Xiaohui: A Text-to-SQL Application Designed for Non-technical Users

Anonymous ACL submission

Abstract

Text-to-SQL, a critical task in natural language processing, aims to translate natural language questions into structured SQL queries for relational databases. Despite significant advancements, existing approaches often struggle with complex queries and domain adaptation due to the inherent ambiguity and variability of natural language. In this paper, a user-centered text-to-SQL system that reduces the usage barrier for non-technical users by abstracting the intermediate language of keyword instructions is designed. The key point is to allow users to independently determine whether the predicted query and execution results are acceptable, rather than relying on the complex SQL statements of traditional methods. Specifically, our implementation, Xiaohui, first designs keyword-to-Trino SQL mapping rules based on the real needs of partner enterprises, such as growth trends, comparisons, and percentage situations commonly found in business analysis. Secondly, it utilizes a LLM-based method fine-tuned with over 9,000 text-to-keyword samples and 1,276 question-query types-keyword examples for reference. On a test dataset designed based on the actual needs of enterprises, using Qwen as the basic model, the proposed system achieves evaluation performance comparable to or even surpassing mainstream prompt-based methods like DAIL-SQL and DIN-SQL.

1 Introduction

Text-to-SQL (Zelle and Mooney, 1996), also known as Natural Language to SQL (NL2SQL), empowers users to effortlessly retrieve data from relational databases by simply using natural language queries, eliminating the need to master intricate SQL query techniques. In recent years, spanning enterprise-level business analytics to individual health monitoring systems, the growing reliance of non-technical users on intuitive data query solutions has created both unprecedented demands and complex challenges for text-to-SQL technology (Floratou et al., 2024).

Over the early years, research in the text-to-SQL field primarily relied on rule-based methods (Mahmud et al., 2015; Xu et al., 2017; Yu et al., 2018b) and neural machine translation techniques (Zhong et al., 2017; Yu et al., 2018a; Ma et al., 2020), with sequence-to-sequence methodologies also playing an important role (Sutskever et al., 2014; Devlin et al., 2019). The recent emergence of large language models (LLMs) has revolutionized the field, marking a paradigm shift in natural language processing (NLP). Capitalizing on their sophisticated language comprehension, reasoning capabilities, and generative potential, LLMs have demonstrated remarkable success across diverse NLP applications, with text-to-SQL conversion being a particularly promising area of implementation. In recent years, various approaches collectively referred to as LLM-based methods have continuously made breakthroughs in the well-known large-scale cross-domain text-to-SQL benchmarks Spider (Yu et al., 2018c) and Bird (Li et al., 2023). These methods can be broadly categorized into two technical paths. The first approach involves utilizing closed-source LLMs to generate high-quality SQL predictions by employing advanced prompt engineering techniques. The second approach entails implementing supervised fine-tuning on open-source LLMs, thereby empowering localized models to assimilate specialized domain knowledge. Furthermore, multi-agent approaches (Wang et al., 2025) that integrate multiple LLMs have been explored for further improvements. However, advanced models such as GPT-4 present significant challenges in cost efficiency, data privacy, and execution time, as refining output quality (Li et al., 2024) often requires multiple LLM calls, significantly increasing processing time. Finally, both technical approaches inevitably face the challenge of user verification. Current research methods cannot entirely avoid erroneous predictions, and issues such as ambiguous queries, varying user expectations for results, and hallucinations (Qu et al., 2024) from LLMs remain challenges. When an executable yet semantically flawed SQL query is produced, non-technical users may lack the expertise to identify such errors. Overlooking these mistakes can lead to misguided decisions based on incorrect data, ultimately causing significant business consequences.

Furthermore, looking back at the development of text-to-SQL technology, high-quality datasets have also played a significant role in boosting its progress. For example, large-scale, cross-domain datasets such as Spider and its subsequent versions, Spider-Syn (Gan et al., 2021a), Spider-DK (Gan et al., 2021b), Spider-Realistic (Deng et al., 2021), require text-to-SQL systems to generalize on previously unseen databases (Zhong et al., 2020). Meanwhile, datasets such as Bird and Spider2V (Cao et al., 2024a) target more sophisticated database structures and complex query scenarios, significantly enhancing their relevance to real-world industrial implementations. Nevertheless, these datasets still fall short of fully encompassing the diverse and multifaceted requirements of practical

applications. Firstly, it is widely acknowledged that enterprise databases are often characterized by highly complex schemata, frequently containing hundreds of tables and thousands of fields (Floratou et al., 2024; Cao et al., 2024a; Lian et al., 2024). In actual enterprise environments, access to database information is strictly controlled, with users in different departments granted varying levels of access. This segmentation creates significant challenges for text-to-SQL systems, which must account for access permissions and operate on pre-filtered sub-schemata. Secondly, employees in commercial companies typically need to analyze numerical metrics such as sales growth, logistics performance, or market share proportions. However, existing datasets such as Spider and Bird lack training examples that reflect such real-world commercial analytics queries. This absence of domain-specific examples significantly impacts LLM-based methods that rely on in-context learning (ICL), limiting their ability to generalize to practical use cases. Thirdly, standard SQL, commonly used in academic research, is not suitable for commercial applications. Enterprises employ a variety of database management systems, each with distinct SQL dialects (MySQL, PostgreSQL, Trino, and others). Seamless translation between these dialects is often infeasible, which directly reduces the volume of training data available for enterprise-grade text-to-SQL systems.

In response to the aforementioned issues and challenges, we introduce a keyword representation as an intermediate language for the text-to-SQL task, which can succinctly capture the user's query intent and easily be checked by users without SQL expertise. Through continuous communication with commercial companies to understand specific needs and extensive testing, we have developed a stable and controllable keyword-to-SQL mapping rule. This approach can handle the majority of general query tasks while also allowing for the customization of query shortcuts based on enterprise requirements. Meanwhile, in this study, we assume that users can access pre-processed sub-tables, which are customized according to their permissions. Enterprises commonly use wide tables to consolidate data from multiple tables, providing access to sub-tables based on user-specific permissions. Schema-linking (Pourreza and Rafiei, 2023; Cao et al., 2024b) and business intelligence tools can now be used to obtain relevant sub-schema for specific queries. Moreover, to tackle the lack of training data for commercial analytics, we construct a dataset of more than 1,200 question-to-keyword pairs derived from real-world databases. These pairs serve as exemplars for LLMs, providing domain-specific references that facilitate effective contextual learning. Finally, recognizing the importance of SQL dialects in enterprise-grade applications, we select Trino SQL as our target language. High-performance database management systems such as Trino are critical for meeting the demands of commercial use cases. By aligning our approach with practical enterprise requirements, we aim to bridge the gap between academic research and real-world applications. Our main contributions are as follows:

- **User-oriented Text-to-SQL system based on keyword representation:** We develop a user-friendly text-to-SQL system centered on keyword representation, which has been refined through extensive real-world feedback and iterative improvements. Our intermediate language enables users to independently correct the system's predictions.

- **Extensive dataset annotation for training:** Utilizing datasets such as Spider, CSpider (Min et al., 2019), and Bird, we manually annotate over 16,000 question-to-keyword pairs for the English dataset and clean 9,028 data pairs for the Chinese corpus. Additionally, we generate a set of 1,276 newly annotated examples with detailed query types classifications. These resources enable the fine-tuning or training of models capable of generating keyword-based statements, facilitating their use in diverse applications.

- **SOTA-level execution accuracy through cost-effective methods:** Through comprehensive experiments on commercial analytics-related test datasets and real-world databases, we achieve execution accuracy comparable to state-of-the-art methods while employing more cost-efficient approaches.

2 Related Work

Figure 1 provides a brief overview of the development trajectory of text-to-SQL technology. Rule-based approaches have a long history in the field of database query parsing, with early work dating back to 1996. However, early hand-designed templates were limited in their ability to handle increasingly complex SQL queries. At the same time, enterprise users often have highly specific and domain-targeted query requirements. While basic query needs can typically be met, achieving a higher degree of customization often makes the system more suitable for specific roles or specialized tasks within an organization. In our approach, we address these challenges by categorizing query types into 17 distinct categories, offering fine-grained adaptability to diverse user needs. This categorization ensures extensive coverage of SQL structures, enabling the system to accommodate a wide range of query scenarios, including advanced and highly specific use cases. By balancing foundational query capabilities with high customizability, our approach is particularly well-suited for enterprise applications where user roles and tasks demand tailored solutions.

In the early stages of text-to-SQL research, neural text-to-SQL models were widely adopted. These models, such as RAT-SQL (Wang et al., 2020), SyntaxSQLNet (Yu et al., 2018b), and IRNet (Guo et al., 2019), were based on an encoder-decoder architecture (Sutskever et al., 2014) and focused on tasks such as encoding database schema information and user queries. The attention mechanisms (Vaswani et al., 2017) mapped user intent to schema elements, facilitating the alignment between queries and database schema. Early schema-linking methods in these neural models primarily identified items explicitly mentioned in the user's query. However, constructing a complete SQL statement often requires incorporating tables and fields that are not directly referenced in the query (Gan et al., 2021c). SQL statements are not a direct, word-for-word translation of natural language queries. During the era of neural text-to-SQL models, researchers introduced intermediate representations (IRs) to ease the generation of SQL statements (Guo et al., 2019; Wang et al., 2020; Yu et al., 2018b; Gan et al., 2021c). These IRs simplified subclauses such as JOIN ON, FROM, and GROUP BY, making them easier for models to generate while still preserving the overall SQL structure.

In modern approaches, schema-linking remains a crucial step, particularly in LLM-based methods. The objective is for LLMs to infer all the necessary fields and tables required to answer a query, even when they are not explicitly mentioned. Schema-linking can reduce input noise by filtering irrelevant schema elements (Talaei et al., 2024), significantly shortening the tokenized input length and enabling LLMs to focus on

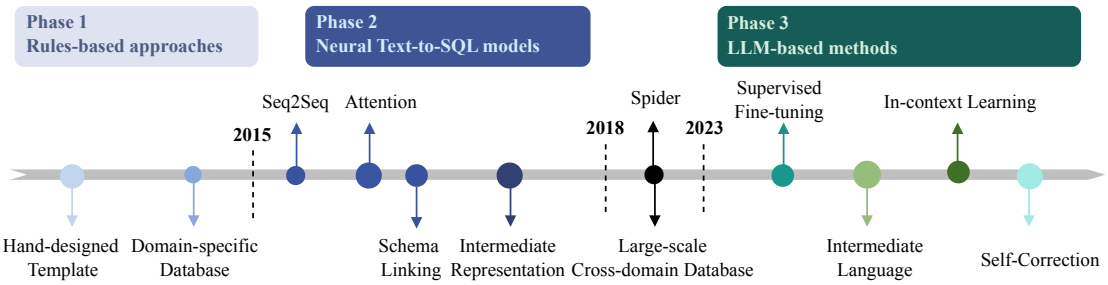

Figure 1: A brief description of the development trajectory of text-to-SQL technology, the introduction of Spider, and the rise of LLMs as a game changer, which have propelled text-to-SQL technology a significant step closer to enterprise-level applications.

information directly relevant to the query. However, recent research (Cao et al., 2024b) has shown that schema-linking can introduce errors if critical information is mistakenly excluded during the process. To address this challenge, we assume that reasoning occurs within pre-processed sub-tables, which limits the input schema length provided to the LLMs. However, LLMs still face difficulties in selecting precise field names from these sub-tables and extracting specific contextual values such as strings, dates, or other attributes from user queries or supplementary descriptions. In this context, schema-linking facilitates alignment between query elements and database components, thereby enhancing the LLMs' ability to generate accurate SQL statements.

