

The Death of Schema Linking? Text-to-SQL in the Age of Well-Reasoned Language Models

Karime Maamari¹ Fadhil Abubaker¹ Daniel Jaroslawicz¹ Amine Mhedhbi²
¹Distyl AI ²Polytechnique Montréal
 {karime,fadhil,daniel}@distyl.ai
 amine.mhedhbi@polymtl.ca

Abstract

Schema linking is a crucial step in Text-to-SQL pipelines, which translate natural language queries into SQL. The goal of schema linking is to retrieve relevant tables and columns (*signal*) while disregarding irrelevant ones (*noise*). However, imperfect schema linking can often exclude essential columns needed for accurate query generation. In this work, we revisit the need for schema linking when using the latest generation of large language models (LLMs). We find empirically that newer models are adept at identifying relevant schema elements during generation, without the need for explicit schema linking. This allows Text-to-SQL pipelines to bypass schema linking entirely and instead pass the full database schema to the LLM, eliminating the risk of excluding necessary information. Furthermore, as alternatives to schema linking, we propose techniques that improve Text-to-SQL accuracy without compromising on essential schema information. Our approach achieves 71.83% execution accuracy on the BIRD benchmark, ranking first at the time of submission.

1 Introduction

We address the task of Text-to-SQL: generating a database-executable SQL query given a natural language inquiry (Androustopoulos et al., 1995; Quamar et al., 2022). Text-to-SQL is of vital importance as it democratizes data access by allowing for natural language as the querying interface. The advent of large language models (LLMs) has significantly advanced Text-to-SQL by simplifying the translation of natural language into SQL.

LLM-based Text-to-SQL approaches typically follow a multi-stage generation pipeline, as shown in Fig. 1 (Hong et al., 2024; Li et al., 2024a; Liu et al., 2024; Zhang et al., 2024). The pipeline begins with a retrieval stage to collect contextual knowledge (non-parametric) such as the definition of technical terms and database schema elements. This is followed by a generation stage, where a candidate SQL query is produced using an LLM. Finally, in the correction stage, the pipeline regenerates the SQL if needed based on any errors during execution.

Selecting relevant elements of the database schema — known as *schema linking* — is an important aspect of Text-to-SQL. Identifying relevant tables and columns in the retrieval stage provides the necessary context for the LLM to produce correct SQL in the downstream generation stage. Note that effective schema linking implies retrieving *all* the relevant database components associated with the query. Missing even a single required column generates incorrect SQL that when executed produces errors. Thus, it is vital to ensure that all essential columns are extracted in the retrieval stage to maximize the *signal* passed to the LLM. However, this does not mean that the retrieval stage should be overly inclusive. Research has shown that *noise*, i.e., the number of irrelevant columns passed to the LLM, can often degrade text-to-SQL accuracy. For example, (Floratou et al., 2024) showed that even when the entire database schema can fit into the context of an LLM, it is still advantageous to perform schema linking. At the same time, attempts to prune out unnecessary columns during

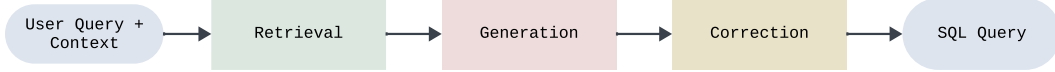


Figure 1: A typical Text-to-SQL pipeline comprised of retrieval, generation and correction stages.

schema linking may also remove some required columns. Thus, schema linking contains an inherent tension between minimizing schema noise while also preserving signal.

As LLM reasoning capabilities improve, we challenge the conventional wisdom that schema linking is necessary for accurate Text-to-SQL. We find empirically that as model reasoning improves, the benefits of noise reduction become less significant, *i.e.*, newer models are more capable of sifting through noise in their context window to identify signal compared to older ones (Laban et al., 2024; Li et al., 2024b). For these models, schema linking can often filter out necessary columns, negating any benefits from noise reduction. We present alternatives to schema linking that improve accuracy without introducing any relevant schema information loss. We propose an approach based on these empirical insights that currently ranks first in execution accuracy at 71.83% on the BIRD benchmark (Li et al., 2023).

2 Background

To frame our experiments, we first outline the key elements of the Text-to-SQL pipeline, with a focus on schema linking and its implications.

2.1 Text-to-SQL Pipeline

In current state-of-the-art solutions, Text-to-SQL pipelines are comprised of retrieval, generation, and correction stages.

