Knowledge Base Construction for Knowledge-Augmented Text-to-SQL

Anonymous ACL submission

Abstract

Text-to-SQL aims to translate natural language queries into SQL statements, which is practical as it enables anyone to easily retrieve the desired information from databases. Recently, many existing approaches tackle this problem with Large Language Models (LLMs), leveraging their strong capability in understanding user queries and generating corresponding SQL code. Yet, the parametric knowledge in LLMs might be limited to covering all the diverse and domain-specific queries that require grounding in various database schemas, which makes generated SQLs less accurate oftentimes. To tackle this, we propose constructing the knowledge base for text-to-SQL, a foundational source of knowledge, from which we retrieve and generate the necessary knowledge for given queries. In particular, unlike existing approaches that either manually annotate knowledge or generate only a few pieces of knowledge for each query, our knowledge base is comprehensive, which is constructed based on a combination of all the available questions and their associated database schemas along with their relevant knowledge, and can be reused for unseen databases from different datasets and domains. We validate our approach on multiple text-to-SQL datasets, considering both the overlapping and non-overlapping database scenarios, where it outperforms relevant baselines substantially[1].

1 Introduction

Text-to-SQL aims to transform natural language queries from users into Structured Query Language (SQL) statements, to interact with and retrieve the information from databases (Zelle and Mooney, 1996; Xu et al., 2017; Yaghmazadeh et al., 2017; Cai et al., 2018), as illustrated in Figure 1 (A). This task has recently gained much attention since it allows non-experts to access and manipulate database information without needing to understand complex database languages. In the meantime, Large Language Models (LLMs) have shown impressive capabilities in processing and generating text and code, which have been further extended for text-to-SQL (Rajkumar et al., 2022; Gao et al., 2024).

Despite their huge successes, transforming user queries into SQL statements may still be challenging due to the need for specific domain knowledge and an understanding of the underlying database schemas, which poses a significant hurdle even for the most advanced LLMs to achieve high accuracy across diverse datasets (Li et al., 2023). For example, consider a scenario where the user asks for the query: "What is the WACC for Company X?". To accurately translate this into an SQL statement, the text-to-SQL model should understand the concept and calculation of Weighted Average Cost of Capital (WACC), which involves multiple factors including the cost of equity, cost of debt, and the respective proportions of each in the capital structure. In addition, the model needs to comprehend the specific schema of the financial database, where relevant data is distributed across multiple tables such as 'Equity', 'Debt', and 'Capital Structure'.

To tackle the aforementioned limitations due to the lack of the domain-specific knowledge for SQL generation, recent studies have proposed collecting and annotating explicit knowledge, which is then leveraged for SQL generation (Dou et al., 2022; Li et al., 2023). However, while these approaches substantially improve the performance of existing text-to-SQL models, they rely on extensive human annotations, which may be suboptimal (and nearly impractical) to conduct for all queries considering a diverse source of domain-specific knowledge from numerous databases. To address this issue, recent work proposes generating a few pieces of knowledge for each query based on the query itself and its relevant database schema (Hong et al., 2024) (see Figure 1 (B)). However, although this method

[1]We will release code, requiring approval after acceptance.

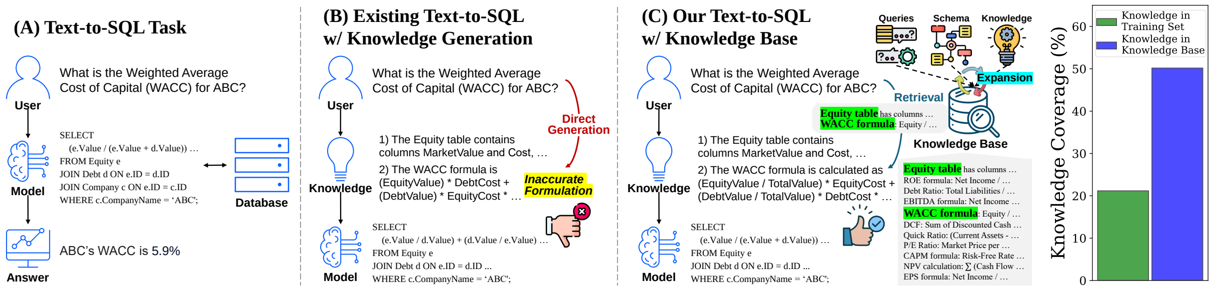

Figure 1: (A) Text-to-SQL aims to translate a user query into a SQL statement executable over a database, to access the desired information. (B) Existing Text-to-SQL with Knowledge Generation approaches first generate the knowledge relevant to the user query and then formulate the SQL statement with this generated knowledge. (C) Our Text-to-SQL with Knowledge Base Construction approach builds the repository of the knowledge and then reuses the knowledge within it across multiple queries and databases. (Right:) We observe that the knowledge in the training set of the text-to-SQL benchmark dataset (Li et al., 2023) covers 21% of the knowledge required for test-time queries, and our constructed knowledge base further covers 50% of them.

demonstrates promise in automatic knowledge generation, certain knowledge required for one query can be directly reused or provide insights for multiple queries within the same database, as shown in Figure 1 (Right). Also, this knowledge can be generalizable to other queries for different databases.

Motivated by these observations, this work proposes an automatic approach to build a knowledge base, designed to serve as a comprehensive repository of domain-specific knowledge for text-to-SQL and capable of providing knowledge for multiple queries with the same database and even across the different databases. To construct this knowledge base, we generate knowledge entries based on available samples and their associated database schemas through LLM prompting, and then compile all of them together. During this prompting process, we provide LLMs with relevant examples to contextualize and guide the generation of useful knowledge in the right format that is further grounded in the database schema. Then, once constructed, the knowledge base allows for the retrieval of relevant knowledge for the given test-time query, which is then used alongside the query to formulate the SQL statement. Note that while ideally the knowledge base would cover all possible queries, it may not always do so. Nevertheless, the existing knowledge in it could still offer valuable insights for generating the required knowledge for new queries. Thus, by leveraging similar knowledge from the knowledge base, we further prompt LLMs to produce the most suitable knowledge for the query at inference time. We call our method Knowledge-Augmented Text-to-SQL (KAT-SQL), depicted in Figure 1 (C).

We experimentally validate the proposed KAT-SQL on two different text-to-SQL scenarios, involving both the overlapping and non-overlapping databases between training and test phases, showing that the proposed knowledge base construction-based text-to-SQL approach surpasses the existing (knowledge-augmented) text-to-SQL baselines. We also assess the generalizability of our knowledge base constructed from one dataset by applying it to different datasets that lack any annotated knowledge, demonstrating that our knowledge base is versatile and can effectively improve SQL generation for even unseen databases from other datasets.

