DTS-SQL: Decomposed Text-to-SQL with Small Large Language Models

Mohammadreza Pourreza
University of Alberta
pourreza@ualberta.ca

Davood Rafiei
University of Alberta
drafiei@uablerta.ca

Abstract

Leading models for the text-to-SQL task heavily rely on proprietary Large Language Models (LLMs), posing concerns over data privacy. Closing the performance gap between small open-source models and large proprietary models is crucial to mitigate this reliance. To this end, we introduce a novel two-stage fine-tuning approach that decomposes the task into two simpler tasks. Through comprehensive evaluation on three large cross-domain datasets and two small LLMs, we show that this approach improves execution accuracy by 3 to 7 percent, effectively aligning the performance of open-source models with their proprietary counterparts. Our proposed method has achieved 60.31% execution accuracy on BIRD hold-out test set, which is the highest performance among methods using 7B parameter models.

1 Introduction

Natural language interfaces for databases allow users to derive insights from structured databases using natural language instead of complex SQL queries. Leading open-source methods (Pourreza and Rafiei, 2024; Gao et al., 2023; Wang et al., 2023) for this task heavily depend on proprietary Large language models (LLMs) like GPT-4 and GPT-3.5-turbo, which have demonstrated superior performance in Text-to-SQL benchmarks (Yu et al., 2018; Li et al., 2023c; Gan et al., 2021). However, this reliance on large proprietary models has privacy and cost implications. For instance, many large enterprises cannot share their customer data with the model-providing companies due to privacy considerations. Additionally, cost is a factor, especially for small businesses, in adopting these models.

Recent attempts to utilize open-source LLMs (Gao et al., 2023) and fine-tune them using question-SQL query pairs have fallen short of the zero-shot performance of GPT-3.5-turbo. Table 1 presents a performance comparison of the fine-tuned open-source LLMs on the Spider development set, contrasting with methods that employ GPT-4's prompting techniques. *Our hypothesis is that the task of text-to-SQL is too complex to be mastered in a single stage using small LLMs.* We aim to address this disparity by introducing a novel two-step decomposed fine-tuning method, employing two smaller LLMs. This approach, utilizing a model with a parameter size of 7 billion, achieves a performance comparable to methods using GPT-4 with few-shot learning and well-designed prompts.

Model	EX	EM
Fine-tuning methods		
Llama2 7B (Gao et al., 2023)	66.7	63.9
Llama2 13B (Gao et al., 2023)	67.0	62.7
Prompting methods		
DAIL-SQL + GPT4 (Gao et al., 2023)	84.4	74.4
DIN-SQL + GPT4 (Pourreza and Rafiei, 2024)	74.2	60.1

Table 1: Performance comparison of the prompting methods and finetuning methods on Spider validation dataset. EX stands for Execution accuracy and EM stands for Exact Match accuracy.

We evaluate the performance of our proposed method using three Text-to-SQL benchmarks: Spider (Yu et al., 2018), BIRD (Li et al., 2023c), and Spider-SYN (Gan et al., 2021), along with two 7B LLMs: DeepSeek DeepSeek-AI (2024) and Mistral Jiang et al. (2023). Our approach demonstrates a performance improvement of approximately 3 to 7 percent in execution accuracy compared to the conventional single-step fine-tuning method employed in previous studies (Gao et al., 2023). This consistent performance gain across these datasets highlights the generalizability of our method. Moreover, our fine-tuning strategy, utilizing a 7 billion parameter LLM, surpasses all previous open-source methods on the Spider development set and achieves comparable results to the state-of-the-art open-source methods using GPT-4 (Pourreza and Rafiei, 2024; Gao et al., 2023) on the Spider test set. On the hold-out BIRD test set, our method with DeepSeek 7B surpasses all of the methods using 7B parameter models (Li et al., 2024) and ranked second on the leaderboard among methods with publicly available papers, with 60.31% execution accuracy. All the necessary code to replicate the results provided in our GitHub repository [1].