In the era of LLM-based methods, further progress has been made toward generating IRs. For example, Eyal et al. (2023) and Wolfson et al. (2020) reformulated IRs into textual descriptions. These methods break down the execution steps of SQL queries into natural language descriptions, providing a detailed, step-by-step explanation of the SQL execution process. However, these representations primarily focus on SQL execution details and do not take into account the usability or accessibility needs of non-technical users. To address these limitations, our keyword representation abstracts the query intent from the original question and represents it using concise and intuitive keyword combinations. This representation is specifically designed to involve only simple and user-friendly components, such as field selection, numerical filtering, and time filtering, which makes it easier for ordinary users to access query-to-SQL processes and still provides the necessary level of abstraction for LLM inference. DIN-SQL, a representative of classical prompt-based methods, addresses the text-to-SQL task by breaking it into sub-tasks and designing specific prompting strategies for each sub-task. Although effective in some cases, this approach faces several challenges, including excessively long prompts, slow execution speeds, and high API call costs.

Furthermore, the linearized execution process inherent in these methods may lead to the accumulation of errors, which can ultimately mislead the generation of the final SQL statement. To mitigate these challenges, LLM-based methods often include a self-correction module. This module leverages the knowledge embedded in advanced LLM pre-training and contextual information from the text-to-SQL task to correct erroneous SQL predictions. Methods such as CodeS (Li et al., 2024) and RSL-SQL (Cao et al., 2024b) further enhance this approach by executing the generated SQL statements and providing the execution results as feedback to the LLMs, enabling iterative refinement of predictions. However, even with an extended workflow, current methods cannot completely eliminate prediction errors. Errors often arise when LLMs encounter unseen user queries. Although humans can easily resolve such issues through analogy, LLMs may fail to recognize the similarity between the new query and previously learned examples. Additionally, some queries may have multiple valid solutions, making it difficult for LLMs to identify the optimal response. For human users, however, making small adjustments to an incorrect prediction is often sufficient to produce a usable SQL statement. Our proposed keyword representation addresses these challenges by emphasizing usability and simplicity. Extensive user experiments have demonstrated the approach's user-friendly nature, showing that users can quickly master the syntax rules of the keyword language with just a few template examples. This enables non-expert users to efficiently interpret and correct prediction errors, making the text-to-SQL process more accessible and intuitive.

3 Datasets

This study constructs three core datasets to facilitate LLMs in learning keyword representation. The data collection and annotation processes are conducted by a team of six data science graduate students under the supervision of five database experts.

3.1 Model Fine-tuning Dataset

Keyword Representation is the core of our research methodology. To enable LLMs to learn the syntactic rules of keyword representation, we fine-tune the model using a large number of question-keyword pairs. The text-to-SQL English datasets from Spider and Bird serve as the initial corpus. Under the guidance of our database experts and in accordance with the translation rules for keyword syntax, we translate a significant number of question-keyword pairs, employing keyword query statements to perform search tasks. For the Cspider dataset (Min et al., 2019), we translate table names and field names and annotate them with corresponding keyword statements in the Chinese context. Upon completion of the annotation, we ensure the quality of the annotations through cross-checking, where different annotators' translations are verified for consistency and accuracy.

In addition, we are focused on addressing the query needs that are commonly encountered in business analytics, particularly those related to growth trend analysis, percentage calculations, and comparative analysis. However, current benchmark datasets widely used in the field do not specifically cover such types of queries. Using Spider and Bird as the foundational keyword training corpora, we expand the dataset by generating new types of question-keyword pairs with LLMs and artificial seed data. This process results in 9,023 Chinese samples and

16,630 English samples, which are used to fine-tune models such as GPT and Qwen to learn the syntactic rules of keyword representation.

3.2 Dataset For In-Context Learning

In-Context Learning (ICL) is the core method for all LLM-based text-to-SQL tasks, and high-quality exemplars are crucial for enabling LLMs to perform ICL (Gao et al., 2023). The first requirement is comprehensive query coverage, which should include both common query questions and domain-specific ones, such as those related to warehouse logistics, insurance data, and other specialized fields. The second requirement is the need for a systematic method to describe the dataset. We have identified 17 query types to describe the action instructions required for each keyword statement. By labeling the query types, we can create a more detailed example database, which also allows for more accurate example selection. Based on real-world databases and user queries, we manually annotate and clean 1,276 natural language questions, along with their corresponding keywords and query type labels. The number of example samples included under each query type is shown in Figure 2, of which a single question may involve multiple query types (please refer to Appendix E for the query type prediction task).

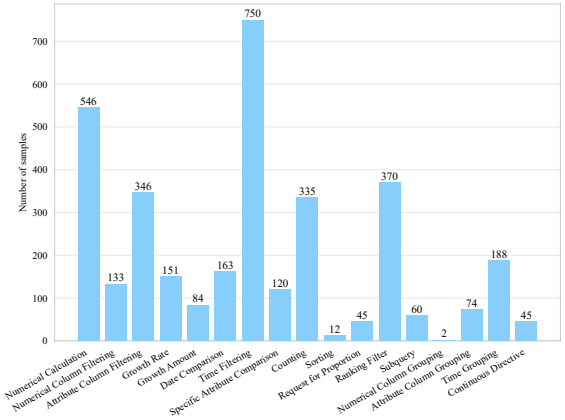

Figure 2: Exemplars for In-Context Learning.

3.3 Test Dataset

The existing Spider and Bird datasets do not fully meet the actual business requirements. To evaluate model performance, we design a test dataset that extends beyond the training data, presented as triples: natural language question-query types-keyword. A total of 200 question pairs are selected as the test set. The query type labels also facilitate fine-grained correction and analysis during error analysis of the prediction results.

The number of test question samples under each query type is shown in Figure 3. To better align with real-world business analysis scenarios, the test set places greater emphasis on time-related query types (such as time filtering, date comparison, growth rate and continuous instructions). Approximately half of the questions focus on numerical calculations (such as averages, summation, counting, and other related operations). The design of these questions is intended to comprehensively assess the model's ability to analyze real-world business data.

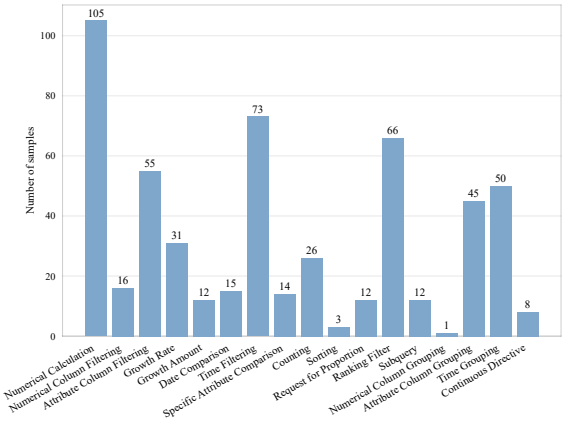

Figure 3: Test set for text-to-keyword task.

4 Methodology

We propose the text-to-keyword task, in which keyword representation serves as an intermediate language and acts as a central reference point for user interaction with the text-to-SQL system, the schematic diagram of the structure is shown in Figure 4. This enables non-technical users to independently assess whether the keyword query expression accurately reflects the intended query in the current iteration. In the text-to-keyword task, the given dataset consists of pairs of q_i and k_i,

$$\mathcal{F} = \{(q_i, k_i, \mathcal{D}_i)\}, \tag{1}$$

$$\Gamma = \{(q_i, k_i, \mathcal{D}_i, q_{t_i})\}, \tag{2}$$

where q_i refers to the natural language question and k_i refers to the corresponding keyword statement on the database $\mathcal{D}_i$. Besides, q_{t_i} refers to the query types for which we manually annotate the question and its corresponding keyword statement. Similar to the text-to-SQL task, this type of generative task, utilizing the context learning paradigm (Dong et al., 2024), can be formulated as:

$$\max_{S',\sigma} \mathbb{P}_{\mathcal{M}}\left(k^* \mid \sigma(q, \mathcal{D}, S')\right), \tag{3}$$

where $S' \subset \mathcal{F}$ represents the most relevant sample examples selected from the compiled example database $\mathcal{F}$. And $\sigma(\cdot,\cdot,\cdot)$ denotes the process of interpretation carried out by the LLM $\mathcal{M}$ based on the input data. The objective of in-context learning for text-to-keyword task is to maximize the probability that the LLM generates the correct target keyword statement k^* given the user query q and the associated database information D. This task involves two key stages: firstly, analyzing the user's query intent, which aligns with the semantic analysis step in text-to-SQL tasks. Secondly, describing the user's query intent using a keyword-based language, which leverages the LLM's generative capabilities. To achieve this, keyword grammar rules R_k must be embedded into the LLM acting as the translator. Both prompt engineering and supervised fine-tuning are feasible approaches to teach the LLM unseen rules during pre-training and enable it to generate translations based on these rules. We then evaluate the model's predictions by adopting the evaluation method based on intent-based metrics (Floratou et al., 2024), a relaxed evaluation standard which takes into account three factors: the original query intent, the generated keyword statement, and the execution results. The