The retrieval stage gathers relevant contextual information, including schema elements, domain knowledge, and example queries (Wang et al., 2024; Pourreza & Rafiei, 2023; Lee et al., 2024; Gao et al., 2023; Dong et al., 2023).

Generation often involves more than just producing a single output SQL query associated with the input context. Rather, approaches frequently augment the generation process through techniques like decomposed generation (Pourreza & Rafiei, 2023; Wang et al., 2024; Maamari & Mhedhbi, 2024) and chain-of-thought prompting (Wei et al., 2023). In addition, most approaches employ methods like self-consistency and multi-choice selection to produce multiple results, selecting the best outcome (Lee et al., 2024; Dong et al., 2023; Gao et al., 2023).

Finally, the correction stage will often employ some combination of execution-based feedback (Wang et al., 2018b; Lin et al., 2020; He et al., 2019; Lyu et al., 2020) or model-based feedback (Talaie et al., 2024; Askari et al., 2024; Wang et al., 2018a) to correct the generated query.

2.2 Schema Linking

Within the retrieval stage, schema linking leverages sophisticated prompting techniques to produce variable-length representations, hierarchically retrieve components, and iteratively process the schema (Talaie et al., 2024; Dong et al., 2023; Pourreza & Rafiei, 2023; Lee et al., 2024; Wang et al., 2024; Gao et al., 2023).

These techniques can vary in i) how they represent the schema and ii) how they perform linking on that representation. For instance, some may represent the schema in natural language, while others utilize code-like structures (Gao et al., 2023). The approach to linking can also differ across techniques, with some directly filtering the schema (Talaie et al., 2024; Lee et al., 2024; Dong et al., 2023), and others using intermediate representations to identify relevant tables and columns (Qu et al., 2024a). The choice of schema representation and linking strategy can have a significant influence on accuracy (Gao et al., 2023). This variability underscores the importance of selecting an appropriate method tailored to the specific requirements of the task, as the degree of filtering can directly impact the loss of schema information incurred during the schema linking process.



Figure 2: The various prompts used for generation and schema linking. **(Left, Green)** Table-to-Column Schema Linking (TCSL): first identifying relevant tables, followed by identifying relevant columns; GPT-4o used for all runs. **(Middle, Green)** Single-Column Schema Linking (SCSL): identifying relevance of a particular column independent of the rest of the schema; GPT-4o-Mini used for all runs. **(Red)** SQL Generation: generating a SQL query from an input query and schema; model varied according to run. **(Blue)** Sample schema and query representations used across all prompts.

Variability aside, most prior investigations around the impacts of schema linking have arrived at the same conclusion – schema linking yields meaningful gains in performance (Guo et al., 2019; Li et al., 2024a; Talaei et al., 2024). However, these explorations used prior generation models that are more sensitive to noise in the schema, where reducing the noise yielded meaningful gains in performance (Floratos et al., 2024).

3 Experimental Setup

Our experiments aim to answer two research questions: (i) What is the impact of including irrelevant schema elements (noise) and filtering relevant schema elements (signal) on SQL generation accuracy? and (ii) What is the relative impact of other techniques and stages besides schema linking on SQL generation accuracy?

Next, we cover the details of our experimental setup (datasets and models) and our methodology.

3.1 Datasets

We conducted our experiments using the the BIRD benchmark for all experiments (Li et al., 2023), which is widely regarded as the most challenging Text-to-SQL benchmark. BIRD contains queries from 95 databases spanning a wide breadth of domains, such as education and hockey, and is designed to the complexity of real-world databases. This complexity arises from its “dirty” format, where data, queries, and external knowledge may contain flaws — queries can be incorrect, schemata might be improperly described, and databases containing null values and unexpected encodings. Our evaluation set consisted of 10% of the entries from each database in the development set, as done in evaluations in the literature (Talaei et al., 2024). Our training set consisted of 500 of the 9,428 available samples in the BIRD training dataset.

3.2 Models

We used the following language models in our experiments:

ft:GPT-4o (fine-tuned)	Llama 3.1-405b	Claude 3.5 Sonnet
GPT-4o	Llama 3.1-70b	Claude 3 Opus
GPT-4o-Mini	Llama 3.1-8b	Mixtral-8x22B
GPT-4-Turbo	Deepseek Coder-V2	Gemini 1.5 Pro

All models have context windows large enough to handle the entire schema for each query in the evaluation set. We trained GPT-4o in an iterative manner, in which the model was prompted to self-reflect on what portion of the training set would be best suited to correct existing weaknesses, building upon existing techniques in self-evaluation and self-improvement (Qu et al., 2024b).

