *prompt engineering*. Inicialmente, investigamos o impacto de incluir exemplos de valores dos dados armazenados no banco de dados em relação à estratégia `Baseline`. A Tabela 5 apresenta os resultados dessa modificação para os conjuntos Spider-Dev e CNPJ.

De forma geral, como mostra o gráfico da Figura 5, a adição de exemplos de valores resultou em ganhos marginais de desempenho na maioria dos modelos. Por exemplo, no Spider-Dev, o modelo de 3B passou de 57% para 61,0%, enquanto no CNPJ o modelo de 14B subiu de 71,5% para 78,5%. O modelo de 32B obteve 83,5% no CNPJ e 80,4% no Spider-Dev, configurando o melhor desempenho geral até o momento. Embora os ganhos não tenham sido expressivos, o uso de exemplos de valores foi adotado como nova `Baseline` para as próximas estratégias, pois, na maioria dos casos, houve alguma melhora.

Uma exceção notável foi o modelo de 1.5B, cujo desempenho caiu em ambos os conjuntos após a adição de exemplos de valores. Esse resultado sugere que, para modelos menores, a introdução de informações adicionais pode aumentar a complexidade do *prompt* a ponto de confundir o modelo, especialmente em consultas simples.

Tabela 5. Desempenho nos conjuntos Spider-Dev e CNPJ Adicionando Valores de Exemplos (VEX)

	Spider-Dev					CNPJ				
	Easy	Medium	Hard	Extra Hard	All	Easy	Medium	Hard	Extra Hard	All
0.5B	48,4	27,6	15,5	9,0	27,6	65,7	34,4	16,7	3,7	25,5
1.5B	54,8	47,3	30,5	27,7	43,1	68,6	31,1	16,7	8,7	27,0
3B	79,0	66,1	47,7	34,3	61,0	94,3	80,3	45,8	41,2	63,0
7B	83,9	79,1	61,5	48,2	72,3	100	90,2	75,0	53,7	75,5
14B	85,1	79,6	63,2	47,6	73,0	97,1	86,9	79,2	63,7	78,5
32B	91,1	85,7	75,9	54,8	80,4	100	88,5	95,8	68,8	83,5

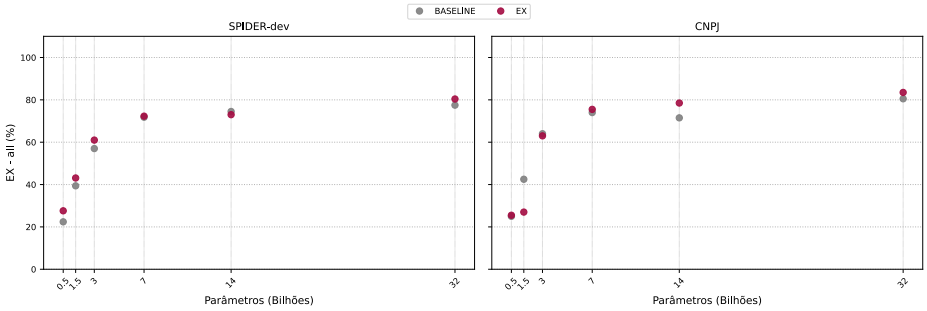

Figura 5. Desempenho dos Modelos - EX - Métricas Gerais (*all*)

Dessa forma, concluímos que, embora o impacto da inclusão de valores não seja substancial para modelos maiores, a técnica é benéfica o suficiente para ser incorporada como padrão. As estratégias de *schema-linking* avaliadas nas próximas seções consideram essa modificação no *prompt* como parte da configuração base.

5.3. Schema-Linking

Para enriquecer nossa análise experimental, avaliamos duas estratégias de *schema-linking*, aplicadas apenas aos modelos de 3B, 7B, 14B e 32B. Os modelos de 0.5B e 1.5B de parâmetros foram excluídos desta etapa devido ao desempenho inferior apresentado nas etapas anteriores, quando comparados aos demais modelos. A Tabela 6 apresenta os resultados obtidos.