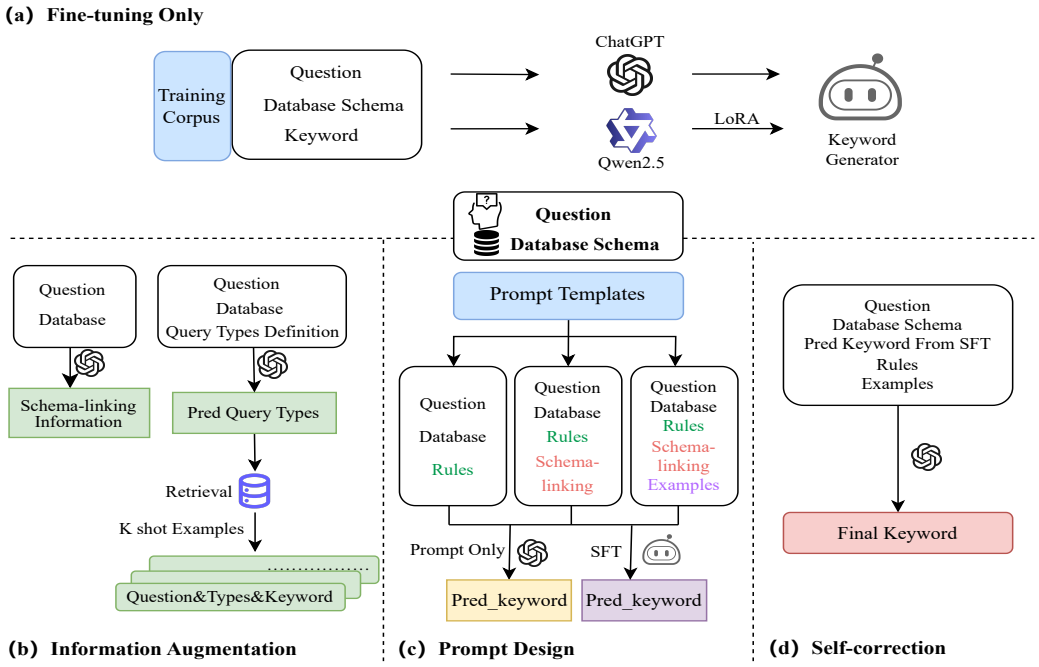

Figure 4: The illustration of our method is mainly divided into four modules. (a) Fine-tuning Only refers to fine-tuning LLMs using triplet corpus data. (b) Information Augmentation refers to deriving richer information starting from the current question. (c) Prompt Design refers to the organization of information in prompts for the text-to-keyword task. (d) Self-correction refers to using a closed-source LLM to correct the keyword predictions from the previous round.

final keyword utterance reflects the LLM's understanding of the user's query intent, and model performance is assessed based on this execution accuracy.

Our approach consists of four modules. We illustrate Figure 4 (a) through the fine-tuning method in keyword syntax rules embedding. For each new user query to be addressed, we need to invoke the LLM to initialize auxiliary information (Figure 4 (b)), schema-linking information and the predicted query types, from which we can retrieve examples information. This is then input into the keyword generator LLM to obtain the initial predicted keyword (Figure 4 (c)). Finally, we test closed-source LLMs to correct keyword predictions (Figure 4 (d)).

4.1 Keyword Syntax Rules Embedding

After pre-training, LLMs demonstrate rich prior knowledge and possess strong natural language understanding and reasoning capabilities. Due to the SQL-related content in their pre-training data, LLMs can typically generate approximate SQL translations for text-to-SQL tasks without the need for explicit rule specifications. However, they have never encountered a text-to-keyword translation task based on specific rules. To obtain keyword expressions in the desired format, we must consider embedding keyword syntax rules into the model.

1) Prompt-based methods If all rules are input through prompts, the challenge lies in how to explain the rules to the LLMs as concisely as possible. In addition to explanatory instruction statements, we have categorized the query types into 17 categories, with the total number of allowed keyword instructions across all types exceeding 100. If we design examples for every possible scenario and include many of these in the prompt, the input text for a single question becomes excessively lengthy, thus reducing the proportion of core information—namely, the current question and its database information within the text. Gao et al. (2023) and Chang and Fosler-Lussier (2023) have shown that overlong prompts may actually reduce performance. This is because irrelevant information in the input can distract or confuse the LLMs. We also verify this in our subsequent experiments. Overlong prompts increase the costs in terms of API calls and time required to answer each question.

2) Fine-tuning methods The LLM-based fine-tuning methods also show good results in the text-to-keyword task. Open-source LLMs, such as Qwen 2.5, Llama 3.1, and Chat-GLM4, can be fine-tuned with the question-schema-keyword triplet data to adapt to the keyword syntax rules. During the fine-tuning phase, LLMs can acquire rich corpus knowledge, including a comprehensive range of keyword phrases, query types, and so on, to help the model learn the proper usage of keyword components. High-quality question-keyword pairs are essential for supervised fine-tuning, as LLMs typically perform poorly when tasked with answering unseen questions from the fine-tuning corpus.

4.2 Information Augmentation

Existing research decomposes the text-to-SQL task into a series of interconnected sub-tasks, such as schema-linking, question classification, and other components. These sub-tasks help infer additional information, which ultimately aids in generating the final output SQL statement (Pourreza and Rafiei, 2023; Dong et al., 2023). Exemplar selection is crucial for ICL in LLMs. In a prompt-based approach, leveraging exemplars to embed task rules into LLMs may be the most effective strategy. Numerous text-to-SQL studies (Gao et al., 2023; Pourreza and Rafiei, 2023; Dong et al., 2023) demonstrate excellent performance with few-shot learning (Brown et al., 2020). To

address this, we define 17 distinct query types. Expressing the query intent of a single problem q may require multiple keyword sub-statements of different query types. Each query type maps to an SQL sub-statement or a highly integrated SQL statement template (e.g., growth rate, proportion calculation). We use the embedding model (Reimers and Gurevych, 2019) to convert the sample data (q_i, k_i, q_{t_i}) into embedding vectors, which are stored in a vector database as reliable question-query types-keyword examples. Similarity measures are then used to retrieve the most relevant example for the current query q. Previous work (Zhang et al., 2023; Gao et al., 2023) either computes the semantic similarity of the questions, the similarity of the target SQL queries, or considers both factors to rank and filter the samples. Whether it is question-to-SQL pairs or question-to-keyword pairs, in practical applications, we observe that the word order and choice of phrasing in the query can influence the results of semantic similarity-based retrieval. Specifically, the same query intent may be expressed in various textual formulations, which can lead to situations where the stored example cannot be found for the same query intent.

It is also observed that queries from different domains typically exhibit low semantic similarity, primarily due to the significant differences in tokens such as field names, values, and time. However, their target statements are often quite similar in terms of implementation. This implies that cross-domain query intents may have a similar target statement structure, but retrieving them based solely on the query may not be effective. If we instead begin by retrieving examples based on the target statement, we would need to predict an intermediate target statement and perform a secondary prediction using the LLM once the example is retrieved. Datasets like Spider and Bird provide a vast collection of training samples that can be used as an example database. However, selecting the most relevant example from over 10,000 samples each time consumes significant computational resources and results in slow response times. Keyword statements intuitively describe the set of operations corresponding to the query intent. This structural similarity should also be considered when selecting examples. However, calculating semantic similarity based on the question or SQL query itself does not yield good results. We instead label the query intent using predefined query types, and by comparing the similarity of query types, we can identify structurally similar keyword statements, where the sub-statements of the keywords can be equivalently viewed as sub-queries of the SQL statement. By utilizing query type labels q_t, we can retrieve samples that are cross-domain but structurally similar when performing text similarity-based retrieval.

4.3 Prediction Correction

1) LLM self-correction Existing works leverage more advanced LLMs (such as GPT-4, Claude3.5) to correct predicted SQL statements. In DIN-SQL, the improvement in zero-shot correction is limited, and the correction based on query result feedback does not offer effective control over task response time. In this study, we only utilize the strong reasoning capabilities of online models, in conjunction with relevant examples and keyword rules, to refine the predicted keyword statements. In our ablation experiments, we examine the ability of models such as GPT-4o, DeepSeek, and Qwen2.5-32B to correct keyword statements.

2) User verification The correction of LLM-generated statements is a critical aspect of the interaction between non-expert users and the text-to-SQL system, as it directly affects the system's practicality. Ordinary users need to view the data tables, including the current database schema, field names, and previews of the table contents.

LLMs inevitably introduce errors when predicting SQL or keywords, such as using incorrect field names, as illustrated in Case 1 of Table 1, or missing some advanced instructions, as illustrated in Case 2 of Table 1. The remaining keyword prediction error types are detailed in Appendix A. Users only need to verify whether the predicted keyword corresponds to the initial query. Using the keyword statement as an anchor point, users can independently perform corrections, thereby completing the full pipeline of the text-to-SQL task.

5 Experiments and Results

5.1 Experiment Settings

1) Dataset The evaluation dataset we designed includes two labels: golden query types and golden keywords, where the golden query type indicates the action instructions required to solve the query problem. We conduct extensive experiments on a set of 200 test samples to assess the practicality of the text-to-keyword task.

2) Evaluation metric The execution accuracy (EX) is used as the evaluation metric for different models. Floratou et al. (2024) introduced an evaluation metric with lenient acceptance thresholds, referred to as Intent-based Match, which is based on execution match. The predicted query may contain additional query items, or the original question may be ambiguous. In cases where the execution results are valid, such predictions are still considered acceptable. We also apply the Intent-based Match evaluation metric based on the execution results.

3) Models To ensure data privacy, many enterprises are reluctant to upload database schema information to the internet. Deploying a text-to-SQL system locally using open-source models is a viable solution. We evaluate the effectiveness of keyword statement generation on multiple open-source models, including the Qwen2.5 series (Yang et al., 2024; Hui et al., 2024) (Qwen2.5-7B-Instruct, Qwen2.5-14B-Instruct, Qwen2.5-Coder-14B-Instruct), llama3.1-8B (Dubey et al., 2024), and glm-4-9B-chat-hf (GLM et al., 2024). We fine-tune open-source LLMs using the 9,023 training samples mentioned earlier. Meanwhile, various closed-source LLMs demonstrate more advanced fundamental capabilities. By leveraging ICL to infer keyword syntax rules, we also compare the performance of GPT-4o (Hurst et al., 2024), GPT-4o-mini, GPT-3.5-turbo, and DeepSeek (Liu et al., 2024) in the text-to-keyword task.

4) Comparison methods We compare our approach with two SOTA ICL methods on the test dataset, namely DAIL-SQL (Gao et al., 2023) and DIN-SQL (Pourreza and Rafiei, 2023). These methods use the costly GPT-4 (Achiam et al., 2023) model to generate SQL queries, and we directly evaluate the acceptability of their execution results.

5.2 Main Results

Table 2 presents the acceptability of execution results for our methods compared with two competitive methods on our custom enterprise requirement dataset. Overall, using Qwen2.5-Coder-14B-Instruct and the method proposed in this paper, we achieve better performance on the test dataset compared to the existing SOTA benchmark methods, while our method requires lower API costs.

To explain this, firstly, additional exemplars can help the model effectively predict more complex query SQL statements.