1.1 Related Works

Early efforts by the database community, such as custom templates, marked initial advancements but required substantial manual effort (Zelle and Mooney, 1996). Recently, text-to-SQL methodologies have increasingly incorporated transformer-based models, particularly sequence-to-sequence architectures (Vaswani et al., 2017; Sutskever et al., 2014).

Initial sequence-to-sequence models, such as IRNet, utilized bidirectional LSTM architecture and self-attention to encode database schema representation (Guo et al., 2019). Advanced models like RAT-SQL (Wang et al., 2019) and RASAT (Qi et al., 2022) used relation-aware self-attention mechanisms. Models like SADGA (Cai et al., 2021) and LGESQL (Cao et al., 2021) adopted graph neural networks to represent relational structures between database schema and queries.

The field has also benefited from recent methodological innovations in large language models (LLMs). Early approaches leveraged the zero-shot in-context learning capabilities of LLMs for SQL generation (Rajkumar et al., 2022). Subsequent models like DIN-SQL (Pourreza and Rafiei, 2024), DAIL-SQL (Gao et al., 2023), MAC-SQL (Wang et al., 2023), and C3 (Dong et al., 2023) have enhanced performance through task decomposition and techniques like Chains of Thought (CoT) (Wei et al., 2022), and self-consistency (Wang et al., 2022). Concurrent with our work, Blar-SQL (Domínguez et al., 2024) proposed a fine-tuning approach for improving the performance of smaller LLMs. However, their method showed significantly lower performance compared to ours on the BIRD benchmark, with roughly 9% percent gap on BIRD development set.

[1]`https://anonymous.4open.science/r/DTS-SQL-2A42`

2 Methodology

A notable development in LLMs is their post-pretraining refinement, which enhances their alignment with preferred behaviors, as documented by Mishra et al. (2021); Victor et al. (2022); Thoppilan et al. (2022). Common methods of alignment include Supervised Fine-Tuning (SFT) using human demonstrations, as reported by Ouyang et al. (2022); Tunstall et al. (2023) and Reinforcement Learning from Human Feedback (RLHF), as detailed by Christiano et al. (2017); Ziegler et al. (2019); Stiennon et al. (2020); Bai et al. (2022).

The absence of extensive datasets containing either human or AI feedback (Lee et al., 2023) has led to a predominant focus on supervised fine-tuning in the text-to-SQL field. This approach necessitates a collection of specific instructions or prompts along with their corresponding outputs or responses. In the following section, we will delve into the established methods of supervised fine-tuning for LLMs within the Text-to-SQL context. Subsequently, we introduce our novel two-step fine-tuning approach, designed to enhance the performance of models in the Text-to-SQL domain.

2.1 Supervised fine-tuning for Text-to-SQL

In this section, we explore the supervised fine-tuning process for Text-to-SQL tasks, as practiced in the open-source community (Gao et al., 2023). Given a set of databases D_i comprising pairs of questions q_i and corresponding SQL queries s_i, the goal is to fine-tune a large language model M using a set of training data $T = \{(q_i, s_i, D_i)\}$, where q_i and s_i represent the natural language question and its associated SQL query on database D_i. The objective of supervised fine-tuning is to minimize the empirical loss defined as:

$$\min_{M^*} \frac{1}{|T|} \sum_{i=1}^{|T|} \mathcal{L}(M^*(\sigma_f(q_i, D_i)), s_i) \tag{1}$$

where $\mathcal{L}$ is the next token prediction loss function used to measure the difference between the SQL queries generated by the model and the actual, correct (ground truth) queries. The function σ_f determines the formatting of the question, the database schema, and the SQL queries. A key challenge during inference is that we do not know in advance among all of the tables inside the database which tables are relevant to a given question for generating accurate SQL queries. Therefore, a common approach in fine-tuning involves including the all of the tables within the prompts together with the question and SQL pairs. This method serves a dual purpose: teaching the model to generate the correct SQL query and to identify the relevant tables from among all the provided tables. This approach of training for two objectives simultaneously complicates the SQL generation task for LLMs. Each task – generating SQL queries and correctly linking to the relevant schema – demands its own reasoning process. A significant proportion of errors in large language models can be attributed to incorrect schema linking, highlighting this as a major challenge in the field (Pourreza and Rafiei, 2024; Dong et al., 2023).