Tabela 6. Resultados por Estratégia e Tamanho, com separação por dificuldade

	Estratégia	Esquema	Spider-Dev					CNPJ				
			Easy	Medium	Hard	Extra Hard	All	Easy	Medium	Hard	Extra Hard	All
	VEX		79,0	66,1	47,7	34,3	61,0	94,3	80,3	45,8	41,2	63,0
	SQL-SQL	tables	83,1	72,9	50,0	36,1	**65,6**	85,7	72,1	50,0	42,5	60,0
3B	SQL-SQL	tables&columns	73,0	67,3	51,1	38,6	61,3	85,7	82,0	70,8	47,5	**67,5**
	SL-SQL	tables	81,9	66,6	45,4	33,1	61,3	91,4	82,0	54,2	38,8	63,0
	SL-SQL	tables&columns	62,9	44,2	32,8	31,3	44,7	60,0	27,9	29,2	25,0	32,5
	VEX		83,9	79,1	61,5	48,2	72,3	100	90,2	75,0	53,7	75,5
	SQL-SQL	tables	86,3	83,2	66,1	50,6	**75,8**	100	88,5	58,3	56,2	74,0
7B	SQL-SQL	tables&columns	83,1	78,5	62,6	51,2	72,5	100	90,2	66,7	63,7	**78,5**
	SL-SQL	tables	85,1	82,7	67,2	50,6	75,5	97,1	86,9	62,5	57,5	74,0
	SL-SQL	tables&columns	79,4	68,6	58,0	44,0	65,5	91,4	63,9	58,3	45,0	60,5
	VEX		85,1	79,6	63,2	47,6	73,0	97,1	86,9	79,2	63,7	78,5
	SQL-SQL	tables	90,3	83,2	66,7	50,6	76,9	97,1	90,2	83,3	67,5	81,5
14B	SQL-SQL	tables&columns	87,5	82,7	65,5	48,8	75,5	97,1	91,8	66,7	68,8	80,5
	SL-SQL	tables	90,7	83,2	69,5	50,6	**77,5**	97,1	90,2	87,5	77,5	**86,0**
	SQL-SQL	tables&columns	89,1	83,0	63,8	53,0	76,4	97,1	86,9	75,0	62,5	77,5
	VEX		91,1	85,7	75,9	54,8	80,4	100	88,5	95,8	68,8	83,5
	SQL-SQL	tables	92,3	88,1	75,3	53,6	81,4	97,1	90,2	87,5	67,5	82,0
32B	SQL-SQL	tables&columns	90,3	85,0	77,0	56,6	80,4	97,1	90,2	87,5	71,3	83,5
	SL-SQL	tables	92,3	89,7	71,3	59,6	**82,4**	100	91,8	91,7	63,7	82,0
	SQL-SQL	tables&columns	92,3	87,0	73,0	59,6	81,5	100	91,8	91,7	73,8	**86,0**

De maneira geral, como mostra o gráfico da Figura 6, a estratégia SQL−SQL com filtragem apenas de tabelas trouxe ganhos mais significativos no conjunto do Spider, e podemos observar que o ganho é maior no modelo de 3B com 7,5%, enquanto no de 32B

o ganho foi de apenas 1,2%. Já no CNPJ, essa estratégia reduzindo tabelas e colunas foi a que representou um maior ganho de desempenho, e novamente, quanto menor o modelo, maior foi o impacto. O modelo de 3B teve aumento de 7,1%, o de 7B, 4%, o de 14B, 2,5%, e o de 32B não apresentou ganhos. O modelo de 14B teve um ganho de desempenho considerável, tanto no Spider quanto no CNPJ, com a estratégia de `SL-SQL` e redução apenas de tabelas, com 6,2% de aumento no Spider e 9,6% no CNPJ, indicando capacidade do modelo em realizar a tarefa de *schema-linking*.