Case 1	Car insurance	Error type
Question	查询连续两年赔偿额超过10000的保单的年份 Tell me the years in which there were consecutive two years where the total compensation amount exceeded 10000	
Gold	连续两年　总赔偿金额大于10000 2 consecutive years Total_Compensation_Amount > 10000	
Pred	赔偿额 > 10000 连续2年 2 consecutive years compensation > 10000	字段对齐 Field Alignment
Case 2	**Car insurance**	**Error type**
Question	按年统计保单量的增长量 Caculate the growth in the number of policy number by year.	
Gold	保单号的数量的年增长量 yearly growth amount of count Policy_Number	
Pred	每年　保单号的数量 yearly count Policy_Number	指令缺失 Instruction Missing

Table 1: The examples of predicted intermediate keywords that the user needs to correct.

Method	LLMs	Self-Correction	EX
DIN-SQL	GPT-4	GPT-4	80/200
DIN-SQL+NE	GPT-4	GPT-4	113/200
DAIL-SQL+NE (Spider)	GPT-4	GPT-4	85/200
DAIL-SQL+NE (Bird)	GPT-4	GPT-4	**135/200**
Xiaohui_1	GPT-3.5 fine-tuning	–	83/200
Xiaohui_2	GPT-3.5 fine-tuning	–	98/200
Xiaohui_3	GPT-3.5 fine-tuning	–	110/200
Xiaohui_4	GPT-4o-mini-2024-07-18 fine-tuning	–	**151/200**
Xiaohui_a	Qwen2.5-Coder-14B-Instruct	–	133/200
Xiaohui_b	Qwen2.5-Coder-14B-Instruct	DeepSeek	**160/200**

Table 2: Execution Accuracy results of Xiaohui, DIN-SQL and DAIL-SQL. NE denotes the introduction of new query-type examples (please refer to Appendix D for details).

The accuracy of DIN-SQL+NE is 16.5% higher than that of DIN-SQL (here NE denotes the introduction of new query-type examples). Therefore, it is important to select examples that are most relevant to the current question, which could involve the same SQL solving logic or problems from similar domains. Defining the distance between the question and the example is the key to example selection. Secondly, the quality of fine-tuning corpora directly impacts the model's performance. As seen from Xiaohui_3 and Xiaohui_4, even though the base model used in Xiaohui_3 (GPT-3.5) is superior to the base model in Xiaohui_4 (GPT-4o-mini), the model fine-tuned with cleaned high-quality data (Xiaohui_4) shows significant performance improvement, with a 20.5% increase in accuracy on the test set. Thirdly, the introduction of the self-correction mechanism significantly improved the model's execution accuracy. As shown in Xiaohui_a and Xiaohui_b, although both use the same base model (Qwen2.5-Coder-14B-Instruct) for fine-tuning, the performance of Xiaohui_b, which incorporates the self-correction mechanism (with DeepSeek), sees a significant improvement. The execution accuracy of Xiaohui_b is 80%, a 13.5% increase compared to Xiaohui_a, highlighting the crucial role of the self-correction mechanism in improving the accuracy of model results.

The error analysis results of different methods are shown in Figure 5. The results in Figure 5 (a) indicate that the frequency of errors in query types such as filtering (ranking filtering, time filtering, attribute column filtering), calculation (growth rate, growth amount, request for proportion, numerical calculation), and comparison (specific attribute comparison, time/date comparison) is high across all models. Relatively speaking, among the four models, DAIL-SQL performs better, with fewer errors in filtering and calculation query types compared to DIN-SQL. Furthermore, the DAIL-SQL+NE (Bird) model has fewer errors in filtering and calculation query types compared to DAIL-SQL+NE (Spider), largely because the query types in the Bird dataset are more diverse than those in the Spider dataset. Figure 5 (b) shows that compared to Xiaohui_a, the model with the self-correction module (Xiaohui_b) significantly reduces the error rates in field alignment and attribute column filtering query types. By combining both charts, we observe that the models obtained using the method in this paper perform differently from those obtained using existing prompt engineering methods in terms of several query error types. The models from our method perform significantly better in calculation, comparison, and time filtering query types compared to DIN-SQL and DAIL-SQL, while performing slightly worse in ranking filtering and field alignment query types.

5.3 Ablation Study

We conduct further ablation studies on prompt engineering to evaluate the effectiveness of prompt information, includ-

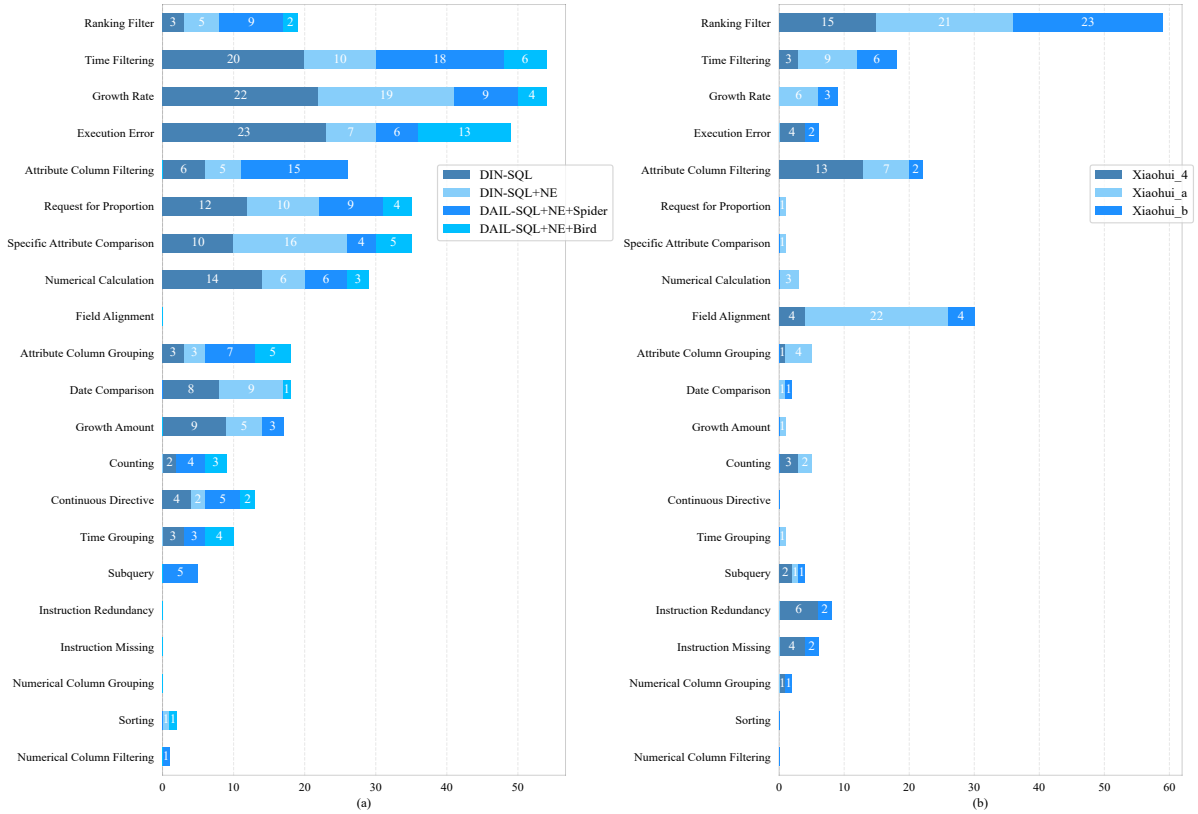

Figure 5: Error analysis of different methods on Text-to-Keyword task Dataset.

ing the selection of the text-to-keyword task base model, and the assessment of the self-correction module (refer to Appendix B.2). Additionally, we provide experimental analysis of keyword rules, schema-linking information, and exemplar selection in the Appendix C.

We evaluate current open-source LLMs by assessing their keyword prediction ability under a unified prompt template after fine-tuning on our text-to-keyword corpora. In contrast, online large models excel in language understanding and reasoning, indicating superior In-Context Learning capabilities. The additional error analysis can be found in Appendix B.1.

Method	LLMs	EX
Xiaohui_a	Qwen2.5-Coder-14B-Instruct	**133/200**
	Qwen2.5-7B-Instruct	0/200
	Qwen2.5-14B-Instruct	109/200
	GLM-4-9b-chat-hf	67/200
	Llama3.1-8B	89/200
	DeepSeek	118/200
	GPT-4o	**133/200**
	GPT-4o-mini	124/200

Table 3: Execution accuracy results of open-source LLMs and closed-source LLMs.

Notably, the results in Table 3 show that the output of Qwen2.5-7B-Instruct fails to adhere closely to the format specified in the prompt and differs significantly from the output of other models. Our method, Xiaohui_a, achieves the same accuracy as GPT-4o through a more cost-effective approach. More importantly, its predicted keywords are better suited for user verification.

6 Conclusion

In this research, we achieve interaction between non-technical users and the text-to-SQL system through keyword representation as an intermediate language. The results of our methods surpass the current SOTA methods by 12.5% on the test dataset. We gain comparable execution accuracy with less cost, while also allowing non-expert users to independently perform sentence correction. The Xiaohui_4 version, which is based on corpus cleaning, achieves a high execution accuracy of 75.5% and still has room for further improvement. Keyword-based sentences, as anchor points for human-machine interaction, provide a practical path for the implementation of text-to-SQL technology.

7 Limitations

Our research method currently has the potential for further optimization and improvement. For example, LLMs still struggle with understanding query types like ranking filtering, and we may need to address such issues by modifying keyword mapping rules, or by establishing more detailed rule explanations and more suitable example selections. Additionally, open-source LLMs currently face more difficulties when handling Chinese tasks, particularly in field alignment requirements. Many LLMs are unable to distinguish between Chinese synonyms (e.g., “订购数量” and “订单数量”, which can be translated as "order quantity" or "order amount). The time cost for running the entire process is not recorded. In example selection, further optimization can be made on the available

examples in each query type library, such as deduplication, supplementing the number of examples for each query type, or establishing cluster centers. Additionally, we do not conduct a quantitative analysis to assess the quality of our examples. When retrieving based on semantic similarity, no specific optimization is applied for Chinese texts. Regarding the datasets, we only conduct experiments on the text to keyword system in the Chinese environment. The experiments in the English environment are our upcoming work.

8 Ethical Considerations

Our datasets will be progressively made available, and the training data, example library, and test set used for fine-tuning have undergone rigorous review to ensure they do not contain politically sensitive or biased content. For data privacy reasons, our database information has been anonymized. The open-source and closed-source models employed in our study are publicly accessible online. The llama-factory is an open framework that we use for fine-tuning and inference, configured based on the inherent characteristics of open-sourced LLMs.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 1877–1901, Online. Curran Associates, Inc.