2 Related Work

LLM-Powered Text-to-SQL LLMs have shown remarkable performances across a wide range of tasks (OpenAI, 2023; Anil et al., 2023; AI@Meta, 2024), including text-to-SQL, due to their strong capability in understanding natural language and generating structured code (Rajkumar et al., 2022; Gao et al., 2024). Specifically, various studies have developed and advanced the prompting techniques for text-to-SQL, for example, using Chain-of-Thought (CoT) (Wei et al., 2022; Liu and Tan, 2023; Tai et al., 2023), investigating sophisticated prompt design strategies (Chang and Fosler-Lussier, 2023), and aggregating LLM-generated outputs from multiple prompts (Lee et al., 2024; Dong et al., 2023) akin to self-consistency (Wang et al., 2023b). In addition, another line of study proposes decomposing the text-to-SQL problem into multiple subtasks, and feeding the solutions of subtasks (from multiple models or agents) into the LLM to derive the final SQL statement (Gu et al., 2023; Pourreza and Rafiei, 2023; Wang et al., 2023a). The knowledge internalized in LLMs might however not be sufficient to handle diverse queries, which oftentimes requires grounding in the database schemas or additional domain-specific information for specialized domains, which gives rise to the need for leveraging external knowledge for text-to-SQL.

Knowledge-Augmented Text-to-SQL There are a few recent studies that propose augmenting text-to-SQL models with explicit knowledge. Specifically, Dou et al. (2022) collect formulaic knowledge (e.g., Trade Balance = Exports – Imports) available from public resources such as finance reports and store the collected knowledge into a knowledge bank with proper human-involved post-processing. The text-to-SQL model then retrieves relevant knowledge for any given query from the knowledge bank and uses it to convert the query into the SQL statement. In addition, Li et al. (2023) release a large-scale benchmark dataset for the text-to-SQL task, where each question is associated with specific knowledge that is manually annotated by humans. Manual annotation is however costly and time consuming, requiring effort and expertise on the part of domain-experts. To address this challenge, more recent work proposes automatically generating the knowledge based on the question and database schema, and utilizing this knowledge for text-to-SQL (Hong et al., 2024). In our work, instead of generating only a few pieces of knowledge for each question, we propose to construct a comprehensive knowledge base. This provides a repository of reusable knowledge that can be leveraged across multiple queries, which can be further adapted to various databases over different domains in a scalable way, in contrast to existing work.

Data Generation with LLMs The recent advent of LLMs has revolutionized the field of data generation, as they can produce vast amounts of high-quality samples without costly human annotation. Specifically, several efforts around LLM-based synthetic data generation, such as Self-Instruct (Wang et al., 2023c), Alpaca (Taori et al., 2023), Evol-Instruct (Xu et al., 2023), Orca (Mukherjee et al., 2023), and InstructLab (Sudalairaj et al., 2024), propose generating a large number of samples from LLMs by prompting them. Also, motivated by the capabilities of LLMs in generating synthetic data and memorizing factual knowledge, some other work aims to populate an encyclopedic knowledge base like Wikidata (Vrandecic and Krötzsch, 2014) with LLMs (Alivanistos et al., 2022; Nayak and Timmapathini, 2023; Veseli et al., 2023). Most of the knowledge in such encyclopedic knowledge bases is however unsuitable for text-to-SQL since it is neither relevant to formulate SQL statements from user queries nor aware of database schemas necessary for the query conversion. Thus, unlike them, our approach stands apart as the first to automatically construct a text-to-SQL knowledge base.

3 Method

In this section, we present Knowledge-Augmented Text-to-SQL (KAT-SQL), an approach that automatically constructs a knowledge base and utilizes the relevant knowledge from it for text-to-SQL.

3.1 Problem Statement

We begin with formally explaining text-to-SQL and the knowledge augmentation technique for it.

Text-to-SQL Text-to-SQL aims to translate a natural language query from a user into a syntactically correct and semantically precise SQL statement. Formally, let $\boldsymbol{q}$ be the user query (consisting of a sequence of tokens) and $\mathcal{D}$ be the database schema containing multiple tables and columns. Then, the SQL generation model f can be represented as follows: $\boldsymbol{s} = f(\boldsymbol{q}, \mathcal{D})$ where $\boldsymbol{s}$ is the SQL statement (consisting of a sequence of tokens) that attempts to retrieve the information requested by $\boldsymbol{q}$ over $\mathcal{D}$.

In this work, we operationalize f with LLMs, to harness their strong capability in understanding the semantics of $\boldsymbol{q}$ and generating the corresponding SQL code $\boldsymbol{s}$, as follows: $\boldsymbol{s} = \texttt{LLM}_\theta(\mathcal{T}(\boldsymbol{q}, \mathcal{D}))$ where θ is the model parameters and $\mathcal{T}$ is the prompt template. Typically, the model parameters θ remain fixed due to the high costs associated with further fine-tuning of them and sometimes their limited accessibility. Also, the prompt template $\mathcal{T}$ serves as a structured format that outlines the context, which includes task descriptions and instructions as well as few-shot demonstrations, to guide the model in generating accurate SQL codes.

Notably, while there have been great successes in advancing the LLM itself and optimizing its usage for text-to-SQL, such as using advanced prompting techniques or breaking down the task into multiple subtasks (Wei et al., 2022; Liu and Tan, 2023; Tai et al., 2023; Gu et al., 2023; Pourreza and Rafiei, 2023; Wang et al., 2023a), these improvements alone may not be sufficient to fully handle queries that require the deep domain knowledge or precise understanding of complex database schemas. In other words, the internal parametric knowledge of LLMs, while robust, may not fully encompass the diverse range of query variations and database structures, especially when these databases have distinct schemas or certain specialized terminology.

Knowledge-Augmented Text-to-SQL To tackle the aforementioned limitations, we focus on augmenting text-to-SQL with the knowledge relevant to the query, providing valuable insights into the domain-specific terminology and complex database schemas. If we denote this knowledge as $\boldsymbol{k}$, then the previous text-to-SQL process is redefined to incorporate it, as follows: $\boldsymbol{s} = \texttt{LLM}_\theta(\mathcal{T}(\boldsymbol{q}, \boldsymbol{k}, \mathcal{D}))$.