Observando apenas o modelo de 32B, adotar ou não as estratégias quase não impacta o desempenho, nem positivamente, nem negativamente. Já no de 3B, as estratégias trazem mais variações, inclusive com queda significativa ao adotar `SL-SQL` com redução de tabelas e colunas, indicando baixa capacidade do modelo em fazer *schema-linking*, especialmente com colunas, e que faltar informações tem um impacto bem maior do que fornecer informações irrelevantes.

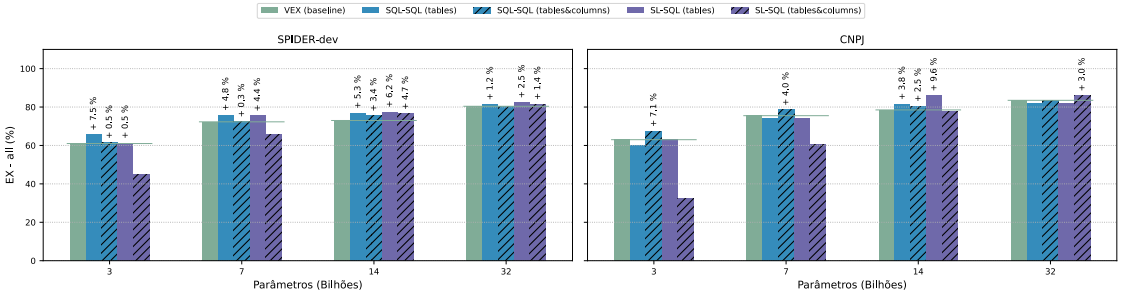

Figura 6. Comparação entre `VEX`, `SQL-SQL` e `SL-SQL` - Desempenho Geral (*all*)

Avaliando por nível de dificuldade, para consultas *easy*, nenhuma estratégia trouxe ganhos no CNPJ, enquanto no Spider, `SQL-SQL` e `SL-SQL` com redução apenas de tabelas, apresentaram pequenos ganhos. Como as consultas são de nível fácil, não é necessário resolver o problema em duas etapas, como ocorre nas estratégias com *schema-linking*. Já para consultas do tipo *hard* e *extra hard*, as estratégias de *schema-linking*, principalmente `SQL-SQL` com filtragem de tabelas e colunas, foram mais eficazes, evidenciando que o impacto das estratégias aumenta conforme cresce a complexidade da tarefa. Isso reforça a importância do *schema-linking* em cenários mais desafiadores, nos quais a compreensão da estrutura do banco se torna mais crítica.

5.4. Discussão

Esta seção discute a relação custo–desempenho dos modelos da família Qwen2.5 na tarefa de *Text-to-SQL*, avaliada no *benchmark* Spider-Dev. Os preços de inferência (USD por 1M *tokens*) foram obtidos na plataforma *Together.ai*[7] e estão sumarizados na Tabela 7. Para o modelo de 32B parâmetros, consideramos a variante *Qwen2.5 Coder 32B*, única opção ofertada. Para os modelos de 0.5B, 1.5B e 3B, adotou-se o valor padrão aplicado pela plataforma aos demais modelos com até 4B parâmetros.

Os resultados apresentados nas Tabelas 5 e 6, combinados com os custos de inferência da Tabela 7, revelam que o modelo Qwen2.5-3B oferece o melhor compromisso entre desempenho e custo. Com um valor de inferência de apenas $0,10 por milhão de *tokens*, o modelo alcança 65,6% de EX geral, com desempenho expressivo inclusive em

[7]Disponível em: together.ai/pricing. Acesso em maio/2025.

consultas de dificuldade *medium* (72,9%). Tal resultado representa um avanço significativo frente aos modelos menores, Qwen2.5-0.5B e Qwen2.5-1.5B, que compartilham o mesmo custo de inferência, mas apresentam desempenho geral consideravelmente inferior (27,6% e 43,1%, respectivamente).