Ruisheng Cao, Fangyu Lei, Haoyuan Wu, Jixuan Chen, Yeqiao Fu, Hongcheng Gao, Xinzhuang Xiong, Hanchong Zhang, Wenjing Hu, Yuchen Mao, Tianbao Xie, Hongshen Xu, Danyang Zhang, Sida Wang, Ruoxi Sun, Pengcheng Yin, Caiming Xiong, Ansong Ni, Qian Liu, Victor Zhong, Lu Chen, Kai Yu, and Tao Yu. 2024a. Spider2-v: How far are multimodal agents from automating data science and engineering workflows? In *Advances in Neural Information Processing Systems*, pages 107703–107744, Vancouver, Canada. Curran Associates, Inc.

Zhenbiao Cao, Yuanlei Zheng, Zhihao Fan, Xiaojin Zhang, and Wei Chen. 2024b. Rsl-sql: Robust schema linking in text-to-sql generation. *arXiv preprint arXiv:2411.00073*.

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. In *NeurIPS 2023 Second Table Representation Learning Workshop*, Vancouver, Canada. Curran Associates, Inc.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2021. Structure-grounded pretraining for text-to-SQL. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1337–1350, Online. Association for Computational Linguistics.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minnesota, USA. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, Miami, Florida, USA. 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*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Ben Eyal, Moran Mahabi, Ophir Haroche, Amir Bachar, and Michael Elhadad. 2023. Semantic decomposition of question and SQL for text-to-SQL parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13629–13645, Singapore. Association for Computational Linguistics.

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. 2024. Nl2sql is a solved problem... not! In *14th Conference on Innovative Data Systems Research*, Hawaii, America. www.cidrdb.org.

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. 2021a. 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.

Yujian Gan, Xinyun Chen, and Matthew Purver. 2021b. Exploring underexplored limitations of cross-domain text-to-SQL generalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8926–8931, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R. Woodward, John Drake, and Qiaofu Zhang. 2021c. Natural SQL: Making SQL easier to infer from natural language specifications. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2030–2042, Punta Cana, Dominican Republic. 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*.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

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. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

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. *Proceedings of the ACM on Management of Data*, 2(3):1–28.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Ma Chenhao, Guoliang Li, Kevin 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*, pages 42330–42357, New Orleans, USA. Curran Associates, Inc.

Jinqing Lian, Xinyi Liu, Yingxia Shao, Yang Dong, Ming Wang, Zhang Wei, Tianqi Wan, Ming Dong, and Hailin Yan. 2024. Chatbi: Towards natural language to complex business intelligence sql. *arXiv preprint arXiv:2405.00527*.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.

Jianqiang Ma, Zeyu Yan, Shuai Pang, Yang Zhang, and Jianping Shen. 2020. Mention extraction and linking for SQL query generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6936–6942, Online. Association for Computational Linguistics.

Tanzim Mahmud, K. M. Azharul Hasan, Mahtab Ahmed, and Thwoi Hla Ching Chak. 2015. A rule based approach for nlp based query processing. In *2015 2nd International Conference on Electrical Information and Communication Technologies (EICT)*, pages 78–82, Khulna, Bangladesh. Institute of Electrical and Electronics Engineers.

Qingkai Min, Yuefeng Shi, and Yue Zhang. 2019. A pilot study for Chinese SQL semantic parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3652–3658, Hong Kong, China. Association for Computational Linguistics.

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*, pages 36339–36348, New Orleans, USA. Curran Associates, Inc.

Ge Qu, Jinyang Li, Bowen Li, Bowen Qin, Nan Huo, Chenhao Ma, and Reynold Cheng. 2024. Before generation, align it! a novel and effective strategy for mitigating hallucinations in text-to-SQL generation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5456–5471, Bangkok, Thailand. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks.

In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, page 3104–3112, Montreal, Canada. Curran Associates, Inc.

Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. 2024. Chess: Contextual harnessing for efficient sql synthesis. *arXiv preprint arXiv:2405.16755*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, page 6000–6010, California, USA. Curran Associates Inc.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, LinZheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. 2025. MAC-SQL: A multi-agent collaborative framework for text-to-SQL. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 540–557, Abu Dhabi, UAE. Association for Computational Linguistics.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198.

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. TypeSQL: Knowledge-based type-aware neural text-to-SQL generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 588–594, New Orleans, Louisiana. Association for Computational Linguistics.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018b. SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663, Brussels, Belgium. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018c. 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*, pages 3911–3921, Brussels, Belgium. 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 national conference on artificial intelligence*, pages 1050–1055, Menlo Park, CA. Association for the Advancement of Artificial Intelligence.

Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. 2023. ACT-SQL: In-context learning for text-to-SQL with automatically-generated chain-of-thought. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3501–3532, Singapore. Association for Computational Linguistics.

Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic evaluation for text-to-SQL with distilled test suites. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 396–411, Online. Association for the Advancement of Artificial Intelligence.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

A Label Definitions for Error Analysis

The predicted results are denoted as 'pred', while the golden keywords provide the reference for each sub-statement, and the selected column name denotes as 'XXX'. If the same sub-statement appears in 'pred' but with errors in details, it is defined as a query type error. As shown in Table 4, using incorrect column values (different from those actually stored in the database) results in no matches, causing an error in the attribute column filtering subquery, thus marked as an attribute column filtering error.

query types definition

- 增长率 (Growth Rate) : This typically refers to situations where the user explicitly requests the calculation of growth changes,which may include year-on-year growth rate, quarter-on-quarter growth rate, base-year growth rate, or both year-on-year and quarter-on-quarter growth rates. The user may also refer to specific time periods such as year, month, day, week, or quarter, like monthly growth rate, annual growth rate, quarterly growth rate, or weekly growth rate. Specific examples include: 'year-on-year growth rate of XXX'; 'month-on-month growth rate of XXX count'; 'monthly growth rate of unique count of XXX'.

- 增长量 (Growth Amount) : This usually refers to situations where the user explicitly requests the calculation of growth changes, which may include calculating year-on-year growth, quarter-on-quarter growth, base-year growth, or both year-on-year and quarter-on-quarter growth. It may be used together with 'Time Grouping' for calculations such as monthly growth, annual growth, quarterly growth, weekly growth, or daily growth. Specific examples include: 'year-on-year growth amount of XXX'; 'month-on-month growth amount of XXX count'; 'monthly growth amount of unique count of XXX'.

- 属性列筛选 (Attribute Column Filtering) : The fields involved in the user's query are of string type, and the values mentioned in the columns require string matching operations for the query intent.

- 数值列筛选 (Numerical Column Filtering) : The user's query involves numerical fields and specific values, requiring comparison or filtering operations based on the numerical values for the query intent.

- 时间筛选 (Time Filtering) : The user's query involves timestamp fields, and filtering or matching operations are performed on time periods such as days, weeks, months, quarters, or years.

- 排名筛选 (Ranking Filter) : The query intent requires directly ranking the values in a numerical column, and includes three types of usage intentions: 1. Perform grouping and summing operations (included in numerical calculation directives) on the numerical column before ranking and filtering, such as 'top 1 sum XXX'; 2. Rank the quantity after a 'count' operation, such as 'top 1 count XXX'; 3. Directly rank the numerical column without needing summing operations, such as 'top n XXX'.

- 属性列分组 (Attribute Column Grouping) : The user's question involves non-numerical columns or time columns. This directive is not used independently. Typically, the content of these attribute columns consists of categorical variables that need to be grouped and listed before performing other directive actions.

- 数值列分组 (Numerical Column Grouping) : The user's question involves numerical fields. When the intent to perform grouping statistics on a numerical column is recognized, it triggers a numerical column grouping operation directive.

- 时间分组 (Time Grouping) : Grouping operations based on dates for time-related fields involving timestamps. This is usually used in combination with other operations.

- 子查询 (Subquery) : Nested operations may be required. A subquery is a set of specific filtering conditions. A subquery may involve exclusion subqueries (EXCEPT), numerical calculation and filtering—referred to as an 'aggregated subquery' (for example: a numerical column greater than the average of that numerical column)—and selection conditions (subquery affiliation, selecting categories that meet the subcriteria), among other filtering conditions. The subquery must be executed before proceeding with subsequent directive actions.

- 排序 (Sorting) : The user requests sorting of numerical columns from high to low, or from large to small, or vice versa. This requires executing commands for ascending or descending order. Competitive ranking and dense ranking can be considered as types of sorting methods, though they differ from regular sorting. These two rankings are not filtering actions, so they are classified as sorting operations.

- 求占比 (Request for Proportion) : The user explicitly requests an operation to calculate the proportion. This is generally after a grouping instruction or for a specific group that has been filtered, to calculate the proportion or percentage of each group. It might follow instructions like time filtering and time grouping, or attribute column grouping, and requires executing the 'request for proportion' operation. Specific example: 'proportion of XXX'; combined with counting operations, 'proportion of count XXX'.

- 具体属性对比 (Specific Attribute Comparison) : The user has declared the intention to make a comparison (vs) in the question, generally comparing two or more 'attribute column filter' conditions, often used in conjunction with other directive actions. Specific examples: Attribute column filter1 vs Attribute column filter2; and comparison with the entire set 'Attribute column filter1 vs all'.

- 时间日期对比 (Date Comparison) : The user has expressed an intent to make a comparison (vs) and has provided two or more specific times. This action is typically used in conjunction with the 'Time Filtering' command. Specific usage: 1. Comparison between two time periods: 'Time filter1 vs Time filter2'; 2. Comparison with the entire time range: 'Time filter1 vs all'.

- 计数 (Counting) : Calculating the number of samples under the selected field, which may be: 'count XXX', 'unique count XXX'. Special considerations: 1. Does

Case 1		Error Type
Question	肥胖水平为二级超重的条件下，男性和女性的数量分别是多少? What is the count of males and females for individuals who's obesity level contains Class 2 Overweight?	
Gold	肥胖水平包含"二级超重" 按性别统计 性别的数量 count Gender Obesity_Level contains 'Class 2 Overweight'by Gender	
Pred	肥胖水平包含"2级超重" 按性别统计 性别的数量 count Gender Obesity_Level contains 'class 2 overweight'by Gender	属性列筛选 Attribute Column Filtering

Table 4: The error example of "Attribute Column Filtering" type.

the current question require grouping before counting? 2. Does the current question require using distinct counting (unique count XXX)?

- 数值计算 (Numerical Calculation) : The user's question involves numerical fields, and the query intent involves numerical calculation directives such as finding the maximum, minimum, average, sum, total, standard deviation and variance. When grouping statistics are involved (whether it's by time, attribute column, or numerical column), it may trigger a summing (aggregation) numerical calculation operation. The specific issue should be analyzed on a case-by-case basis.