2.2 Decomposed Supervised Fine-tuning

We propose a two-stage fine-tuning process, which separates schema linking and SQL generation, aiming to enhance overall performance.

2.2.1 Schema-linking Fine-tuning

Schema linking involves identifying the pertinent columns and tables in a database in response to natural language queries. It has been demonstrated to enhance cross-domain generalizability (Lei et al., 2020) and has been a part of the pipeline for both early seq-to-seq models (Cao et al., 2021; Guo et al., 2019; Xu et al., 2021) and recent in-context learning methods using LLMs (Pourreza and Rafiei, 2024; Wang et al., 2023). However, it has not been treated as a separate module for fine-tuning LLMs. In this work, we treat schema linking as a distinct task and explicitly fine-tune LLMs to identify relevant tables and columns when presented with a natural language query. Given a training dataset $T = \{(q_i, s_i, D_i)\}$, we extract all of the columns and tables used in the SQL queries and create a new dataset of $T = \{(q_i, T_i, C_i, D_i)\}$ where T_i and C_i represent lists of tables and columns used in the SQL query s_i. The primary objective during supervised fine-tuning for schema linking is to minimize the empirical loss, as defined by the following equation:

$$\min_{M^*} \frac{1}{|T|} \sum_{i=1}^{|T|} \mathcal{L}(M^*(\sigma_s(q_i, D_i)), C_i, T_i) \tag{2}$$

Here, $\mathcal{L}$ represents the next token prediction loss, comparing the predicted column and table names with the actual ground truth names. Since our objective is to predict the set of relevant tables and columns, and order does not matter in set prediction but can affect the next token prediction loss, we always sort the schema in the prompt. By doing so, we ask the model to predict the schema in alphabetically sorted order to incorporate order in the prediction. Additionally, since the number of tokens required to include all columns and tables can exceed the context window size of smaller LLMs, we first extract the ground truth tables. Then, we sort the remaining tables based on their embedding similarity to the question's embedding. We continue adding tables in order of similarity until we reach the context window limit or there are no more tables to include. To avoid introducing any order bias, we shuffle the tables in the schema at the final step.

2.2.2 SQL Generation Fine-tuning

After identifying the appropriate tables for SQL generation, the next step is to utilize a model that constructs the SQL query based on the question and the schema of the correct tables. Since we have already identified the potentially correct tables using the schema-linking module, there is no need to include all tables in the input for the SQL generation model. In contrast to previous approaches for fine-tuning LLMs, we extract the relevant tables from the training dataset $T = \{(q_i, s_i, D_i)\}$ corresponding to the ground truth SQL queries. We then fine-tune the LLM while minimizing the following loss function:

$$\min_{M^*} \frac{1}{|T|} \sum_{i=1}^{|T|} \mathcal{L}(M^*(\sigma_g(q_i, T_i)), s_i) \tag{3}$$

The loss function is same as the loss function defined in Section 2.1. This decomposition of the Text-to-SQL training process allows LLMs to be trained with a singular objective. By segregating the schema-linking and SQL query generation tasks, we improve the training process, enabling more focused and effective fine-tuning.

3 Experiments

3.1 Models

Our methodology was evaluated using two recent LLMs from distinct architectures, namely Mistral 7B (Jiang et al., 2023) and DeepSeek 7B (DeepSeek-AI, 2024). Mistral 7B, not specifically pretrained for code generation, surpasses

Model	EX	EM
DAIL-SQL + GPT-4 (Gao et al., 2023)	86.6	-
DIN-SQL + GPT-4 (Pourreza and Rafiei, 2024)	85.3	60
DTS-SQL + DeepSeek 7B Ours	84.4	73.7
C3 + ChatGPT + Zero-Shot (Dong et al., 2023)	82.3	-
RESDSQL-3B + NatSQL (Li et al., 2023a)	79.9	72
DTS-SQL + Mistral Ours	77.1	69.3
Graphix-3B + PICARD (Li et al., 2023b)	-	74

Table 2: The comparison of different methods on test set of Spider.