3.3 Methodology

For all experiments, we simplify the Text-to-SQL pipeline to retrieval containing only schema linking and SQL generation. The structure of the input prompt for SQL generation and schema elements organization within the prompt are shown in Fig. 2. The schema information for each column includes samples of data values: if the column is non-numeric, we include five values based on semantic similarity to the input request. For numerical columns, we include the five most frequent unique values if the size of the unique value set is less than fifteen. Otherwise, we include all unique numerical values. In all runs, the temperature was set to zero and structured output was used whenever possible. Given that not all models contain reliable structured output generation, all generations were fed through a GPT-4o-Mini identity call after the fact, in JSON mode, to handle any potential issues with output formatting.

3.4 Metrics

The BIRD benchmark has two metrics for evaluation: Execution Accuracy (EX) and Valid Efficiency Score (VES). EX measures how closely the output table of the predicted SQL query matches that of the ground truth SQL query. If they fully match, the prediction is correct. EX reports the percentage of correct predictions over all queries across multiple databases. VES assesses the efficiency of correctly predicted queries by comparing their execution speed to that of the corresponding ground truth queries. In this study, we focus on EX, as our research questions are primarily concerned with SQL generation accuracy.

4 Results

4.1 Experiment 1: Impact of Noise on EX Given Perfect Schema Linking Recall

We assess the impact of noise on SQL generation accuracy by adding irrelevant schema columns as contextual information. We run our experiments by building an oracle to retain all columns necessary for a given query after schema linking while varying the percentage of irrelevant columns. Our oracle is built using SQLGLOT, an open-source Python SQL Parser and Transpiler. If there is not a sufficient number of columns in a target database to inject the desired amount of noise (e.g., for 10 signal columns, 900 noise columns are required for a noise percentage of 99%, but a database may only have 50 columns), we supplement from other databases with additional columns not conflicting in name. This setup represents an ideal scenario for schema linking where there is perfect recall and therefore no issues in generation due to missing required columns. We refer to the accuracy obtained as the *Idealized Execution Accuracy* (IEX).

The results are shown in Fig. 3. As noise decreases, IEX increases. Notably, models with higher maximum IEX, such as Gemini 1.5 Pro, demonstrate greater resilience to noise compared to lower-performing ones like Llama 3.1-8b. This relationship is clearly depicted in the right panel of Fig. 3, where a strong negative correlation is observed between a model’s maximum IEX and its sensitivity to noise ($d(\text{IEX})/d(\text{Noise})$).

Empirical Observation: as model performance improves, the impact of noise on IEX lessens.

Approach	Noise Metric	Loss
Full Schema	94.62	0.0
Hybrid Single-Column Schema Linking	82.08 ± 0.05	9.64 ± 0.78
Single-Column Schema Linking	67.23 ± 0.24	11.23 ± 0.75
Hybrid Table-to-Column Schema Linking	19.85 ± 0.60	17.00 ± 0.92
Table-to-Column Schema Linking	9.79 ± 0.57	22.56 ± 1.34

Table 1: The noise metric (NM) and contextual Loss metrics (CLMs) associated with five schema linking approaches: Full Schema, Hybrid Single-Column Schema Linking (HySCSL), Single-Column Schema Linking (SCSL), Hybrid Table-to-Column Schema Linking (HyTCSL), and Table-to-Column Schema Linking (TCSL).