While there have been few studies that explore this knowledge-augmented text-to-SQL paradigm, there are still a couple of challenges. Specifically, Dou et al. (2022) and Li et al. (2023) propose collecting and annotating the explicit knowledge required to convert queries into SQL statements. Yet, to operationalize, this annotation-based approach can be costly and time-consuming, especially when dealing with a large number of diverse queries. On the other hand, Hong et al. (2024) propose an automatic generation of knowledge, based on the question and its associated database schema. However, this method is still limiting as it generates only a few pieces of knowledge for each query without leveraging the potential for reuse. In contrast, since much of the knowledge used for one query can be applicable to multiple similar queries (See Figure 1, Right), we aim to design a more effective approach for knowledge augmentation, discussed below.

3.2 Knowledge Base Construction

To address the aforementioned limitations of existing approaches in knowledge augmentation for text-to-SQL, we propose a novel approach to automatically construct a comprehensive and reusable knowledge base. Ideally, this can serve as a foundational resource, encapsulating diverse domain information and offering insights into various database schemas, to enhance the understanding of queries and their associated database structures.

Formally, we design this knowledge base $\mathcal{K}$ as a collection of knowledge entries, each represented as a concise sentence, denoted as follows: $\boldsymbol{k} \in \mathcal{K}$. For instance, in the medical domain, one knowledge entry might be "Abnormal white blood cell count refers to WBC $\leq$ 3.5 or WBC $\geq$ 9.0", which describes the abnormal range of white blood cell counts and its corresponding column name "WBC" in the database schema, applicable to queries related to abnormal white blood cells. The next question to answer is then how to construct this knowledge base based on the available resources.

In this work, we start with collecting all the existing knowledge entries from the publicly available dataset (Li et al., 2023), which includes the knowledge and its related pair of query and database schema. Yet, while this initial collection can serve as the foundational layer of our knowledge base, it may not capture the full scope of the required information. To address this gap, we propose an automatic knowledge base expansion technique that leverages LLMs, which possess domain-specific knowledge and the ability to comprehend the given context (including instructions, codes, and database structures) by generating additional knowledge entries. Specifically, given the query and its associated database schema from the available datasets, we prompt LLMs (along with a prompting template $\mathcal{T}$ for knowledge generation) to produce the knowledge, formulated as follows: $\boldsymbol{k} = \texttt{LLM}(\mathcal{T}(\boldsymbol{q}, \mathcal{D}))$, and then store this knowledge $\boldsymbol{k}$ into the knowledge base $\mathcal{K}$. In addition, as it may be more accurate and reliable to provide the LLM with relevant examples (which can help it understand the context, nuances, and expectations of the desired output), we further prepend the small number of relevant examples into the prompt of LLM. It is worth noting that these examples are comprised of the triplets of the user queries, their associated database schemas, and the knowledge they are derived from, and that those triplets come from the existing dataset (used to construct the initial knowledge base). Also, we select only those highly relevant to the query based on its embedding-level cosine similarities with samples from the existing dataset, calculated by MPNet (Song et al., 2020). This process can ultimately enable the LLM to generate more precise and contextually appropriate knowledge for text-to-SQL.

In addition to this relevant example-based knowledge generation approach, to further enrich the diversity and comprehensiveness of the knowledge base, we implement a simple yet effective strategy that involves sampling and permutation of few-shot examples provided to the LLM. Specifically, for the given query and its associated database schema, instead of generating their corresponding knowledge only once, we iteratively sample a different set of relevant examples (provided to contextualize the LLM) multiple times and further permute their order. This can allow the LLM to explore different contextual nuances and generate a wider range of knowledge entries, with the goal of ultimately increasing the robustness and applicability of the knowledge base for a broader range of queries.

3.3 Text-to-SQL with Knowledge Base

Based on the LLM-powered knowledge base construction process, we now have the knowledge base $\mathcal{K}$. Hereafter, the next question to answer is then how to use this knowledge base for text-to-SQL.

Algorithm 1 Knowledge-Augmented Text-to-SQL

Require: Dataset $\boldsymbol{D}$ containing query-schema pairs $(\boldsymbol{q}, \mathcal{D})$; LLM model LLM; Prompt templates $\mathcal{T}$
Ensure: SQL statement $\boldsymbol{s}$ for a given query $\boldsymbol{q}$

Phase 1: Knowledge Base Construction
$\mathcal{K} \leftarrow \{\} \cup \boldsymbol{D}$ ▷ Initialize knowledge base
for all $(\boldsymbol{q}, \mathcal{D}) \in \boldsymbol{D}$ **do**
  $\mathcal{E} \leftarrow$ Retrieve top-k relevant examples from $\boldsymbol{D}$
  $\boldsymbol{k}_{\text{new}} \leftarrow \texttt{LLM}(\mathcal{T}_{\text{gen}}(\boldsymbol{q}, \mathcal{D}, \mathcal{E}))$ ▷ Generate knowledge
  $\mathcal{K} \leftarrow \mathcal{K} \cup \boldsymbol{k}_{\text{new}}$ ▷ Store knowledge
end for
Phase 2: Knowledge-Augmented SQL Generation
function KAT-SQL($\boldsymbol{q}, \mathcal{D}, \mathcal{K}$)
  $\{\boldsymbol{k}_i\}_{i=1}^{j} \leftarrow$ Retrieve top-j knowledge from $\mathcal{K}$
  $\boldsymbol{k}' \leftarrow \texttt{LLM}(\mathcal{T}_{\text{ref}}(\boldsymbol{q}, \{\boldsymbol{k}_i\}_{i=1}^{j}, \mathcal{D}))$ ▷ Refine knowledge
  $\boldsymbol{s} \leftarrow \texttt{LLM}(\mathcal{T}_{\text{text-to-SQL}}(\boldsymbol{q}, \boldsymbol{k}', \mathcal{D}))$ ▷ Generate SQL
  return $\boldsymbol{s}$
end function

Figure 2: A simplified overview of the proposed KAT-SQL method. Please see Algorithms 2 and 3 for detailed versions.

Given the extensive nature of $\mathcal{K}$, containing a large number of entries, it is crucial to identify and retrieve the most pertinent entries for the query $\boldsymbol{q}$. Formally, this process can be represented as follows: $\{\boldsymbol{k}_i\}_{i=1}^{j} = \texttt{Retriever}(\boldsymbol{q}, \mathcal{K})$. Also, this can be operationalized by calculating the embedding-level similarities between the query and all the knowledge entries in the knowledge base, then selecting the top-j similar entries $\{\boldsymbol{k}_i\}_{i=1}^{j}$, where embeddings are obtained from a sentence embedding model (Karpukhin et al., 2020; Song et al., 2020). Moreover, to further enhance the retrieval accuracy, we train this embedding model with contrastive learning, which maximizes the similarity between the query and its relevant knowledge while minimizing the similarities of others, denoted as follows: $-\log \frac{\exp(\text{sim}(\boldsymbol{q},\boldsymbol{k}^+)/\tau)}{\exp(\text{sim}(\boldsymbol{q},\boldsymbol{k}^+)/\tau)+\sum_{\boldsymbol{k}^-}\exp(\text{sim}(\boldsymbol{q},\boldsymbol{k}^-)/\tau)}$, where $\text{sim}(\boldsymbol{q}, \boldsymbol{k})$ denotes the similarity measure between query $\boldsymbol{q}$ and knowledge $\boldsymbol{k}$, τ is the temperature parameter, $\boldsymbol{k}^+$ is the relevant knowledge, and $\boldsymbol{k}^-$ represents the set of irrelevant knowledge.