- 连续指令 (Continuous Directive) : The user's question involves timestamp fields, specifically for filtering over a continuous period such as X consecutive days, X consecutive weeks, X consecutive months, X consecutive quarters, X consecutive years. This requires the use of the continuous directive.

non–query types definition

- 字段对齐 (Field alignment) : The field names used in the pred do not correspond to those found in the schema based on the question, and fields that do not exist in the schema used.

- 执行错误 (Execution error) : The pred is an irrelevant translation of the original question, or the LLM did not perform keyword translation and simply restated the content of the question. Alternatively, the pred does not follow the syntax of the keywords and is not recognized by the system during execution.

- 指令冗余 (Instruction redundancy) : The pred includes extra filtering targets that are not present in the golden statement, which causes the execution result to deviate from the original query intent, and is marked as instruction redundancy.

- 指令缺失 (Instruction missing) : The pred lacks necessary filtering instructions that are present in the golden statement.

B Supplementary Ablation Experiments

We investigate the error analysis of selecting the optimal local LLMs and examine the role of self-correction in enhancing prediction accuracy.

B.1 Error analysis of Selecting the Optimal Local LLMs

As shown by Figure 6, the error types in the keyword predictions of these LLMs, "Ranking Filtering" is difficult for both local and online models. For example, when a retail company needs to find the top three regions by sales amount, it must first calculate the total sales amount for each region and then apply ranking filtering. It is not simply about filtering the top three sales amounts (without aggregation) and then outputting the corresponding regions. Except for a few errors in the "top n XXX", most ranking filtering requires combining other instructions, such as "top n sum XXX" or "top n count XXX". Except for Xiaohui_a, the performance of the other local LLMs is inferior to that of the online models. Even for the same Qwen2.5-14b model, different versions have varying abilities to learn keyword syntax. Qwen2.5-14b-Instruct performs worse than Qwen2.5-Coder-14b-Instruct on the text-to-keyword task.

B.2 Self-correction

Self-correction is a crucial component of prompt-based research, such as DAIL-SQL (Gao et al., 2023) and DIN-SQL (Pourreza and Rafiei, 2023), significantly enhancing execution accuracy. We also investigate the ability of LLMs to iteratively refine keyword predictions. By adjusting the prompts, we enable the LLMs to incorporate rules, examples, and prior keyword predictions for improved accuracy. The intermediate keyword prediction is provided by Xiaohui_a.

Method	LLMs	Predicted Information	EX
Xiaohui_b	DeepSeek	Xiaohui_a keyword	**160/200**
	GPT-4o	Xiaohui_a keyword	150/200
	GPT-4o	Golden keyword	**166/200**
	GPT-4o	Golden keyword + schema-linking	143/200

Table 5: The ability of open-source LLMs to correct keyword predictions.

DeepSeek outperforms GPT-4o in refining local keyword predictions, with an accuracy improvement of 13% compared to Xiaohui_a, possibly because its Chinese-language corpus is of higher quality, leading to better performance in Chinese keyword translation tasks. Additionally, we examine the ability of online large models to verify golden-standard keywords. Correcting the golden keywords generates more erroneous statements, with an accuracy of 166/200, which marks the performance upper limit of our current correction method. The additional introduction of schema-linking information, however, makes GPT-4o's correction results worse. Cao et al. (2024b) have pointed out that the downside of schema-linking is the introduction of more noise, leading to the generation of more redundant predicted keywords. From the Figure 7, this is indeed the case. When GPT-4o corrects the golden keyword without using schema-linking information, it does

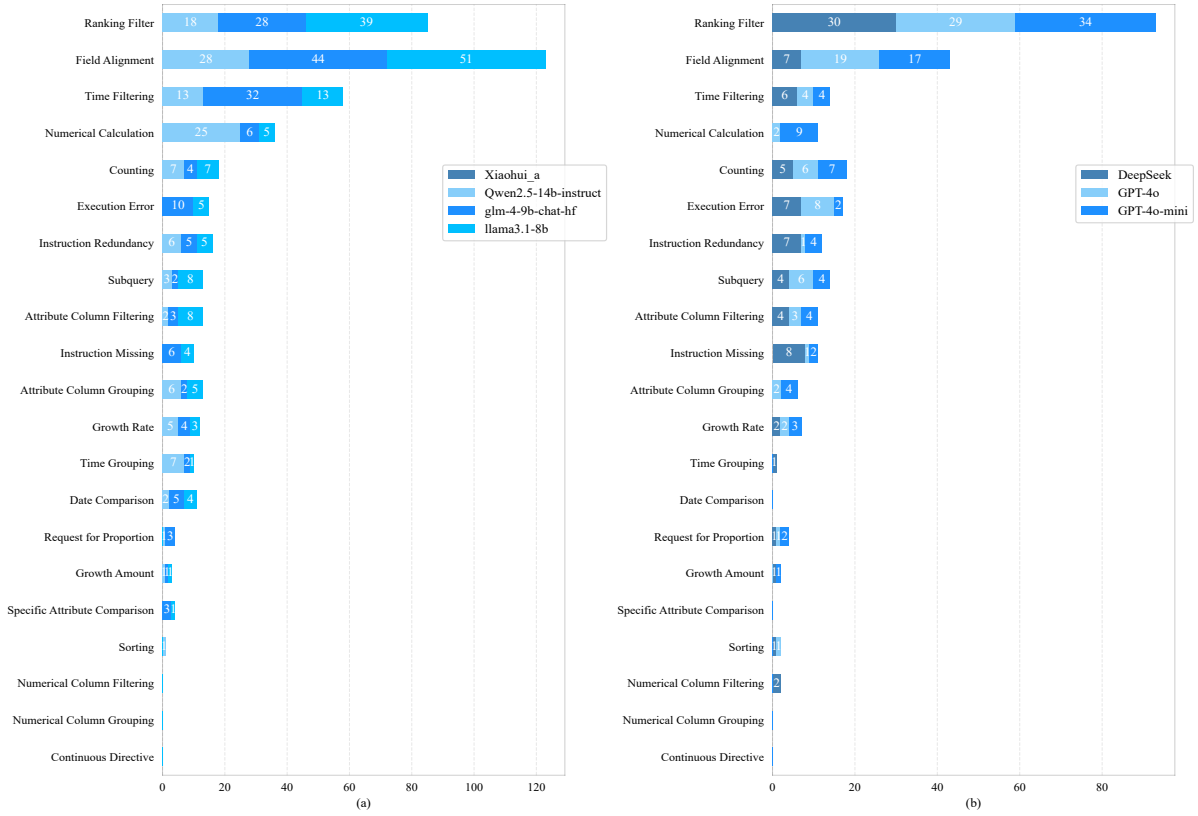

Figure 6: Error analysis for selecting the optimal local LLMs, and comparing to closed-source LLMs

not even make field alignment errors, and there are few types of instruction redundancy, instruction missing, or execution errors.

C Supplementary Experiments Conducted for Information Augmentation

Furthermore, we study the benefits of the information augmentation module presented in Figure 4 for the text-to-keyword task, and conduct ablation experiments on Rules, schema-linking information, and Exemplars selection.

C.1 Rules

First, we study the impact of rule explanations on the open-source model Qwen2.5-Coder-14b-instruct and compare it with the DeepSeek model that only uses context learning. We set up experiments with only text definitions (only-definition), currently allowed instructions (allowance), and fixed examples for the current query type (query type definition-example-CoT tuple). The results are shown in Table 6. Xiaohui_a is our fine-tuned Qwen2.5 model. As can be seen, with deeper rule explanations, the performance of the LLM improved further. The local 14B model is fine-tuned with keyword corpora, but there is still a gap compared to the online large models in terms of context learning capabilities.

C.2 Schema-linking Information

Second, we study the impact of schema-linking information on the text-to-keyword task. In this study, we select rule information by removing the fixed examples from each query type and only retaining the basic definitions and allowed instructions. The prediction of query types and schema-linking information adds steps to the entire text-to-keyword system, which increases costs, particularly in terms of time. We understand that enterprises also have limitations on the runtime of prediction systems. The number of interactions with the LLM is positively correlated with the system's final performance, but it must be balanced against runtime and other costs.

Regarding the direction of information acquisition in Table 7, 'forward' indicates that the schema-linking information is inferred from exemplars, the current question and schema and rules to determine the required field names for the current question. Since the keyword phrases in the exemplars have already selected the necessary fields for their respective questions, we expect these exemplars to assist in inferring the current schema-linking information. In contrast, 'reverse' means that this information is derived by backtracking from the golden keyword annotations. The online model DeepSeek further supplements the inference chain (Chain of Thought, CoT) for schema linking.

According to Table 7, the DeepSeek model, lacking guidance on specific query types, fails to generate keyword phrases when reasoning with forward information, with most outputs being SQL statements. Similarly, DeepSeek performs poorly when using reverse information inference. In contrast, when using a fine-tuned Qwen2.5 model for prediction, providing only the selected columns yielded inferior results compared to the other three configurations. However, when using a combination of CoT text and the selected columns as schema-linking information in the forward setting, the performance was comparable to that obtained with reverse information. This result aligns with the accuracy of our ultimately selected (Rules-exemplars) prompt template. Nonetheless, the current method tends to produce more errors in field alignment, which makes corrections more challenging. In comparison, our final