Model	EX	EM
Instruction tuning methods		
DTS-SQL + Mistral 7B (our)	78.6	73.3
DTS-SQL + DeepSeek 7B (our)	**85.5**	**79.1**
Llama2 7B (Gao et al., 2023)	66.7	63.9
Llama2 13B (Gao et al., 2023)	67.0	62.7
Prompting methods		
DIN-SQL + GPT4 (Pourreza and Rafiei, 2024)	74.2	60.1
DIN-SQL + CodeX (Pourreza and Rafiei, 2024)	69.9	57.2
DAIL-SQL + GPT4 (Gao et al., 2023)	84.4	74.4
C3 + GPT-3.5 (Dong et al., 2023)	81.8	-

Table 3: Performance of different methods with LLMs on the dev set of Spider.

many counterparts in its scale category (Jiang et al., 2023). Details about the hyperparameters are included in Appendix A.4.

3.2 Datasets

We conducted our evaluation using three cross-domain, challenging Text-to-SQL datasets: (1) Spider, introduced by Yu et al. (2018), includes 160 schemas allocated for training and development, while the remaining 40 are set aside for testing purposes. (2) Spider-Syn (Gan et al., 2021) modifies the Spider dataset by replacing schema-related words with synonyms and removing explicit mentions of schema links in the questions. (3) BIRD (Li et al., 2023c) is a pioneering, cross-domain dataset that examines the impact of extensive database contents on text-to-SQL parsing with over 12,751 unique question-SQL pairs and 95 databases. Details about the metric used for evaluation is provided in Appendix A.3. For the Spider results, we trained the models on the official Spider training set, and for the BIRD results, we fine-tuned the models on the BIRD training set.

3.3 Results

3.3.1 Spider test set

As depicted in Table 2, our method employing DeepSeek 7B, when tested on the Spider test dataset, achieves results comparable to state-of-the-art open-source methods in terms of execution accuracy and exact set match accuracy.

3.3.2 Spider dev set

In Table 3, we offer a detailed comparison between our method and various other baseline approaches. For the baselines, we selected diverse methods from different families of approaches that are using LLMs and are available as open source. Our two-stage decomposed approach with DeepSeek 7B attained state-of-the-art performance on the Spider development set, surpassing all previous methods that utilized prompting techniques and fine-tuning. Additionally, the results of our two-stage method on Spider-SYN dataset is provided in Table 7 in appendix.

To validate that the performance gain is stemming from the decomposition proposed in this work or from the base LLMs, we also compared our two-stage method using the same models with vanilla fine-tuning. In Table 4, we showcase the results of our two-stage fine-tuning method on the development set of Spider. The performance is compared against two distinct scenarios: firstly, a one-stage scenario where the model is fine-tuned on all tables without employing our two-stage approach, and secondly, a perfect schema linking scenario where we provide the ground truth tables to our fine-tuned SQL generators. This latter scenario is denoted as the 'Upper Bound' in the table. Our two-stage model's performance is measured by initially using our fine-tuned schema linker model to identify potentially relevant tables, which are then provided as context to the SQL generator model.

3.4 BIRD results

To further validate the robustness of our proposed method, we also evaluated the performance of our proposed method on BIRD benchmark test and development sets. As shown in Table 5, our proposed method with DeepSeek 7B achieved the second highest performance among all published works

Model	Tuning	EX	EM
Mistral 7B	FT Tuning	71.9	70.9
Mistral 7B	DTS-SQL	78.6	73.3
Mistral 7B	Upper bound	86.6	80.7
DeepSeek 7B	FT Tuning	82.1	69.0
DeepSeek 7B	DTS-SQL	85.5	79.1
DeepSeek 7B	Upper bound	90.3	84.2

Table 4: Performance of the LLMs with different tuning methods on Spider development set. FT stands for Full tables finetuning, Upper bound performance is the performance which we can achieve with a perfect schema linking.