Note that while the retrieved knowledge entries from $\mathcal{K}$ are relevant to the given query and can assist in SQL statement formulation, they may require additional refinement to perfectly align with the query's specific needs. For instance, if the user query pertains to abnormal data conditions, but the retrieved knowledge primarily focuses on normal data, a direct application of this knowledge could lead to inaccurate SQL generation. To address this issue, we further prompt the LLM to generate the knowledge tailored to the given query by considering its relevant knowledge entries and database schema, as follows: $\boldsymbol{k}' = \texttt{LLM}(\mathcal{T}(\boldsymbol{q}, \{\boldsymbol{k}_i\}_{i=1}^{j}, \mathcal{D}))$, where $\{\boldsymbol{k}_i\}_{i=1}^{j}$ is the knowledge retrieved from $\mathcal{K}$. This refined knowledge $\boldsymbol{k}'$ is subsequently used as input, along with the user query and its associated database schema, to guide the text-to-SQL LLM in generating a more accurate and contextually appropriate SQL statement: $\boldsymbol{s} = \texttt{LLM}(\mathcal{T}(\boldsymbol{q}, \boldsymbol{k}', \mathcal{D}))$. Please see Algorithm 1 for our overall approach.

4 Experimental Setup

4.1 Datasets and Tasks

Datasets To validate the efficacy of KAT-SQL, we first use two widely used text-to-SQL benchmark datasets, namely BIRD (Li et al., 2023) and Spider (Yu et al., 2018). Specifically, BIRD is a recently released large-scale text-to-SQL dataset, built on top of 95 distinct databases spanning 37 domains. Additionally, each query in this dataset is associated with knowledge that is manually annotated by humans, providing a useful prior for formulating SQL statements. Spider is another benchmark dataset, built upon 200 databases across 138 domains. Unlike BIRD, samples in Spider do not have annotated knowledge for text-to-SQL. Lastly, we consider a challenging real-world text-to-SQL data, namely CSTINSIGHT, which is designed with actual customer queries over a data lakehouse with 34 tables without human-annotated knowledge.

Tasks/Scenarios We evaluate our KAT-SQL on three realistic text-to-SQL tasks. First of all, we consider the scenario where the prior information about some samples and their associated knowledge for each database is available, meaning that the databases used in training samples overlap with those in test samples (Overlap). We note that this setting is practical, since annotating a few pairs of questions and their corresponding knowledge for each database in advance is feasible. In addition to this, we test KAT-SQL with the existing benchmark setup, which is more challenging since it assumes there are no overlaps between databases during the training and test phases (Non-Overlap). In other words, no samples from the test-time databases are available beforehand, which means the model should be able to generalize to test-time queries based on the schemas of test-time databases as well as the samples and their associated knowledge from the different (training-time) databases. Lastly, we validate KAT-SQL on the most challenging scenario, where there are no overlaps between the databases used during training and testing, but also no knowledge is available for both training and test samples. This setup aims to test the model's ability to generalize (in the absence of any prior knowledge about the dataset), allowing us to evalu-

Table 1: Main results on text-to-SQL benchmark datasets across multiple scenarios, with the best results in bold.

Methods	BIRD (Overlap) EX	BIRD (Overlap) VES	BIRD (Non-Overlap) EX	BIRD (Non-Overlap) VES	Spider EX	Spider VES	CSTINSIGHT EX	CSTINSIGHT VES
No Knowledge	23.76	28.81	20.66	16.72	70.99	37.53	4.76	5.28
DELLM	34.70	33.15	24.64	19.27	72.44	42.90	11.90	12.02
KAT-SQL (Ours)	**41.18**	**41.33**	**41.07**	**31.14**	**74.56**	**47.20**	**14.29**	**14.50**
Oracle Knowledge	54.67	49.71	49.41	37.93	N/A	N/A	N/A	N/A

ate how well our knowledge base constructed with one dataset performs on different datasets. Notably, since the Spider and CSTINSIGHT datasets have no available knowledge for all queries, we use them for the most challenging last scenario; meanwhile, we use the BIRD dataset for the first two scenarios.

4.2 Baselines and Our Model

We compare our KAT-SQL approach against relevant baselines that target our primary objective of improving knowledge-augmented text-to-SQL systems, which vary in their usage of knowledge. We note that for the fairest comparison, we fix the LLM as the same for all methods, explained as follows: 1. **No Knowledge** – which uses only the queries themselves to formulate the SQL statements without any additional knowledge. 2. **DELLM** – which generates the knowledge based on the query and its relevant database structures, and use this synthesized knowledge for text-to-SQL (Hong et al., 2024). 3. **KAT-SQL** – which is our model, building the knowledge base and utilizing the knowledge from it (with retrieval) for text-to-SQL. 4. **Oracle Knowledge** – which uses oracle knowledge annotated by humans, along with the queries to generate the SQL statements. This approach serves as an upper bound and is not directly comparable to other models due to its reliance on accurate, manually curated knowledge that is typically unavailable.

4.3 Evaluation Metrics

Following the standard evaluation protocols from prior work (Li et al., 2023; Hong et al., 2024), we use the following two metrics: 1) Execution Accuracy (EX), which measures the ratio of generated SQL code that has the same execution results with ground-truth SQL code; 2) Valid Efficiency Score (VES), which considers the efficiency of generated SQLs by weighting them based on their relative efficiency improvement over ground-truth SQLs further multiplied by execution accuracy.

4.4 Implementation Details

We mainly use Llama-3 70B (AI@Meta, 2024) as the basis for text-to-SQL generation and knowledge generation across all baselines and our model variants for most experiments, for a fair comparison, while we also experiment with other LLMs in an analysis (Table 6) to see the robustness of KAT-SQL. For the hyperparameters, except for the temperature (which we set as 0.0 for reproducibility), we use its default values. In addition, for the retriever, we use MPNet (Song et al., 2020), which is based on dense retrieval; we train it with a batch size of 128 and a number of training epochs of 30. We provide the detailed prompts used to elicit the knowledge and SQL generations in Appendix A.