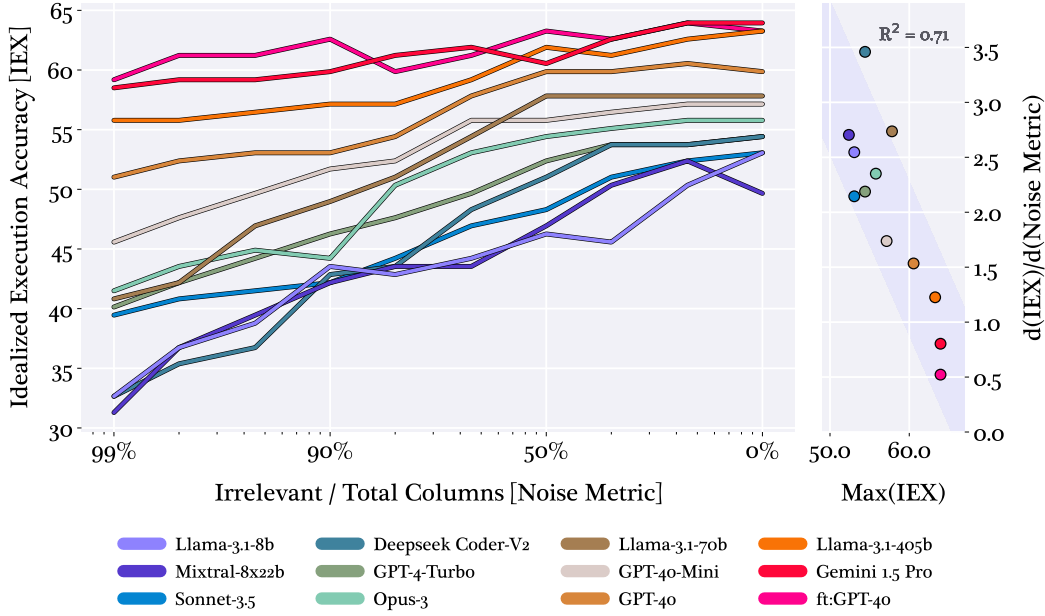


Figure 3: **(Left)** The Idealized Execution Accuracy (IEX) of different language models as the amount of noise varies, while always maintaining 100% signal. The term "idealized" is used because all signal is preserved despite the presence of noise, representing a best-case scenario. **(Right)** The relationship between model performance (maximum IEX) and its sensitivity to noise ($d(IEX)/d(\text{Noise Metric})$). As model performance increases, sensitivity to noise decreases.

4.2 Experiment 2: Impact of Noise on EX Given Actual Schema Linking Recall

In the first experiment all necessary columns for generation were provided, regardless of the noise percentage, to create an idealized scenario with complete signal retention. However, in practice, decreasing the amount of noise requires pruning irrelevant columns, which carries the risk of mistakenly pruning required ones. This second experiment is designed to assess the extent to which schema linking contributes to contextual information loss of required columns, which we refer to as loss for short.

To do so, we employ two initial schema linking approaches with varying precision on reducing noise:

Single-Column Schema Linking (SCSL): Model-determined column-wise relevance. It identifies the relevance of each column without context about other columns and tables. The schema linking system prompt is shown in Fig. 2. The output is a boolean flag per column indicating the relevance of each. This is considered a more cautious approach.