Model	Test EX	Dev EC
SFT CodeS-15B (Li et al., 2024)	60.37	58.47
DTS-SQL + DeepSeek (Ours)	**60.31**	**55.8**
MAC-SQL + GPT-4 (Wang et al., 2023)	59.59	57.56
SFT CodeS-7B (Li et al., 2024)	59.25	57.17
DAIL-SQL + GPT-4 (Gao et al., 2023)	57.41	54.76
DIN-SQL + GPT-4 (Pourreza and Rafiei, 2024)	55.90	50.72
Blar-SQL (Domínguez et al., 2024)	-	46.68

Table 5: The comparison of different methods on test set and development set of the BIRD benchmark.

on the test set, which shows the effectiveness of our method to achieve comparable results with proprietary LLMs or even surpass them with a small LLM.

3.4.1 Schema-linking Performance

As discussed in Section 2, our approach employs two LLMs: one for schema linking and another for SQL query generation. The schema-linking model plays a pivotal role in our pipeline, as inaccuracies in table detection could hinder the SQL generator's ability to formulate the correct SQL queries. We fine-tuned two models, based on the Deepseek and Mistral models, for schema linking. Evaluation metrics, including exact set match, precision, and recall, were used to assess their performance. Detailed information about these models on two distinct datasets can be found in Table 6.

Model	Dataset	EX	PR	RE
DeepSeek	Spider	93.1	98.4	97.7
Mistral	Spider	91.1	97.5	97.8
DeepSeek	Spider-SYN	87.6	94.6	94.7
Mistral	Spider-SYN	85.3	91.2	90.5

Table 6: Performance of the schema-linker model on Spider and Spider-SYN dev sets. PR stands for Precision, RE is recall, and EX is exact set match accuracy.

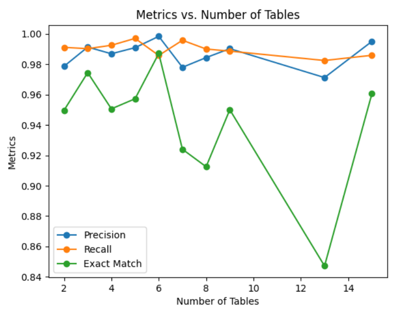

Figure 1: Precision, recall, and exeact set match performance of the schema-linking model on different number of tables

To further investigate the relationship between schema-linking performance and the effect of schema size, we conducted an analysis of precision, recall, and exact set match accuracy for the DeepSeek schema-linking model on the Spider test set with varying numbers of tables. The results, shown in Figure 1, indicate that exact set match accuracy generally decreases as the number of tables increases. However, the schema-linking model consistently maintains precision and recall values above 0.96, even with more than 14 tables.

4 Discussion

While our two-step approach has achieved comparable performance to larger models like GPT-4 on three large cross-domain datasets, there is still significant room for improvement, particularly for the schema-linking models. Currently, our schema-linking models achieve roughly 90% exact set match accuracy. However, as noted in Table 4, the substantial gap between the upper bound performance of the SQL generator and that of DTS-SQL calls for further research into the schema-linking. .

5 Conclusion

Before our research, small open-source models lagged behind large proprietary models in performance on the text-to-SQL task. Our two-stage fine-tuning approach breaks down the task into two simpler components, enabling small open-source models to rival larger ones. Subsequent efforts could focus on enhancing the performance of these stages and exploring improved methods for transferring the output of one stage to the next.

Limitations

This paper has placed its primary emphasis on enhancing the performance of both stages of fine-tuning small large language models for Text-to-SQL task. However, there remains scope for further investigation and comparison of various techniques for schema-linking. Exploring approaches like retrieval methods or in-context learning when applied in conjunction with larger models such as GPT-4 for the schema-linking task could yield valuable insights into identifying the most effective methodologies for schema-linking.

Acknowledgements

We gratefully acknowledge the financial support provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), which made this research possible.