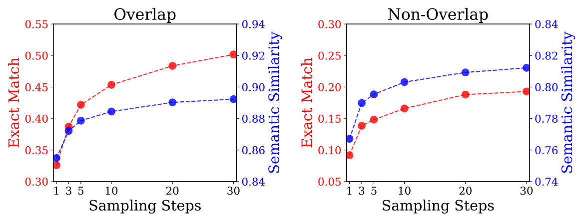

Figure 3: Results for coverage and relevance of knowledge entries in the constructed knowledge base against gold knowledge, with different numbers of knowledge generation steps.

5 Experimental Results and Analyses

Main Results We provide main results in Table 1, which confirms that our KAT-SQL approach consistently outperforms all baselines by large margins. Specifically, while we observe some performance improvement of the knowledge-augmented text-to-SQL approach (namely DELLM, which generates a few pieces of knowledge for each query) over the baseline without knowledge augmentation, KAT-SQL achieves even greater gains, demonstrating the effectiveness of our knowledge base construction-based text-to-SQL paradigm. However, the performance of the (incomparable) model with the oracle knowledge (annotated by human experts) remains superior to all other approaches, which suggests potential future opportunities for developing a more advanced pipeline for knowledge generation.

Analysis on Knowledge Base To further understand the coverage and relevance of the knowledge within our knowledge base, we compare each piece of knowledge required for test-time queries with all the available entries in the knowledge base, as a function of the number of knowledge generation steps during knowledge base construction. For evaluation, we use two metrics: Exact Match, which

Table 2: Results for knowledge generation with and without the use of the Knowledge Base (KB), while varying the prompt construction with and without the relevant few-shot examples.

KB	Few-Shot	Overlap EM	Overlap SS	Non-Overlap EM	Non-Overlap SS
w/o KB	Random	10.96	68.77	7.88	66.77
	Retrieval	20.21	73.62	9.24	68.78
w/ KB	Random	11.13	69.14	7.93	66.80
	Retrieval	**24.97**	**77.87**	**12.94**	**71.24**

identifies whether the knowledge base contains an entry that precisely matches the knowledge required for a given query, and Semantic Similarity, which assesses how closely related the most similar entry (in the knowledge base) is to the required knowledge based on the embedding-level similarity. As shown in Figure 3, we observe that, under the Overlap setting, half of the knowledge entries needed for test-time queries are available in the knowledge base while the Semantic Similarity is around 90%, which demonstrates substantial coverage by our knowledge base. In addition, for the challenging setup where training and test databases are distinct, we still observe that 20% of the test-time knowledge entries are available in the knowledge base and that the Semantic Similarity exceeds 80%, showing the utility of our knowledge base. Finally, as we increase the number of knowledge generation steps for each instance during knowledge base construction, we observe a corresponding improvement in both coverage and relevance of our knowledge base, which supports the effectiveness of our expansion strategy to enrich its diversity.

Analysis on Knowledge Generation Recall that we further refine the retrieved knowledge to make it more suitable for each query, in addition to constructing the knowledge base and retrieving the relevant knowledge. Thus, to see how relevant the generated knowledge is to the human-annotated gold knowledge with regards to the use of our knowledge base, we report comparison results according to Exact Match and Semantic Similarity (SS) in Table 2. We observe that when we retrieve the relevant knowledge from the knowledge base and then use it for knowledge generation, there are performance gains over the case where we do not leverage it, which indicates that the retrieved knowledge is helpful in formulating the necessary knowledge for test-time queries. We also provide few-shot examples to guide the knowledge generation model in generating useful knowledge in the right format, and when we select them based on their similarities with the given query, we observe further gains in the quality of the generated knowledge.

Table 3: Text-to-SQL results without using any knowledge, based on the retrieved knowledge, and based on the refined knowledge from the retrieved knowledge (Our KAT-SQL).

Settings	Models	EX
Overlap	KAT-SQL (Ours)	41.18
	w/o Generation	38.94
	w/o Retrieval & Generation	23.76
Non-Overlap	KAT-SQL (Ours)	41.07
	w/o Generation	38.42
	w/o Retrieval & Generation	20.66

Table 4: Retrieval results with different scenarios and models.

Settings	Models	MRR	Top@3	Top@10
Overlap	BERT	0.5506	0.6621	0.8911
	TAS-B	0.5630	0.6943	0.9035
	TAS-B*	0.8288	0.9143	0.9765
Non-Overlap	BERT	0.2148	0.2692	0.4231
	TAS-B	0.2364	0.3846	0.4615
	TAS-B*	0.7565	0.8347	0.9210

Beyond evaluating the quality of the generated knowledge by comparing it to the human-annotated gold knowledge, we also examine the impact of knowledge generation on downstream text-to-SQL performance with and without the incorporation of generated knowledge. As shown in Table 3, compared to the results without the knowledge retrieval and generation on both Overlap and Non-Overlap settings, there are substantial improvements when we incorporate the retrieved knowledge from our knowledge base into the text-to-SQL generation process. Furthermore, instead of directly using the retrieved knowledge, refining this retrieved knowledge yields additional improvements, underscoring the importance of not only retrieving relevant knowledge but also tailoring it to better align with the specific needs of test-time queries.

Retrieval Analysis We also analyze the accuracy of knowledge retrieval from our knowledge base by reporting its retrieval performance in Table 4 according to Mean Reciprocal Rank (MRR) and Top@K Accuracy. We observe that the retrieval accuracy on the Overlap setting is higher than that on the Non-Overlap setting, due to the less availability of relevant knowledge required for test-time queries in the Non-Overlap setting. Yet, when we replace the knowledge base constructed from our approach with the Oracle knowledge base (*), which includes all the necessary knowledge for test-time queries, the MRR on both settings reaches around 80%, indicating the importance of expanding the coverage of the knowledge base for accurate knowledge retrieval. The table also compares the performance of different basis models for retrieval – BERT (Devlin et al., 2019) and TAS-B (Hofstätter et al., 2021) – with the latter being fine-tuned for retrieval. It can

Table 5: Breakdown text-to-SQL results into overlapping and non-overlapping domain settings between training (knowledge base construction) and test (text-to-SQL evaluation) databases.

Models	Overlap	Non-Overlap
No Knowledge	22.85	16.20
DELLM	27.20	19.43
KAT-SQL (Ours)	**49.37**	**24.19**

be seen that the extra training of the model on retrieval tasks aids in achieving superior performance for retrieving the knowledge for text-to-SQL.