A partir do Qwen2.5-7B, observa-se um ganho contínuo de acurácia, especialmente nas categorias de maior complexidade, como *hard* e *extra hard*. Contudo, esses avanços vêm acompanhados de aumentos substanciais no custo: os modelos Qwen2.5-14B e Qwen2.5-32B, por exemplo, atingem acurácia geral de 77,5% e 82,4%, respectivamente, porém com custo de inferência oito vezes maior que o de Qwen2.5-3B. Embora apresentem desempenho superior, sobretudo em tarefas mais desafiadoras, a razão acurácia por dólar decresce à medida que o modelo escala, refletindo o padrão clássico de retornos decrescentes frequentemente observado na expansão de modelos de linguagem.

Esses resultados indicam que o modelo Qwen2.5-3B é particularmente adequado para cenários com restrições de custo computacional ou operando em ambientes de borda. Por outro lado, para aplicações críticas, nas quais a precisão em consultas complexas é imprescindível, o uso de modelos maiores como o Qwen2.5-32B pode ser justificado, mesmo frente ao custo significativamente mais elevado.

Considerando o custo por requisição individual na tarefa de *Text-to-SQL*, com base no *dataset* CNPJ tem-se em uma média de 2698 *tokens* para a etapa de *schema-linking*, 704 *tokens* para geração da consulta SQL (totalizando 3402 *tokens*). A partir disso, estimamos os valores apresentados na Tabela 7. Observa-se que os modelos Qwen2.5-0.5B, 1.5B e 3B apresentam um custo por requisição extremamente baixo (cerca de 0,00034 USD), o que reforça sua atratividade para aplicações de larga escala ou operando sob restrições orçamentárias. Em contrapartida, os modelos maiores, como Qwen2.5-14B e Qwen2.5-32B, alcançam um custo por requisição próximo a 0,003 USD, o que pode se tornar significativo em cenários de alto volume de requisições.

Tabela 7. Estimativas de custo por milhão de *tokens* e por requisição (3402 *tokens* para modelos Qwen2.5, com base nos preços da Together.ai (maio/2025)

Modelo Qwen2.5	Custo (USD/1M *tokens*)	Custo por requisição (USD)
Qwen2.5-0.5B	\$0.10	\$0.00034
Qwen2.5-1.5B	\$0.10	\$0.00034
Qwen2.5-3B	\$0.10	\$0.00034
Qwen2.5-7B	\$0.30	\$0.00102
Qwen2.5-14B	\$0.80	\$0.00272
Qwen2.5-32B	\$0.80	\$0.00272

Caso se opte por realizar o *deployment* utilizando instâncias da AWS com aceleração por GPU, a análise de custo-desempenho deve levar em conta não apenas a acurácia e o custo por *token*, mas também os requisitos de memória e infraestrutura. Nos testes realizados com entradas de 2201 *tokens* e saídas de 50 *tokens*, observou-se que o Qwen2.5-0.5B consome aproximadamente 0,56GB de VRAM, o Qwen2.5-1.5B 1,37GB, o Qwen2.5-3B 2,36GB, o Qwen2.5-7B 6,07GB, o Qwen2.5-14B 10,86GB e o Qwen2.5-32B 20,95GB. Esses valores são determinantes na escolha das instâncias de inferência.

Instâncias como a `g4dn.xlarge`, com 16GB de VRAM e custo de USD 0,316/h, são capazes de suportar modelos de até 14B parâmetros, como o Qwen2.5-14B, desde que a utilização seja cuidadosamente gerenciada. Modelos menores, como os de 0.5B, 1.5B

e 3B, consomem significativamente menos memória e permitem a execução de múltiplas réplicas em uma única GPU, maximizando a utilização do recurso e reduzindo a latência por requisição. Para modelos como o Qwen2.5-32B, cujo consumo excede os 16GB, seria necessária a adoção de instâncias como a `g5.xlarge`, com 24GB de memória GPU, ao custo de USD 1,006/h.

