Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation

Dawei Gao*
Alibaba Group
gaodawei.gdw@alibaba-inc.com

Haibin Wang*
Alibaba Group
binke.whb@alibaba-inc.com

Yaliang Li
Alibaba Group
yaliang.li@alibaba-inc.com

Xiuyu Sun
Alibaba Group
xiuyu.sxy@alibaba-inc.com

Yichen Qian
Alibaba Group
yichen.qyc@alibaba-inc.com

Bolin Ding
Alibaba Group
bolin.ding@alibaba-inc.com

Jingren Zhou
Alibaba Group
jingren.zhou@alibaba-inc.com

ABSTRACT

Large language models (LLMs) have emerged as a new paradigm for Text-to-SQL task. However, the absence of a systematical benchmark inhibits the development of designing effective, efficient and economic LLM-based Text-to-SQL solutions. To address this challenge, in this paper, we first conduct a systematical and extensive comparison over existing prompt engineering methods, including question representation, example selection and example organization, and with these experimental results, we elaborate their pros and cons. Based on these findings, we propose a new integrated solution, named DAIL-SQL, which refreshes the Spider leaderboard with 86.6% execution accuracy and sets a new bar.

To explore the potential of open-source LLM, we investigate them in various scenarios, and further enhance their performance with supervised fine-tuning. Our explorations highlight open-source LLMs' potential in Text-to-SQL, as well as the advantages and disadvantages of the supervised fine-tuning. Additionally, towards an efficient and economic LLM-based Text-to-SQL solution, we emphasize the token efficiency in prompt engineering and compare the prior studies under this metric. We hope that our work provides a deeper understanding of Text-to-SQL with LLMs, and inspires further investigations and broad applications.

PVLDB Reference Format:
Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, Jingren Zhou. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. PVLDB, 17(5): 1132-1145, 2024.
doi:10.14778/3641204.3641221

PVLDB Artifact Availability:
The source code has been made available at https://github.com/BeachWang/DAIL-SQL.

*Co-first authors.

Proceedings of the VLDB Endowment, Vol. 17, No. 5 ISSN 2150-8097.
doi:10.14778/3641204.3641221

1 INTRODUCTION

Text-to-SQL, as one challenging task in both natural language processing and database communities, maps natural language questions on the given relational database into SQL queries [8, 18]. Most previous work [17, 21, 22, 48, 56] focus on extracting the question-to-SQL patterns and generalizing them by training an encoder-decoder model with Text-to-SQL corpus. In recent years, large language models (LLMs) have emerged as a new paradigm for Text-to-SQL [24, 39, 47]. Notably, equipped with GPT-4 [28], Pourreza et al. [35] achieved the first place in Spider leaderboard [2] with 85.3% execution accuracy. Different from prior studies, the core problem in LLM-based Text-to-SQL solution is how to prompt LLM to generate correct SQL queries, namely prompt engineering. Such prompt engineering involves question representations [6, 12, 31, 35], examples selection [14, 26, 27], and example organization [14].

Text-to-SQL prompt engineering needs a systematic study. Although prior studies have made remarkable progress, there still lacks a systematic study for prompt engineering in LLM-based Text-to-SQL solutions. Specifically, for question representation, most existing research textualize structured knowledge as schema, and further add task instructions and foreign keys to form prompts [19, 27]. Besides, some studies [6, 27] represent tables as several "*CREATE TABLE*" SQL statements, and prompt LLMs to answer the target question in comments. However, even with similar representation, their detailed task instructions can lead to significant performance gap. For example, in OpenAI's official Text-to-SQL demo [31], they employ the pound sign "#" to differentiate prompt from response, yielding an impressive performance [24]; If such a sign is removed, the performance will significantly drop. Therefore, there are burgeoning demands for a systematic study over different representations and examine how to work well with LLMs. Regarding example selection, a common practice is to encode the most similar examples in the same representation with the target question [6, 24, 27]. Nan et al. [27] further underline the importance of diversity in example selection. While for organization, most prior studies represent examples with full information, including instruction, schema, question and ground truth SQL queries. Besides, Guo et al. [14] only keep SQL queries in the selected examples to guide the LLM with less tokens. Together with different LLMs' preferences, the optimal selection and organization strategies in LLM-based Text-to-SQL solution remain ambiguous. Therefore, a systematical study on

prompt engineering, spanning different LLMs, question representations, example selection and organizations, is highly anticipated.

The potential of open-source LLMs is underexplored. Very recently, open-source LLMs are constantly expanding and show remarkable advancement in programming, mathematical reasoning, and text generation tasks. However, previous Text-to-SQL research primarily focuses on OpenAI LLMs, leaving open-source LLMs unstudied. Besides, compared with OpenAI LLMs, open-source ones generally have limited functionality in understanding context and generating coherent response. Thus, a critical challenge for open-source LLMs is to further enhance their performance in Text-to-SQL, which can be achieved by supervised fine-tuning.