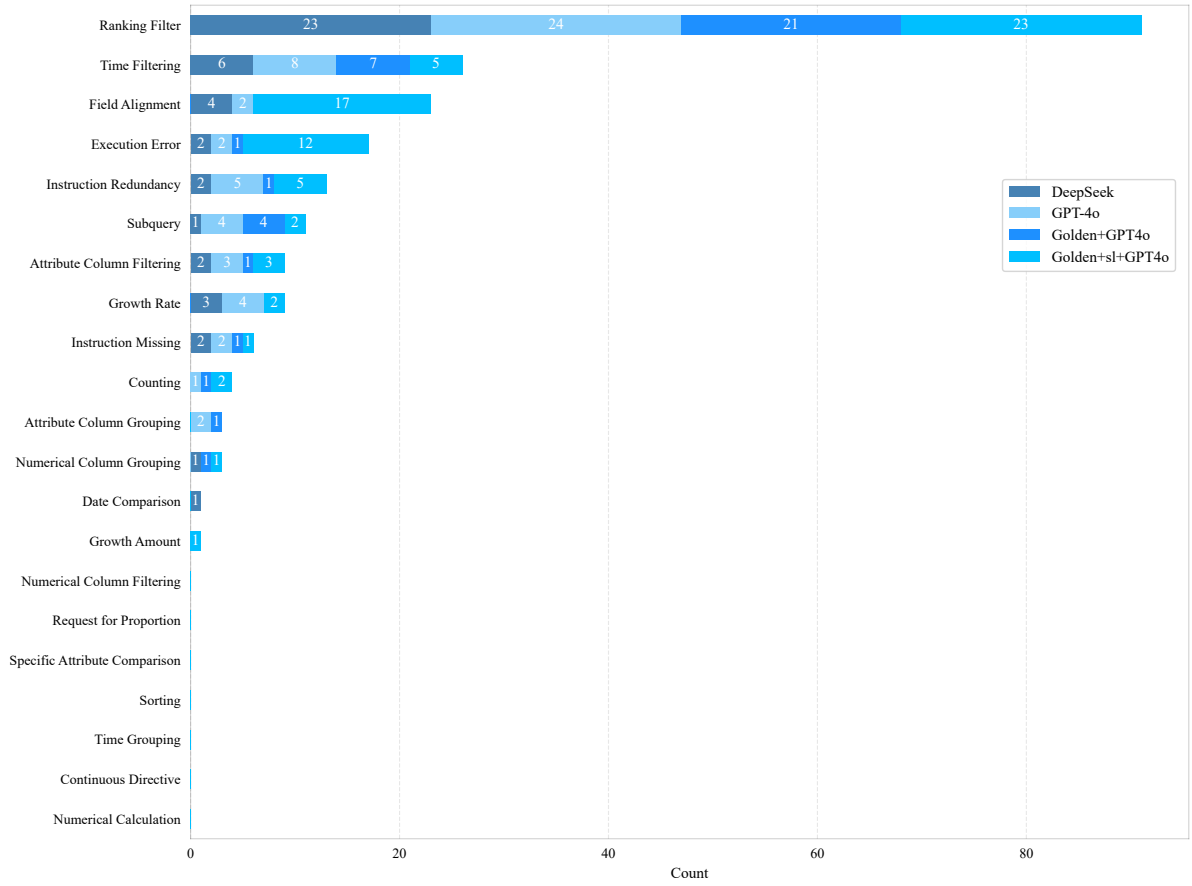

Figure 7: Error analysis of the self-correction results performed by closed-source LLMs.

Rules content	LLMs	EX
only-definition	Xiaohui_a	95/200
definition+allowance	Xiaohui_a	103/200
definition+allowance+fixed example	Xiaohui_a	107/200
definition+allowance+fixed example	DeepSeek	**118/200**

Table 6: Ablation study on the selection of rules content.

Schema-linking information	LLMs	Information acquisition	EX
CoT+selected columns	DeepSeek	forward	0/200
CoT+selected columns	Xiaohui_a	forword	**133/200**
selected columns	Xiaohui_a	forword	117/200
CoT+selected columns	DeepSeek	reverse	1/200
CoT+selected columns	Xiaohui_a	reverse	131/200
selected columns	Xiaohui_a	reverse	131/200

Table 7: Ablation study on the selection of schema-linking information.

method, even when predictions are incorrect, is easier for users to correct and is closer to meeting an acceptable threshold. Moreover, the exemplars also serve to guide the output format of LLMs.

C.3 Exemplars Selection

Third, we examine the impact of exemplar selection on the text-to-keyword task. Our investigation is divided into two main parts. The first part involves retrieving exemplars from the complete set based on similarity, which is further split into two aspects: one based on the question and the other on the query type. The second part entails building a vector database of exemplars for each query type and conducting retrieval by query type to study the effect of the number of exemplars on the performance of LLMs. Here, the number of exemplars (denoted as N) is constrained to be $N \leq m$, $m = 3, 6, 9$.

We compare the performance of Xiaohui_a and DeepSeek, representing local and online LLMs, respectively. As shown

in Table 8, exemplars retrieved from the entire set (with all cases set to 9 exemplars) are less effective than those obtained through query type classification, and even less effective than exemplars retrieved from a query type-specific vector database. The DeepSeek model achieves its highest accuracy with $N \leq 6$, whereas Xiaohui_a performs better when $N \leq 9$. This disparity is related to the quality of exemplar selection, an aspect that is insufficiently evaluated in our study. Additionally, we find that after converting Chinese tokens into embedding vectors, retrieval based on text semantic similarity does not always select the most relevant exemplars. As illustrated in Table 9, Chinese embedding models still require continuous improvement.

C.4 Full Information Prediction

Finally, we evaluate the model's ability to predict using all available information—including rules (definition + allowance), exemplars (9-shot examples utilizing categorized databases by query type), and schema-linking information (reasoning chain-of-thought and selected columns), as shown in Table 10, and conduct an error analysis, as shown in Figure 8.

The results in Figure 8 show that ranking-based filtering and field alignment are error types where all three methods perform poorly. Our method produces fewer execution errors, whereas both DeepSeek and GPT-4o tend to predict keywords in the format of table_name.column_name, which leads to execution errors on keywords. The schema-linking information is originally intended to provide the LLMs with more detailed analytical insights, but its output do not conform to the format specified in the examples. It is likely that GPT-4o's superior contextual understanding enabled it to achieve the best prediction results with longer, richer prompts.

However, in our practical application, the combination of Rules and exemplars already yields an accuracy of 133/200, and the predicted keywords are easy for users to correct with only minimal modifications. In our approach, the schema-linking task represents additional time and overhead, but it does not lead to a significant improvement in the overall performance of the system.

D Explanation of Additional New Exemplars (NE)

The research results based on DAIL-SQL indicate that GPT-4 level LLMs demonstrate exceptional learning capabilities when learning Question-SQL examples. However, since the examples in Din-SQL are written based on the Spider dataset, they fail to cover certain question types present in this study's dataset, such as comparison-type questions, growth rate calculations, continuous time analysis, and proportion calculations. To address this limitation, we design and add examples specifically targeting these question types, ensuring they follow the standardized format of Din-SQL. Additionally, we provide DAIL-SQL with new optional examples to further enhance the diversity and applicability of the dataset. We refer to these newly added examples as additional new examples, denoted as NE.

D.1 New Examples

Q: "Compare the average sales amounts for the years 2017 and 2018."
Schema_links: [product_usage_data.sales_amount, product_usage_data.date]
SQL:SELECT AVG(CASE WHEN strftime('%Y', date) = '2017' THEN sales_amount END) AS average_sales_2017, AVG(CASE WHEN strftime('%Y', date) = '2018' THEN sales_amount END) AS average_sales_2018 FROM product_usage_data.

Q: "Find the top 3 products by sales amount in 'huabei' and 'huadong' regions"
Schema_links: [SalesData.region, SalesData.sales_amount, SalesData.product_name, huabei, huadong]
SQL: with AggregatedSales as (select region, product_name, sum(sales_amount) as total_sales from SalesData where region in ('huabei', 'huadong') group by region, product_name), RankedSales as (select a.region, a.product_name, a.total_sales, count(b.total_sales) as rank from AggregatedSales a left join AggregatedSales b on a.region = b.region and a.total_sales < b.total_sales group by a.region, a.product_name) select region, product_name, total_sales as sales_amount from RankedSales where rank < 3 order by region, total_sales desc.

Q: "Calculate the annual sales growth"
Schema_links: [SalesData.sales_quantity, SalesData.date]
SQL: with yearly_sales as (select cast(strftime('%Y', date) as integer) as year, sum(sales_quantity) as total_sales_quantity from SalesData group by year) select t1.year as year, t1.total_sales_quantity - t2.total_sales_quantity as sales_growth from yearly_sales t1 left join yearly_sales t2 on t1.year = t2.year + 1 order by t1.year.

Q: "Find product categories with consecutive 2 years of sales amount exceeding 600000"
Schema_links: [SalesData.product_category, SalesData.sales_amount, SalesData.date]
SQL: select product_category from (select product_category, sum(sales_amount) as total_sales_amount, strftime('%Y', date) as year from SalesData group by product_category, year having total_sales_amount > 600000) as subquery group by product_category having count(year) >= 3.

Q: "Calculate the sales quantity proportion for each product category"
Schema_links: [SalesData.product_category, SalesData.sales_quantity]
SQL: select product_category, sales_quantity * 1.0 / total_sales_quantity as sales_proportion from (select product_category, sum(sales_quantity) as sales_quantity, (select sum(sales_quantity) from SalesData) as total_sales_quantity from SalesData group by product_category) as subquery order by product_category.

D.2 Classification Prompts

Q: "Compare the average sales in 2017 and 2018"
schema_links: [ProductTrialData.SalesAmount, ProductTrialData.Date]
A: Let's think step by step. The SQL query for the question "Compare the average sales in 2017 and 2018" needs these tables = [ProductTrialData], so we don't need JOIN.
Plus, it doesn't require nested queries with (INTERSECT, UNION, EXCEPT, IN, NOT IN), and we need the answer to the questions = [""]. So, we don't need JOIN and don't need nested queries, then the SQL query can be classified as "EASY".
Label: "EASY"

Q: "Find the top 3 products by sales amount in the 'huabei' and 'huadong' regions"

Exemplars selection	LLM	Retrieved from	EX
questions	DeepSeek	complete collection of questions	126/200
questions	Xiaohui_a	complete collection of questions	119/200
query types	DeepSeek	complete collection of query types	130/200
query types	Xiaohui_a	complete collection of query types	121/200
3-shot	DeepSeek	categorized databases of query types	126/200
3-shot	Xiaohui_a	categorized databases of query types	119/200
6-shot	DeepSeek	categorized databases of query types	**134/200**
6-shot	Xiaohui_a	categorized databases of query types	131/200
9-shot	DeepSeek	categorized databases of query types	133/200
9-shot	Xiaohui_a	categorized databases of query types	133/200

Table 8: Ablation study on the selection of schema-linking information.

Keyword	Question	Golden query types (Using Chinese for retrieval in fact)	Distance
2023年 每年的最后3个月个人支付金额 between 2023/10/1 and 2023/12/31 yearly personal_payment_amount	23年最后3个月订单的个人付款额是多少? Tell me the years in which there were consecutive two years where the total compensation amount exceeded 10000	时间筛选 时间分组 Time Filtering, Time Grouping	0.007959146
时间 花费的总和 time sum spending	分析不同时间段的花费变化趋势 Analyze the trend of expenditure changes over different time periods.	数值计算 时间分组 Numerical Calculation, Time Grouping	0.007959146
每年 商品访客成本的总和 yearly sum cost_per_product_visitor	每年的商品访客成本大概是多少? What is the approximate product visitor cost per year?	数值计算 时间分组 Numerical Calculation, Time Grouping	0.007959146
评价版本的去重数量的周增长率 2023年 按周统计 2023 by week weekly growth rate of unique count evaluation_version	2023年，评价版本的去重数量周增长率是多少? In 2023, what is the weekly growth rate of the deduplication count for the evaluation version?	增长率 时间筛选 时间分组 Growth Rate, Time Filtering, Time Grouping	0.085216679
评价版本的去重数量的月增长率 2023年 按月统计 2023 by month monthly growth rate of unique count evaluation_version	2022年评价版本的去重数量的月增长率 In 2023, what is the monthly growth rate of the deduplication count for the evaluation version?	增长率 时间筛选 时间分组 Growth Rate, Time Filtering, Time Grouping	0.085216679

Table 9: Example similarity ranking of the user questions, based on query type text retrieval.