É importante observar que, em cenários de produção, outros fatores críticos devem ser considerados além do uso de memória de GPU. O consumo de CPU, uso de memória RAM, tempo de carregamento do modelo, latência de inferência e capacidade de paralelismo têm impacto direto na escalabilidade e na experiência do usuário final. Nesses contextos, modelos menores oferecem vantagens operacionais substanciais: além de exigirem menos recursos por instância, eles possibilitam maior densidade de implantação, menor tempo de resposta e menor custo de manutenção.

Assim, mesmo sob a perspectiva de *self-hosting*, o Qwen2.5-3B permanece como uma solução vantajosa, equilibrando requisitos computacionais acessíveis, capacidade de escalabilidade horizontal e desempenho competitivo na tarefa de *Text-to-SQL*.

6. Conclusão

Este trabalho investigou a relação entre o tamanho de modelos da família Qwen2.5 e seu desempenho na tarefa de *Text-to-SQL*, com especial atenção ao custo de inferência e à eficácia de estratégias auxiliares, como o uso de exemplos de valores, *fine-tuning* supervisionado e técnicas de *schema-linking*. Os resultados indicaram que essas estratégias, sobretudo as que envolvem filtragem do esquema do banco de dados, são eficazes para melhorar o desempenho, especialmente em modelos menores, nos quais intervenções mais sofisticadas tornam-se ainda mais relevantes.

A análise também demonstrou que o modelo Qwen2.5-3B representa o melhor ponto de equilíbrio entre custo computacional e desempenho, atingindo bons resultados mesmo em consultas de maior complexidade, com baixo custo por requisição. Modelos maiores, como o Qwen2.5-14B e o Qwen2.5-32B, apresentaram desempenho superior, sobretudo em tarefas de maior dificuldade, mas o uso deles implica um custo significativamente mais elevado.

Do ponto de vista da implantação, os modelos menores oferecem vantagens operacionais importantes, como menor consumo de memória, maior facilidade de escalabilidade horizontal e menor custo de manutenção, o que os torna especialmente adequados para aplicações em larga escala ou com restrições orçamentárias. Por outro lado, aplicações críticas e sensíveis à precisão podem justificar o uso de modelos maiores, mesmo diante do aumento nos custos.

Como trabalhos futuros, propõe-se explorar estratégias adaptativas que selecionem dinamicamente o modelo ou a abordagem mais adequada com base na complexidade da consulta, visando otimizar a relação entre custo e desempenho em tempo de execução. Além disso, pretende-se avaliar o impacto das técnicas propostas em outros modelos open-source de diferentes tamanhos, bem como testar seu desempenho em bancos de dados em português, com perguntas formuladas em língua portuguesa, inspirado na iniciativa do trabalho [José and Cozman 2021]. Nesse contexto, destaca-se a intenção de explorar o estudo de caso do CNPJ, como forma de ampliar a aplicabilidade dos resultados em cenários reais de uso no contexto brasileiro.

Referências

[Dettmers et al. 2023] Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). QLoRA: Efficient Finetuning of Quantized LLMs. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023*.

[Dong et al. 2023] Dong, X., Zhang, C., Ge, Y., Mao, Y., Gao, Y., Chen, L., Lin, J., and Lou, D. (2023). C3: Zero-shot Text-to-SQL with ChatGPT. *ArXiv*, abs/2307.07306.

[Gao et al. 2024] Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., and Zhou, J. (2024). Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. *Proc. VLDB Endow.*

[Hong et al. 2024] Hong, Z., Yuan, Z., Zhang, Q., Chen, H., Dong, J., Huang, F., and Huang, X. (2024). Next-Generation Database Interfaces: A Survey of LLM-based Text-to-SQL. *ArXiv*, abs/2406.08426.

[Hu et al. 2022] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). LoRA: Low-Rank Adaptation of Large Language Models. In *The Tenth International Conference on Learning Representations, ICLR 2022*.