Ethics Statement

In this paper, we place a strong emphasis on the significance of ethical considerations in every aspect of our research, from its inception to its presentation. We wholeheartedly commit to adhering to the ACL Ethics Policy and upholding ethical principles and guidelines throughout our research journey.

We have taken proactive measures to minimize any potential biases or discriminatory elements in our research design, data selection, and interpretation of results. Our dedication to transparency, precision, and fairness in reporting our findings is unwavering, and we have duly acknowledged and cited the work of others to give proper credit.

By incorporating this ethics statement, we aim to underscore our unwavering commitment to conducting research with integrity, respecting ethical principles, and contributing responsibly to the advancement of knowledge in our field.

References

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Ruichu Cai, Jinjie Yuan, Boyan Xu, and Zhifeng Hao. 2021. Sadga: Structure-aware dual graph aggregation network for text-to-sql. *Advances in Neural Information Processing Systems*, 34:7664–7676.

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. Lgesql: line graph enhanced text-to-sql model with mixed local and non-local relations. *arXiv preprint arXiv:2106.01093*.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Tri Dao. 2023. FlashAttention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*.

DeepSeek-AI. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

José Manuel Domínguez, Benjamín Errázuriz, and Patricio Daher. 2024. Blar-sql: Faster, stronger, smaller nl2sql. *arXiv preprint arXiv:2401.02997*.

Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, Jinshu Lin, Dongfang Lou, et al. 2023. C3: Zero-shot text-to-sql with chatgpt. *arXiv preprint arXiv:2307.07306*.

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. 2021. Towards robustness of text-to-SQL models against synonym substitution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2505–2515, Online. Association for Computational Linguistics.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *arXiv preprint arXiv:2308.15363*.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. *arXiv preprint arXiv:1905.08205*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. 2023. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*.

Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the role of schema linking in text-to-sql.

In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6943–6954.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13067–13075.

Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. 2024. Codes: Towards building open-source language models for text-to-sql. *arXiv preprint arXiv:2402.16347*.

Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, and Yongbin Li. 2023b. Graphix-t5: Mixing pretrained transformers with graph-aware layers for text-to-sql parsing. *arXiv preprint arXiv:2301.07507*.

Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, et al. 2023c. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *arXiv preprint arXiv:2305.03111*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Mohammadreza Pourreza and Davood Rafiei. 2024. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems*, 36.

Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Yu Cheng, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. Rasat: Integrating relational structures into pretrained seq2seq model for text-to-sql. *arXiv preprint arXiv:2205.06983*.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498*.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. 2023. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Sanh Victor, Webson Albert, Raffel Colin, Bach Stephen, Sutawika Lintang, Alyafeai Zaid, Chaffin Antoine, Stiegler Arnaud, Raja Arun, Dey Manan, et al. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2019. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *arXiv preprint arXiv:1911.04942*.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Qian-Wen Zhang, Zhao Yan, and Zhoujun Li. 2023. Mac-sql: Multi-agent collaboration for text-to-sql. *arXiv preprint arXiv:2312.11242*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Kuan Xu, Yongbo Wang, Yongliang Wang, Zujie Wen, and Yang Dong. 2021. Sead: End-to-end text-to-sql generation with schema-aware denoising. *arXiv preprint arXiv:2105.07911*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

Model	Tuning	EX	EM
Mistral 7B	FT Tuning	67.0	63.9
Mistral 7B	DTS-SQL	71.1	64.6
Mistral 7B	Upper bound	81.9	74.5
DeepSeek 7B	FT Tuning	70.4	56.6
DeepSeek 7B	DTS-SQL	76.2	68.9
DeepSeek 7B	Upper bound	85.5	78.1

Table 7: Performance of the LLMs with different tuning methods on Spider-SYN dev set. FT stands for Full tables finetuning, Upper bound performance is the performance which we can achieve with a perfect schema linking.

A Appendix

A.1 Spider-SYN dataset

To assess the efficacy of our proposed method, we evaluated its performance on the development set of Spider-SYN. Although Spider-SYN possesses a distinct training set, we opted to test our fine-tuned models directly on its development set, without any additional tuning on the Spider-SYN training set. The same performance gain is observed on this dataset (see Table 7) even though the model was not directly trained in this dataset.