Generalization Analysis to Different Domains To see whether our knowledge base can be generalizable to databases of different domains (that are not overlapped with those for knowledge base construction), we breakdown the performance based on whether test databases share domains with training databases or belong to different domains (according to 37 domains categorized from Li et al. (2023)). As shown in Table 5, our KAT-SQL achieves substantially higher performance when test databases overlap with training domains compared to those from unseen domains; however, even in the latter case, KAT-SQL still outperforms existing baselines. These results indicate that, while the lack of domain overlaps degrades the performance, our knowledge base still provides meaningful benefits for unseen domains, demonstrating its generalizability.

Analysis with Different LLMs To evaluate how robust our KAT-SAL approach is across different LLMs, we conduct the additional analysis instantiating the text-to-SQL and knowledge generation models with other recent LLMs such as Granite 34B (Mishra et al., 2024) and Mixtral 8x7B (Jiang et al., 2024); results are shown in Table 6. From this, we observe that KAT-SQL consistently outperforms all baselines regardless of the choice of LLMs, which demonstrates the effectiveness and versatility of our proposed approach.

Finally, we augment the state-of-the-art text-to-SQL model (in the setting without oracle knowledge) on the BIRD leaderboard (Li et al., 2023), namely ExSL + granite-20b-code, using the knowledge generated from our proposed knowledge base construction-based approach. As shown in Table 7, we observe that the text-to-SQL model combined with our KAT-SQL approach establishes the new state-of-the-art performance, highlighting the value of our method as a powerful tool for text-to-SQL.

Analysis on Efficiency While our primary focus is on improving the text-to-SQL accuracy through knowledge base construction and augmentation, we also consider the efficiency of our approach. It is worth noting that the construction of the knowledge base is performed offline and does not affect real-time query processing; therefore, the extra computational overhead comes from retrieving relevant knowledge and generating the SQL statement in response to the query. In this regard, our retrieval process accounts for only 2% of the overall generation time, thanks to efficient search algorithm (Douze et al., 2024), making its impact negligible. Also, although incorporating knowledge into the text-to-SQL pipeline increases the prompt length by 30%, this overhead aligns with other knowledge augmentation methods (such as DELLM) and does not introduce additional latency specific to our approach. Overall, each query is processed under 5 seconds.

Table 6: Text to SQL results with different LLMs.

LLMs	Methods	Overlap	Non-Overlap
Llama	No Knowledge	23.76	20.66
	DELLM	34.70	24.64
	KAT-SQL	41.18	41.07
	Oracle Knowledge	54.67	49.41
Granite	No Knowledge	25.83	17.75
	DELLM	34.04	20.21
	KAT-SQL	39.28	35.83
	Oracle Knowledge	46.56	38.32
Mixtral	No Knowledge	11.75	10.58
	DELLM	27.17	11.29
	KAT-SQL	29.31	20.30
	Oracle Knowledge	37.26	30.88

Table 7: Results of our KAT-SQL approach with the state-of-the-art text-to-SQL model on the BIRD leaderboard.

Models	EX
ChatGPT	24.05
ChatGPT + CoT	25.88
ExSL + granite-20b-code	51.69
ExSL + granite-20b-code w/ KAT-SQL (Ours)	57.56
ExSL + granite-20b-code w/ Oracle Knowledge	65.38

Examples We provide examples for the knowledge generation and text-to-SQL in Table 9 as well as the entries in the knowledge base in Table 10.

6 Conclusion

In this work, we proposed a novel knowledge base construction-based text-to-SQL approach called KAT-SQL, based on the motivation that one piece of knowledge can be reused across multiple queries and databases. Our approach involves the creation of the knowledge base from which relevant knowledge is retrieved and utilized to generate SQL statements from queries. Through extensive evaluations on multiple datasets with two different scenarios, we showed that our KAT-SQL outperforms relevant knowledge-augmented text-to-SQL baselines. In addition, our detailed analyses highlight the effectiveness of each component in the knowledge generation and retrieval processes, but also the high coverage and relevance of the entries in the base.

Limitations

In this work, we propose constructing a knowledge base and then leveraging it for text-to-SQL tasks, showcasing the clear advantages of constructing the knowledge base for text-to-SQL. However, as the performance gaps between the models with oracle knowledge and the generated knowledge from our knowledge base indicate, there is still room to improve the coverage of the knowledge base, which is a promising avenue for future work.

Ethics Statement

We recognize that any text-to-SQL system, including our proposed approach, may carry the inherent risk of generating SQL queries that may inadvertently or intentionally access, modify, or delete sensitive information within a database. While this vulnerability is not exclusive to our method and is a well-known challenge in the broader field of text-to-SQL systems, it underscores the importance of implementing robust security measures and access controls before deploying such systems. Similar to this, safety is particularly crucial in our application, so as to avoid the risk of sensitive information being stored in the knowledge base and subsequently being inappropriately reused.

References

AI@Meta. 2024. Llama 3 model card.

Dimitrios Alivanistos, Selene Baez Santamaría, Michael Cochez, Jan-Christoph Kalo, Emile van Krieken, and Thiviyan Thanapalasingam. 2022. Prompting as probing: Using language models for knowledge base construction. In *Proceedings of the Semantic Web Challenge on Knowledge Base Construction from Pre-trained Language Models 2022 co-located with the 21st International Semantic Web Conference (ISWC2022), Virtual Event, Hanghzou, China, October 2022*, volume 3274 of *CEUR Workshop Proceedings*, pages 11–34. CEUR-WS.org.

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. 2023. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Ruichu Cai, Boyan Xu, Zhenjie Zhang, Xiaoyan Yang, Zijian Li, and Zhihao Liang. 2018. An encoder-decoder framework translating natural language to database queries. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 3977–3983. ijcai.org.

Shuaichen Chang and Eric Fosler-Lussier. 2023. How to prompt llms for text-to-sql: A study in zero-shot, single-domain, and cross-domain settings. *arXiv preprint arXiv:2305.11853*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

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

Longxu Dou, Yan Gao, Xuqi Liu, Mingyang Pan, Dingzirui Wang, Wanxiang Che, Dechen Zhan, Min-Yen Kan, and Jian-Guang Lou. 2022. Towards knowledge-intensive text-to-sql semantic parsing with formulaic knowledge. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5240–5253. Association for Computational Linguistics.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2024. Text-to-sql empowered by large language models: A benchmark evaluation. *Proc. VLDB Endow.*, 17(5):1132–1145.

Zihui Gu, Ju Fan, Nan Tang, Lei Cao, Bowen Jia, Sam Madden, and Xiaoyong Du. 2023. Few-shot text-to-sql translation using structure and content prompt learning. *Proc. ACM Manag. Data*, 1(2):147:1–147:28.

Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 113–122. ACM.

Zijin Hong, Zheng Yuan, Hao Chen, Qinggang Zhang, Feiran Huang, and Xiao Huang. 2024. Knowledge-to-sql: Enhancing SQL generation with data expert LLM. *arXiv preprint arXiv:2402.11517*.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Dongjun Lee, Choongwon Park, Jaehyuk Kim, and Heesoo Park. 2024. MCS-SQL: leveraging multiple prompts and multiple-choice selection for text-to-sql generation. *arXiv preprint arXiv:2405.07467*.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chen-Chuan Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can LLM already serve as A database interface? A big bench for large-scale database grounded text-to-sqls. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Xiping Liu and Zhao Tan. 2023. Divide and prompt: Chain of thought prompting for text-to-sql. *arXiv preprint arXiv:2304.11556*.

Mayank Mishra, Matt Stallone, Gaoyuan Zhang, Yikang Shen, Aditya Prasad, Adriana Meza Soria, Michele Merler, Parameswaran Selvam, Saptha Surendran, Shivdeep Singh, Manish Sethi, Xuan-Hong Dang, Pengyuan Li, Kun-Lung Wu, Syed Zawad, Andrew Coleman, Matthew White, Mark Lewis, Raju Pavuluri, Yan Koyfman, Boris Lublinsky, Maximilien de Bayser, Ibrahim Abdelaziz, Kinjal Basu, Mayank Agarwal, Yi Zhou, Chris Johnson, Aanchal Goyal, Hima Patel, S. Yousaf Shah, Petros Zerfos, Heiko Ludwig, Asim Munawar, Maxwell Crouse, Pavan Kapanipathi, Shweta Salaria, Bob Calio, Sophia Wen, Seetharami Seelam, Brian Belgodere, Carlos A. Fonseca, Amith Singhee, Nirmit Desai, David D. Cox, Ruchir Puri, and Rameswar Panda. 2024. Granite code models: A family of open foundation models for code intelligence. *arXiv preprint arXiv:2405.04324*.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of GPT-4. *arXiv preprint arXiv:2306.02707*.

Anmol Nayak and Hariprasad Timmapathini. 2023. LLM2KB: constructing knowledge bases using instruction tuned context aware large language models. In *Joint proceedings of the 1st workshop on Knowledge Base Construction from Pre-Trained Language Models (KBC-LM) and the 2nd challenge on Language Models for Knowledge Base Construction (LM-KBC) co-located with the 22nd International Semantic Web Conference (ISWC 2023), Athens, Greece, November 6, 2023*, volume 3577 of *CEUR Workshop Proceedings*. CEUR-WS.org.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: decomposed in-context learning of text-to-sql with self-correction. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

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

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Shivchander Sudalairaj, Abhishek Bhandwaldar, Aldo Pareja, Kai Xu, David D. Cox, and Akash Srivastava. 2024. LAB: large-scale alignment for chatbots. *arXiv preprint arXiv:2403.01081*.

Chang-Yu Tai, Ziru Chen, Tianshu Zhang, Xiang Deng, and Huan Sun. 2023. Exploring chain of thought style prompting for text-to-sql. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 5376–5393. Association for Computational Linguistics.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Blerta Veseli, Simon Razniewski, Jan-Christoph Kalo, and Gerhard Weikum. 2023. Evaluating the knowledge base completion potential of GPT. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 6432–6443. Association for Computational Linguistics.

Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Qian-Wen Zhang, Zhao Yan, and Zhoujun Li. 2023a. MAC-SQL: A multi-agent collaborative framework for text-to-sql. *arXiv preprint arXiv:2312.11242*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023c. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13484–13508. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Sqlizer: query synthesis from natural language. *Proc. ACM Program. Lang.*, 1(OOPSLA):63:1–63:26.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 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, Brussels, Belgium, October 31 - November 4, 2018*, pages 3911–3921. Association for Computational Linguistics.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, USA, August 4-8, 1996, Volume 2*, pages 1050–1055. AAAI Press / The MIT Press.

A Prompts

We provide the prompts used to elicit the knowledge generation and the SQL generation in Table 8.

B Algorithms

We provide the pseudo-code for knowledge base construction in Algorithm 2 and the pseudo-code for our full KAT-SQL approach in Algorithm 3.

C Additional Experimental Results

Knowledge Base Statistics The resulting knowledge base for the database overlapping and non-overlapping scenarios contains 86,254 and 117,328 knowledge entries, respectively, which are greater than the original number of knowledge entries annotated in the BIRD dataset, which is 12,751.

Knowledge Base Construction Cost While the construction of the knowledge base is performed offline and does not impact real-time operations of text-to-SQL, we provide the cost to construct the knowledge base for our KAT2SQL approach to enable researchers to estimate resource requirements for scaling and implementation. Note that the exact computational costs and time required for knowledge base construction vary depending on hardware types and configurations, and with four H100 GPUs that can process 2K tokens per second and generate 10 tokens per second for Llama 70B, the time required to generate each knowledge entry is around 2 seconds. Therefore, for the knowledge base with 100K entries, the total generation time would be 56 hours divided by the number of parallel models (it completes in 7 hours with 8 models).

Retrieval over Different Knowledge Sources It is worthwhile to note that, for text-to-SQL tasks, it is crucial to consider the relationship between the query and the database (in addition to the consideration of the domain-specific knowledge for domain-specific queries); therefore, using the unstructured knowledge sources (such as web search) may not be optimal for this purpose since they often lack the structured, schema-specific information necessary for accurately formulating SQL queries. Nevertheless, to further validate this claim, we perform retrieval over Wikipedia, instead of performing retrieval over the constructed knowledge base, and observe only the marginal performance gain (3%) compared to the baseline without augmentation.

Table 8: A list of prompts that we use for knowledge generation and SQL generation. It is worth noting that the variable inside the parentheses {} is replaced with its actual values.