[José and Cozman 2021] José, M. and Cozman, F. (2021). mRAT-SQL+GAP: A Portuguese Text-to-SQL Transformer. In *Anais da X Brazilian Conference on Intelligent Systems, BRACIS 2021*.

[Li et al. 2024a] Li, H., Zhang, J., Liu, H., Fan, J., Zhang, X., Zhu, J., Wei, R., Pan, H., Li, C., and Chen, H. (2024a). CodeS: Towards Building Open-source Language Models for Text-to-SQL. *Proc. ACM Manag. Data*.

[Li et al. 2024b] Li, Z., Wang, X., Zhao, J., Yang, S., Du, G., Hu, X., Zhang, B., Ye, Y., Li, Z., Zhao, R., and Mao, H. (2024b). PET-SQL: A Prompt-Enhanced Two-Round Refinement of Text-to-SQL with Cross-consistency. *ArXiv*, abs/2403.09732.

[Liu et al. 2024] Liu, X., Shen, S., Li, B., Ma, P., Jiang, R., Zhang, Y., Fan, J., Li, G., Tang, N., and Luo, Y. (2024). A Survey of NL2SQL with Large Language Models: Where are we, and where are we going? *ArXiv*, abs/2408.05109.

[Miranda and Campelo 2024] Miranda, B. and Campelo, C. E. C. (2024). How effective is an LLM-based Data Analysis Automation Tool? A Case Study with ChatGPT's Data Analyst. In *Anais do XXXIX Simpósio Brasileiro de Bancos de Dados, SBBD 2024*.

[Mohammadjafari et al. 2024] Mohammadjafari, A., Maida, A. S., and Gottumukkala, R. (2024). From Natural Language to SQL: Review of LLM-based Text-to-SQL Systems. *ArXiv*, abs/2410.01066.

[Oliveira et al. 2024] Oliveira, A., Nascimento, E., Pinheiro, J. a., Avila, C. V. S., Coelho, G., Feijó, L., Izquierdo, Y., García, G., Leme, L. A. P. P., Lemos, M., and Casanova, M. A. (2024). Small, Medium, and Large Language Models for Text-to-SQL. In *Conceptual Modeling: 43rd International Conference, ER 2024*.

[Pourreza and Rafiei 2024] Pourreza, M. and Rafiei, D. (2024). DTS-SQL: Decomposed Text-to-SQL with Small Large Language Models. In *Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2024*.

[Volvovsky et al. 2024] Volvovsky, S., Marcassa, M., and Panbiharwala, M. (2024). DFIN-SQL: Integrating Focused Schema with DIN-SQL for Superior Accuracy in Large-Scale Databases. *ArXiv*, abs/2403.00872.

[Wang et al. 2025] Wang, B., Ren, C., Yang, J., Liang, X., Bai, J., Chai, L., Yan, Z., Zhang, Q., Yin, D., Sun, X., and Li, Z. (2025). MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025*.

[Xu et al. 2023] Xu, L., Xie, H., Qin, S.-Z. J., Tao, X., and Wang, F. L. (2023). Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment. *ArXiv*, abs/2312.12148.

[Yang et al. 2024] Yang, S., Su, Q., Li, Z., Li, Z., Mao, H., Liu, C., and Zhao, R. (2024). SQL-to-Schema Enhances Schema Linking in Text-to-SQL. In *Database and Expert Systems Applications - 35th International Conference, DEXA 2024*.

[Yu et al. 2018] Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., and Radev, D. (2018). Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMLP 2018*.

[Zhang et al. 2024] Zhang, T., Chen, C., Liao, C., Wang, J., Zhao, X., Yu, H., Wang, J., Li, J., and Shi, W. (2024). SQLfuse: Enhancing Text-to-SQL Performance through Comprehensive LLM Synergy. *ArXiv*, abs/2407.14568.

[Zhong et al. 2020] Zhong, R., Yu, T., and Klein, D. (2020). Semantic Evaluation for Text-to-SQL with Distilled Test Suites. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*.