A.2 Error Analysis

In this section, following the exact same setting used in (Pourreza and Rafiei, 2024) for error analysis, we sampled 400 queries from the development set of Spider, and compared our two stage fine-tuning approach with the vanilla full finetuning to evaluate the effect of our proposed method. As it is illustrated in figure 3, our method consistently improve the error cases across different classes of errors with the largest improvement on Schema linking class of errors. Additionally, having a separate schema linking module also improved the error cases for other classes like queries with GROUP BY or NESTED errors, which shows the importance of schema linking to help the LLM generate the query by removing the confusing tables.

A.3 Metrics

For Spider, we used exact set match accuracy (EM) and execution accuracy (EX). EM involves comparing the components of SQL queries, such as select, where, having, group by, and order by clauses, focusing on the matching of columns and predicates without considering the order. EX determines equivalence between a model-generated query and

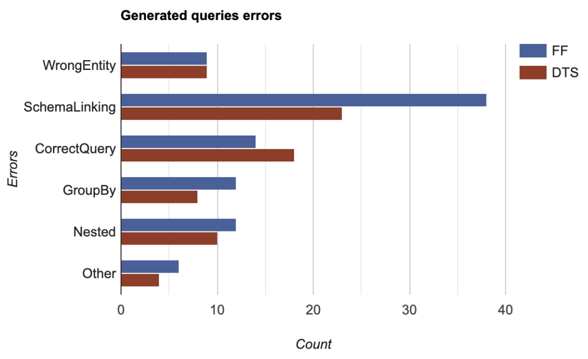

Figure 2: Error analysis on 400 queries from the Spider development set, comparing our DTS-SQL method with vanilla Full fineutning (FF) method.

a reference query if they produce identical results across various database instances. For BIRD, we used two metrics: valid efficiency score (VES), which evaluates SQL query performance by considering both accuracy and execution, and execution accuracy. We achieved a similar ranking compared to other models on VES, but due to its high variance and dependence on the computational environment, we exclude it from the current analysis.

A.4 Hyperparameters

The two LLMs, schema-linking generator and SQL generator, were trained on Nvidia Tesla A100 GPUs, employing a batch sizes of 64 and 32 with a learning rate of 1*e-5 and 5*e-5 respectively. To enhance the training efficiency, we incorporated Flash Attention techniques as detailed in (Dao et al., 2022; Dao, 2023).

A.5 Prompt

In conducting all our experiments on both models, we adhered to a standardized prompt format to ensure consistency and facilitate reliable comparisons. The chosen prompt format is well-established as effective in the Text-to-SQL domain, as demonstrated in prior research by Gao et al. (2023). In this format, we provided information about the foreign key constraints, primary keys, and column types. Furthermore, to guide the models in understanding how data is stored within the database, our prompt incorporated three sample rows, showcasing data entries.

The specific prompt used for our experiments is as follows:

```
Given the following SQL tables, your job is to generate the
Sqlite SQL query given the user's question.
Put your answer inside the ```sql and ``` tags.
{database_schema}
###
Question: {question}
```

Figure 3: The prompt used for SQL generation. The database schema is where we put the tables representations.

```
Given the following SQL tables, your job is to determine the
columns and tables that the question is referring to.
{database_schema}
###
Question: {question}
```

Figure 4: The prompt used for Schema linking. The database schema is where we put the tables representations.

```
CREATE TABLE `concert`(
        concert_ID INT PRIMARY KEY,
        concert_Name TEXT,
        Theme TEXT,
        Stadium_ID TEXT REFERENCES stadium(Stadium_ID),
        Year TEXT );
Sample rows from `concert`:
1, Auditions, Free choice, 1, 2014
2, Super bootcamp, Free choice 2, 2, 2014
3, Home Visits, Bleeding Love, 2, 2015
```

Figure 5: A sample table representation. All of the table in a database are represented as above and used in the prompts.