LLMs	EX
DeepSeek	136/200
GPT-4o	**142/200**
Xiaohui_a	137/200

Table 10: Prediction results using all available information.

schema_links: [ProductTrialData.Region, ProductTrialData.SalesAmount, ProductTrialData.ProductName, huabei, huadong]
A: Let's think step by step. The SQL query for the question "Find the top 3 products by sales amount in the 'huabei' and 'huadong' regions" needs these tables = [ProductTrialData], so we don't need JOIN.
Plus, it requires nested queries with (INTERSECT, UNION, EXCEPT, IN, NOT IN) and window functions (ROW_NUMBER), and we need the answer to the questions = ["Find the top 3 products by sales in each region"]. So, we don't need JOIN and need nested queries, then the SQL query can be classified as "NESTED".
Label: "NESTED"

Q: "Calculate the annual sales growth"
schema_links: [ProductTrialData.SalesQuantity, ProductTrialData.Date]
A: Let's think step by step. The SQL query for the question "Calculate the annual sales growth" needs these tables = [ProductTrialData], so we don't need multiple tables, but we need a self JOIN.
Plus, it requires nested queries with (INTERSECT, UNION, EXCEPT, IN, NOT IN), and we need the answer to the questions = ["Calculate total sales for each year and then find the growth between consecutive years"]. So, we need JOIN and need nested queries, then the SQL query can be classified as "NESTED".
Label: "NESTED"

Q: "Find the product categories that exceeded 600,000 in sales for 2 consecutive years"
schema_links: [ProductTrialData.Category, ProductTrialData.SalesAmount, ProductTrialData.Date]
A: Let's think step by step. The SQL query for the question "Find the product categories that exceeded 600,000 in sales for 2 consecutive years" needs these tables = [ProductTrialData], so we don't need JOIN.
Plus, it requires nested queries with (INTERSECT, UNION, EXCEPT, IN, NOT IN), and we need the answer to the questions = ["Find categories with sales exceeding 600,000 for at least two consecutive years"]. So, we don't need JOIN and need nested queries, then the SQL query can be classified as "NESTED".
Label: "NESTED"

Q: "Calculate the sales proportion for each category"
schema_links: [ProductTrialData.Category, ProductTrialData.SalesQuantity]
A: Let's think step by step. The SQL query for the question "Calculate the sales proportion for each category" needs these tables = [ProductTrialData], so we don't need JOIN.
Plus, it requires nested queries with (INTERSECT, UNION, EXCEPT, IN, NOT IN), and we need the answer to the ques-

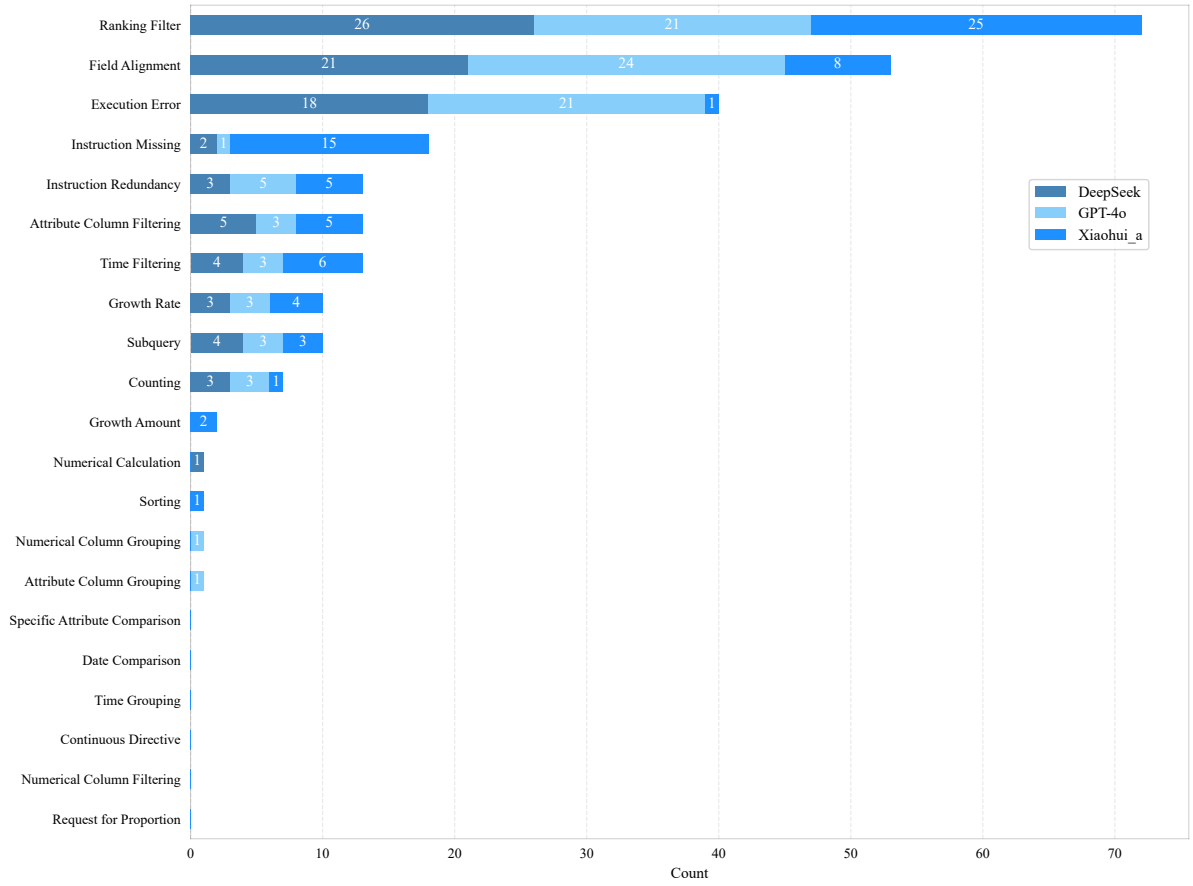

Figure 8: Error analysis of using all available information.

tions = ["Calculate the total sales amount for all categories"]. So, we do not need JOIN and need nested queries, then the SQL query can be classified as "NESTED".
Label: "NESTED"

E Query Types Prediction

Query types prediction marks the starting point of our text-to-keyword task. It requires the LLMs to analyze which action instructions are needed to resolve the current problem, serving as a form of problem decomposition.

However, achieving high accuracy in query type prediction is not our current priority, as the primary bottleneck in our workflow remains the subsequent keyword generation step. LLMs, especially smaller, locally deployed models, do not learn or interpret examples as intuitively as expected. Ideally, they should be able to grasp the usage of various sub-statements from the examples (for instance, understanding the contexts for instructions like repeated count and unique count), but such nuances are difficult to capture solely through question-to-keyword pairs. While examples can indeed help LLMs learn the rules, their effective use still demands considerable engineering effort, making the optimal utilization of examples a matter of prompt engineering. Consequently, in this study, we do not consider query types prediction to be particularly critical.

On the other hand, by classifying queries according to their types and building dedicated example databases, a single question can belong to multiple repositories. For instance, as shown in Table 9, if a question's golden query type labels are A, B, and C, we can retrieve the corresponding example from the A, B, and C databases, storing it as a question–query types–keyword triple. Moreover, we can obtain the schema for the example, which makes it easier for local LLMs to learn from schema-enriched examples. Even if the predicted query types for this question are A, F, and G, we can still retrieve it from the A database.

We use the definition of query types along with fixed examples corresponding to each type to illustrate the query types prediction task. Each query type has up to four question-query types examples. For token length considerations, a small number of examples are supplemented with a schema. The current question and schema are then input to predict the query type for the given question. We test the query type prediction task using GPT-4o and GPT-4o-mini. Let

$$
\begin{aligned}
n_1 &= \sum_{i=1}^{200} \mathbb{I}\left(|G_i \cap P_i| \neq \emptyset\right) \\
n_2 &= \sum_{i=1}^{200} \mathbb{I}\left(|P_i \setminus G_i| = \emptyset\right) \\
n_3 &= \sum_{i=1}^{200} \mathbb{I}\left(|G_i \setminus P_i| \neq \emptyset\right) \\
n_4 &= \sum_{i=1}^{200} \mathbb{I}\left(|P_i \setminus G_i| = \emptyset\right) \cdot \mathbb{I}\left(|G_i \setminus P_i| = \emptyset\right) \\
&= \sum_{i=1}^{200} \mathbb{I}\left(|P_i = G_i|\right)
\end{aligned}
\tag{4}
$$

where G_i represents the set of ground truth query types for the ith sample, and P_i represents the set of predicted query types for the ith sample. And n_1 represents the number of cases where at least one query type in the predicted query types matches the golden (ground truth) query types. n_2 indicates that the predicted query types do not exceed the scope of the golden set, meaning no extra query types are present. n_3 reflects cases where the predicted query types miss one or more query types that appear in the golden set. n_4 signifies perfect predictions, there are neither missing nor extra query types, meaning the prediction is entirely correct. We make predictions for the query types of 200 samples in the test set. Specific results are shown in Table 11.

LLMs	n_1	n_2	n_3	n_4
GPT-4o	**191**	115	86	80
GPT-4o-mini	187	75	117	47

Table 11: Test results for query type prediction.

Table 11 depicts that GPT-4o correctly predicts 191 out of the 200 samples, with 115 predictions showing no redundant instructions, 86 without omissions, and 80 that perfectly matched the golden labels. The effectiveness of LLMs predictions for query type classification depends on various factors, including the definition of query types, the number of examples, and whether the input schema is provided. Conducting extensive experiments is essential to fully understand and optimize these influences. Using a locally deployed open-source LLM for this specific prediction task is also a great implementation approach.

It is evident that, for the next step in examples selection, improvements are needed in areas such as the text embedding model, similarity calculation methods, and the organization of the example library. However, what we ultimately need is keyword prediction that is easy for users to correct. Fine-tuning has allowed the LLMs to initially learn keyword syntax rules, and examples are used to guide and correct predictions in later stages. Unlike traditional step-by-step SQL generation, which can accumulate errors linearly, we start with a step that does not require extremely high accuracy, achieving better keyword prediction results.