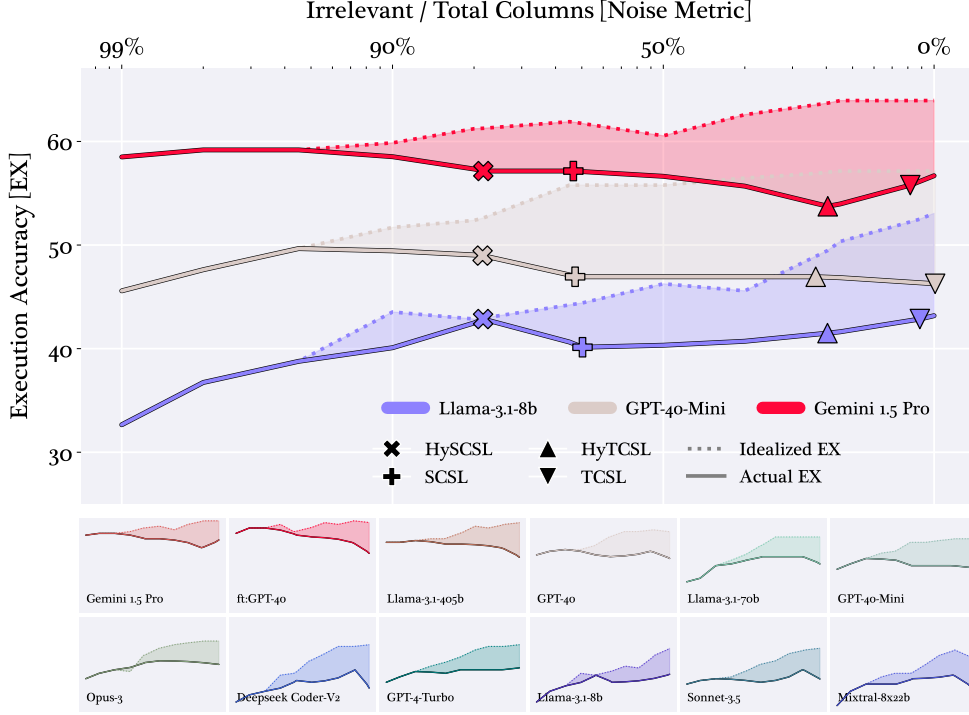


Figure 4: The idealized and actual execution accuracy (EX) of models as noise varies. The idealized EX, assuming 100% signal retention, shows performance gains as noise decreases. In practice, actual EX, determined using four schema linking approaches (TCSL, SCSL, HyTCSL, HySCSL), reveals effects of over-filtering. State-of-the-art models like Gemini 1.5 Pro show idealized gains but net losses in actual performance. Models like GPT-4o-Mini show idealized gains but near-zero actual gains. Smaller, less capable models like Llama-3.1-8b show large idealized gains and smaller net gains in actual performance. That is, while noise reduction ideally improves EX, practical over-filtering can lead to loss, ultimately degrading the performance of state-of-the-art models.

Table-then-Column Schema Linking (TCSL): Model-determined table-then-column filtering approach, as proposed by Talaei et al. (2024) and Pourreza & Rafiei (2023). The model first filters the schema to the relevant tables, then filters the columns within those tables. The schema linking system prompt is shown in Figure 2. The output is a set of relevant columns and tables. This is a more aggressive filtering approach.

For each of these approaches, we also constructed hybrids (HySCSL, HyTCSL) for a total of four approaches. HySCSL and HyTCSL perform a keyword match in addition to prompting the models to aim for higher recall. In our implementation, SCSL and HySCSL use GPT-4o-Mini and TCSL and HyTCSL use GPT-4o. Finally, we introduce two metrics to evaluate the performance of the schema linking techniques on a particular database: (i) *Noise Metric (NM)*: percentage of irrelevant schema columns for a given query after retrieval and before the generation stages; and (ii) *Contextual Loss Metric (CLM)*: percentage of queries missing required columns for SQL generation.

Table 1 reports these metrics for SCSL, TCSL, HySCSL, and HyTCSL as an average from their value on each BIRD database. It further includes these metrics for the case of no schema linking (Full Schema). The table shows that reducing NM leads to an increase in CLM. Since CLM is calculated only based on the retrieval stage, we naturally ask if there is an actual observed impact empirically due to contextual loss?

Fig. 4 shows EX across the various models as NM changes when running the Text-to-SQL pipeline end-to-end using the four schema linking techniques.

Empirical Observation: Worse performing models, such as Llama 3.1-8b, improve in performance with better schema linking. Top performing models, such as Gemini 1.5 Pro, see a net reduction

Method	Execution Accuracy [%]		
	ft:GPT-4o	Gemini 1.5 Pro	Llama 3.1-405b
Full pipeline	67.35	60.54	59.18
w/o Augmentation	64.63 (↓ 2.72)	60.54	59.86 (↑ 0.68)
w/o Selection	65.31 (↓ 2.04)	57.82 (↓ 2.72)	58.50 (↓ 0.68)
w/o Correction	65.99 (↓ 1.36)	57.14 (↓ 3.40)	55.78 (↓ 3.40)
w/ TCSL	62.58 (↓ 4.77)	55.78 (↓ 4.76)	56.46 (↓ 2.72)
w/ SCSL	55.78 (↓ 11.57)	55.10 (↓ 5.44)	54.42 (↓ 4.76)
Base model	59.18 (↓ 8.17)	57.82 (↓ 2.72)	53.74 (↓ 5.44)

Table 2: Ablation of different methods on development set execution accuracy (EX) for fine-tuned GPT-4o, Gemini 1.5 Pro, and Llama 3.1-405b. The table compares the full pipeline to variations without augmentation, selection, and correction operators. It also includes results with Table-to-Column Schema Linking (TCSL) and Single-Column Schema Linking (SCSL), along with the base model performance. Reductions in accuracy (↓) from the full pipeline are indicated in parentheses.

in EX. The majority of models however, *e.g.*, GPT-4o-Mini, are in between. Clever schema linking approaches may still yield for them some gains while standard approaches provide negligible improvements in performance.