Types	Prompts
Knowledge Generation	DB Schema: {Database Schema} Question: {Few-Shot Question 1} Evidence: {Few-Shot Evidence 1} Question: {Few-Shot Question 2} Evidence: {Few-Shot Evidence 2} ... Question: {Few-Shot Question 10} Evidence: {Few-Shot Evidence 10} Question: {Target Question} Evidence:
SQL Generation	DB Schema: {Database Schema} Question: {Few-Shot Question 1} Evidence: {Few-Shot Evidence 1} SQL: {Few-Shot SQL 1} Question: {Few-Shot Question 2} Evidence: {Few-Shot Evidence 2} SQL: {Few-Shot SQL 2} ... Question: {Few-Shot Question 10} Evidence: {Few-Shot Evidence 10} SQL: {Few-Shot SQL 10} Question: {Target Question} Evidence: {Generated Knowledge} SQL:

Algorithm 2 Knowledge Base Construction for KAT-SQL

Require: Dataset $\boldsymbol{D}$ containing query-schema-knowledge triplets $(\boldsymbol{q}, \mathcal{D}, \boldsymbol{k})$; Prompt template $\mathcal{T}$
Ensure: Knowledge base $\mathcal{K}$

$\mathcal{K} \leftarrow \{\}$ ▷ Initialize an empty knowledge base
for all $(\boldsymbol{q}, \mathcal{D}, \boldsymbol{k}) \in \boldsymbol{D}$ **do**
  $\mathcal{K} \leftarrow \mathcal{K} \cup \boldsymbol{k}$ ▷ Add existing knowledge to the knowledge base
end for
for all query-schema pair $(\boldsymbol{q}, \mathcal{D}) \in \boldsymbol{D}$ **do**
  $\mathcal{E} \leftarrow$ Top-k relevant examples to the query $\boldsymbol{q}$ from $\boldsymbol{D}$
  for $i = 1$ to N **do** ▷ Iteratively expand knowledge
    $\mathcal{E}_{\text{perm}} \leftarrow$ Permute examples $\mathcal{E}$
    $\boldsymbol{k}_{\text{new}} \leftarrow \mathtt{LLM}(\mathcal{T}(\boldsymbol{q}, \mathcal{D}, \mathcal{E}_{\text{perm}}))$ ▷ Generate knowledge using LLM with examples
    $\mathcal{K} \leftarrow \mathcal{K} \cup \boldsymbol{k}_{\text{new}}$ ▷ Store generated knowledge in the knowledge base
  end for
end for

Algorithm 3 Knowledge-Augmented Text-to-SQL (KAT-SQL)

Require: Query $\boldsymbol{q}$; Database schema $\mathcal{D}$; Knowledge base $\mathcal{K}$
Ensure: SQL statement $\boldsymbol{s}$

function KAT-SQL($\boldsymbol{q}$, $\mathcal{D}$, $\mathcal{K}$)
  $\{\boldsymbol{k}_i\}_{i=1}^{j} \leftarrow$ RETRIEVER($\boldsymbol{q}$, $\mathcal{K}$) ▷ Retrieve relevant knowledge entries from $\mathcal{K}$
  $\mathcal{T}_{\text{ref}} \leftarrow$ CREATEPROMPT($\boldsymbol{q}$, $\{\boldsymbol{k}_i\}_{i=1}^{j}$, $\mathcal{D}$) ▷ Construct the prompt with retrieved knowledge
  $\boldsymbol{k}' \leftarrow \mathtt{LLM}(\mathcal{T}_{\text{ref}})$ ▷ Refine knowledge using LLM
  $\mathcal{T}_{\text{aug}} \leftarrow$ CREATEPROMPT($\boldsymbol{q}$, $\boldsymbol{k}'$, $\mathcal{D}$) ▷ Augment the prompt with refined knowledge
  $\boldsymbol{s} \leftarrow \mathtt{LLM}(\mathcal{T}_{\text{aug}})$ ▷ Generate SQL with knowledge augmentation
  return $\boldsymbol{s}$
end function
function RETRIEVER($\boldsymbol{q}$, $\mathcal{K}$)
  Compute embeddings for $\boldsymbol{q}$ and all knowledge entries $\boldsymbol{k} \in \mathcal{K}$
  Retrieve top-j relevant knowledge entries $\{\boldsymbol{k}_i\}_{i=1}^{j}$ based on embedding similarities
  return $\{\boldsymbol{k}_i\}_{i=1}^{j}$
end function
function CREATEPROMPT($\boldsymbol{q}$, $\boldsymbol{k}$, $\mathcal{D}$)
  Construct the prompt template $\mathcal{T}$ using the query $\boldsymbol{q}$, knowledge $\boldsymbol{k}$, and database schema $\mathcal{D}$
  return $\mathcal{T}$
end function

Table 9: Case study of knowledge and SQL generation results from queries.

Example	Query	Generated Knowledge and SQL
Example 1	Which position has a lower minimum salary, Account Representative or Trainee?	**Generated Knowledge:** 1) lower minimum salary refers to `MIN(minsalary)` 2) Account Representative and Trainee are position titles **Generated SQL:** `SELECT positiontitle FROM position WHERE positiontitle = 'Account Representative' OR positiontitle = 'Trainee' ORDER BY minsalary ASC LIMIT 1`
Example 2	Among the employees working at the office in New York, how many of them have a good job performance?	**Generated Knowledge:** 1) New York refers to `state = 'NY'` 2) good job performance refers to `performance = 'Good'` **Generated SQL:** `SELECT COUNT(*) FROM employee AS T1 INNER JOIN location AS T2 ON T1.locationID = T2.locationID WHERE T2.state = 'NY' AND T1.performance = 'Good'`
Example 3	For all patients born in 1982, state if their albumin is within normal range.	**Generated Knowledge:** 1) albumin is within normal range refers to `ALB between 3.5 and 5.5` **Generated SQL:** `SELECT CASE WHEN T2.ALB >= 3.5 AND T2.ALB <= 5.5 THEN 'normal' ELSE 'abnormal' END FROM Patient AS T1 INNER JOIN Laboratory AS T2 ON T1.ID = T2.ID WHERE STRFTIME('%Y', T1.Birthday) = '1982'`

Table 10: Examples of original and its similar constructed knowledge within the knowledge base.

Example	Original Knowledge	Constructed Similar Knowledge
Example 1	albumin is within normal range refers to `ALB between 3.5 and 5.5`	1) albumin is outside the normal range refers to `ALB less than 3.5 or greater than 5.5` 2) glucose is within normal range refers to `GLU between 70 and 100 mg/dL` 3) Hemoglobin (Hb) is considered normal for males if levels range from `13.5 to 17.5 g/dL`
Example 2	Eligible free rate for K-12 = Free Meal Count (K-12) / Enrollment (K-12)	1) Eligible reduced-price rate for K-12 = Reduced-Price Meal Count (K-12) / Enrollment (K-12) 2) Eligible free meal rate for students aged 5-17 = Free Meal Count (Ages 5-17) / Enrollment (Ages 5-17) 3) Difference between K-12 and ages 5-17 enrollment = `Enrollment (K-12) - Enrollment (Ages 5-17)`
Example 3	Slovakia can be represented as `Country = 'SVK'`	1) France can be represented as `Country = 'FRA'` 2) Brazil can be represented as `Country = 'BRA'` 3) Monaco can be represented as `Country = 'MCO'`