4.3 Experiment 3: Impact of Non-Filtering Stages and Techniques

Instead of filtering through schema linking, we focus next on techniques that preserve contextual information. We assess the gains of using *augmentation* and *selection* as well as the correction stage, which are detailed as follows:

Augmentation: helps the model with reasoning. It can be broadly categorized into four categories: (i) add contextual information; (ii) reframing existing contextual information; (iii) adding information to output; and (iv) reframe existing output from generation. Some standard examples include chain-of-thought prompting (Wei et al., 2023; Kojima et al., 2023; Wang et al., 2023a), decomposed query generation (Maamari & Mhedhbi, 2024; Pourreza & Rafiei, 2023), iterative generation (Wang et al., 2024), and query reformulation (Maamari & Mhedhbi, 2024).

Selection: generates multiple responses and chooses the most appropriate one. Some examples include self-consistency (Wang et al., 2023b), three-of-thoughts (Yao et al., 2023), graph-of-thoughts (Besta et al., 2024), multi-choice selection (Lee et al., 2024).

Correction: refines generation iteratively based on feedback. Some examples include execution-based errors (Wang et al., 2018b), database administrator instructions (Talaie et al., 2024), and Reflexion (Shinn et al., 2023).

The combination of these techniques and stages generally yields gains irrespective of the use case when added to generation process used in Experiment 1. Table 4.3 shows an ablation of results that adopt them to demonstrate their effectiveness.

Empirical Observation: Each of augmentation, selection, and correction have a noticeable effect on performance. Note that even though base models such as Gemini 1.5 Pro may show comparable performance to a fine-tuned GPT-4o in SQL generation as shown in Experiment 2 (Fig. 3), they differ when evaluated within end-to-end pipelines.

4.4 Discussion

The results from our experiments reveal several key insights regarding the use of schema linking in the Text-to-SQL generation process. First, as models improve, their ability to retrieve relevant information from the schema also improves, reducing their susceptibility to noise within the schema. That is, the

benefit of filtering the schema to remove irrelevant columns becomes less significant. Second, filtering the schema carries the risk of over-filtering, where essential columns are inadvertently excluded, thus reducing accuracy. For models with lower performance, the accuracy gains from filtering the schema outweigh the losses due to over-filtering. However, for state-of-the-art models, the loss in accuracy from over-filtering surpasses the gains from reducing noise. Given these findings, we question the value of filter-based schema linking for newer generation models that have improved reasoning capabilities. For the most capable models, the risk associated with over-filtering the schema no longer warrants the associated gains in performance. Instead, the focus should shift towards enhancing signal identification rather than noise reduction.

References

- I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. Natural language interfaces to databases - an introduction, 1995. URL <https://arxiv.org/abs/cmp-lg/9503016>.
- Arian Askari, Christian Poelitz, and Xinye Tang. Magic: Generating self-correction guideline for in-context text-to-sql, 2024. URL <https://arxiv.org/abs/2406.12692>.
- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefer. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690, March 2024. ISSN 2159-5399. doi: 10.1609/aaai.v38i16.29720. URL <http://dx.doi.org/10.1609/aaai.v38i16.29720>.
- Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, lu Chen, Jinshu Lin, and Dongfang Lou. C3: Zero-shot text-to-sql with chatgpt. *CoRR*, abs/2307.07306, 2023.
- Avrilia Floratou, Fotis Psallidas, Fuheng Zhao, Shaleen Deep, Gunther Hagleither, Wangda Tan, Joyce Cahoon, Rana Alotaibi, Jordan Henkel, Abhik Singla, Alex Van Grootel, Brandon Chow, Kai Deng, Katherine Lin, Marcos Campos, K. Venkatesh Emani, Vivek Pandit, Victor Shnayder, Wenjing Wang, and Carlo Curino. Nl2sql is a solved problem... not! In *Conference on Innovative Data Systems Research*, 2024. URL <https://api.semanticscholar.org/CorpusID:266729311>.
- Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. Text-to-sql empowered by large language models: A benchmark evaluation, 2023. URL <https://arxiv.org/abs/2308.15363>.
- Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. Towards complex text-to-sql in cross-domain database with intermediate representation. *CoRR*, abs/1905.08205, 2019.
- Pengcheng He, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. X-sql: reinforce schema representation with context, 2019. URL <https://arxiv.org/abs/1908.08113>.
- Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. Next-generation database interfaces: A survey of llm-based text-to-sql, 2024. URL <https://arxiv.org/abs/2406.08426>.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *CoRR*, abs/2205.11916, 2023.
- Philippe Laban, Alexander R. Fabbri, Caiming Xiong, and Chien-Sheng Wu. Summary of a haystack: A challenge to long-context llms and rag systems, 2024. URL <https://arxiv.org/abs/2407.01370>.
- Dongjun Lee, Choongwon Park, Jaehyuk Kim, and Heesoo Park. Mcs-sql: Leveraging multiple prompts and multiple-choice selection for text-to-sql generation. *CoRR*, abs/2405.07467, 2024.
- Boyan Li, Yuyu Luo, Chengliang Chai, Guoliang Li, and Nan Tang. The dawn of natural language to sql: Are we fully ready? *ArXiv*, abs/2406.01265, 2024a. URL <https://api.semanticscholar.org/CorpusID:270214429>.

- Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls, 2023. URL <https://arxiv.org/abs/2305.03111>.
- Mo Li, Songyang Zhang, Yunxin Liu, and Kai Chen. Needlebench: Can llms do retrieval and reasoning in 1 million context window?, 2024b. URL <https://arxiv.org/abs/2407.11963>.
- Xi Victoria Lin, Richard Socher, and Caiming Xiong. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing, 2020. URL <https://arxiv.org/abs/2012.12627>.
- Xinyu Liu, Shuyu Shen, Boyan Li, Peixian Ma, Runzhi Jiang, Yuyu Luo, Yuxin Zhang, Ju Fan, Guoliang Li, and Nan Tang. A survey of nl2sql with large language models: Where are we, and where are we going?, 2024. URL <https://arxiv.org/abs/2408.05109>.
- Qin Lyu, Kaushik Chakrabarti, Shobhit Hathi, Souvik Kundu, Jianwen Zhang, and Zheng Chen. Hybrid ranking network for text-to-sql, 2020. URL <https://arxiv.org/abs/2008.04759>.
- Karime Maamari and Amine Mhedhbi. End-to-end text-to-sql generation within an analytics insight engine, 2024. URL <https://arxiv.org/abs/2406.12104>.
- Mohammadreza Pourreza and Davood Rafiei. Din-sql: Decomposed in-context learning of text-to-sql with self-correction, 2023.
- Ge Qu, Jinyang Li, Bowen Li, Bowen Qin, Nan Huo, Chenhao Ma, and Reynold Cheng. Before generation, align it! a novel and effective strategy for mitigating hallucinations in text-to-sql generation, 2024a. URL <https://arxiv.org/abs/2405.15307>.
- Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve, 2024b. URL <https://arxiv.org/abs/2407.18219>.
- Abdul Quamar, Vasilis Efthymiou, Chuan Lei, and Fatma Özcan. Natural language interfaces to data. *Found. Trends Databases*, 11(4):319–414, 2022.
- Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL <https://arxiv.org/abs/2303.11366>.
- Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. Chess: Contextual harnessing for efficient sql synthesis. *CoRR*, abs/2405.16755, 2024.
- Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Linzheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. Mac-sql: A multi-agent collaborative framework for text-to-sql. *CoRR*, abs/2312.11242, 2024.
- Chenglong Wang, Kedar Tatwawadi, Marc Brockschmidt, Po-Sen Huang, Yi Mao, Oleksandr Polozov, and Rishabh Singh. Robust text-to-sql generation with execution-guided decoding, 2018a.
- Chenglong Wang, Kedar Tatwawadi, Marc Brockschmidt, Po-Sen Huang, Yi Mao, Oleksandr Polozov, and Rishabh Singh. Robust text-to-sql generation with execution-guided decoding, 2018b. URL <https://arxiv.org/abs/1807.03100>.
- Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *CoRR*, abs/2305.04091, 2023a.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023b. URL <https://arxiv.org/abs/2203.11171>.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903, 2023.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL <https://arxiv.org/abs/2305.10601>.

Weixu Zhang, Yifei Wang, Yuanfeng Song, Victor Junqiu Wei, Yuxing Tian, Yiyang Qi, Jonathan H. Chan, Raymond Chi-Wing Wong, and Haiqin Yang. Natural language interfaces for tabular data querying and visualization: A survey, 2024. URL <https://arxiv.org/abs/2310.17894>.