Prompt efficiency remains a challenging open question. In LLM-based Text-to-SQL, another critical challenge is efficiency. The reason is that most prior studies focus on OpenAI LLMs, and calling their APIs are expensive, time-consuming and restricted in rate limits [30], especially for in-context learning prompts with multiple examples. However, the prior studies may not well tackle this challenge. Specifically, based on the observed inverted-U shape in execution accuracy with respect to prompt length, Chang et al. [6] conjectures that LLMs may have a sweet spot in terms of prompt length, but leaves exploring efficient prompt engineering a challenging open question.

In light of above challenges, we focus on providing a comprehensive, systematical and fair benchmark for LLM-based Text-to-SQL. Specifically, our benchmark discusses both the effectiveness and efficiency of various prompt engineering strategies, as well as the feasibility of open-source LLMs. They are detailed as follows.

To provide a systematical and in-depth understanding of Text-to-SQL prompt engineering, we empirically evaluate several strategies from prior studies. First, we compare several typical question representations in zero-shot scenario with different LLMs, and identify their pros and cons. After that, we investigate example selection and organization strategies in few-shot scenario. For example selection, we compare different selection strategies and further verify the hypothesis that LLMs learn from the mappings between question and SQL skeleton. Regarding example organization, we explore the option of displaying full information, solely SQL queries or question-SQL pair.

After that, we highlight the potential of open-source LLMs in both in-context learning and supervised fine-tuning. Specifically, we empirically study various open-source LLMs with different prompt engineering strategies, and observe the significant benefits of increasing scale of LLMs and having a good alignment [32]. To further enhance their performance, we fine-tune and evaluate open-source LLMs using various representations. With this comparison, we demonstrate that similar to in-context learning, representation strategy is also critical for supervised fine-tuning. These explorations underline the potential of an effective solution for Text-to-SQL. Moreover, after fine-tuning we also observe a decrease in in-context learning capability, which requires further study. We believe these explorations will benefit practical Text-to-SQL applications.

Towards a more economic and efficient solution, we further evaluate different strategies in terms of token efficiency. Such evaluation aims at searching for a cost-effective strategy, which is supposed to achieve considerable performance with less tokens. To fulfill such goal, we consider token efficiency in the whole process of prompt engineering, including choices for question representation, example selection and organization.

Last but not least, our integrated solution, named DAIL-SQL, refreshes the Spider leaderboard with 86.6% execution accuracy, and wins the first place. Compared with previous solutions, DAIL-SQL encodes structure knowledge as SQL statements, selects examples based on their skeleton similarities and removes cross-domain knowledge from examples for token efficiency. Before DAIL-SQL, the state-of-the-art performance in the Spider leaderboard is 85.3% [35]. Therefore, our solution sets a new bar, and hope our comprehensive study will inspire more further works.

Contribution Our main contributions and results are summarized as follows:

- We systematically study prompt engineering for LLM-based Text-to-SQL methods, including five question representations, two prompt components, four example selections, and three example organizations on four LLMs. The study sheds light on identifying suitable question representations and key points to leverage the in-context learning capacity of LLMs for Text-to-SQL task.
- To the best of our knowledge, we are the first to explore open-source LLMs for both in-context learning and supervised fine-tuning for Text-to-SQL task. We provide insights into the potential of the open-source LLMs by employing SFT on them in Text-to-SQL task.
- We also empirically compare different prompts in terms of cost efficiency, which provides practical guidance for real-world Text-to-SQL applications.
- Last but not least, we propose a new solution, named DAIL-SQL, which successes in leveraging the in-context learning capacity of LLMs and achieving a balance between performance and token efficiency. Notably, it refreshes the Spider leaderboard with 86.6% execution accuracy, which surpasses the best state-of-the-art solution by 1.3% with much less token cost.

2 PRELIMINARY

Text-to-SQL aims at automatically translating natural language questions into SQL queries. It bridges the gap between non-expert users and database systems, greatly improves the efficiency of data processing, and contributes to a wider range of applications such as intelligent database service, automatic data analysis and database question-answering. However, Text-to-SQL is still a quiet challenging task, due to the difficulty in fully understanding natural language questions and generating correct SQL queries [18, 37].

Extensive studies of Text-to-SQL have been conducted in both database and natural language processing communities. Some early studies tackle Text-to-SQL task with pre-defined rules or query enumeration [3, 38, 42], or treat it as a sequence-to-sequence task, focusing on training machine learning models with an encoder-decoder architecture [5, 34, 36]. With rapid advancement of deep learning, numerous techniques are applied to help Text-to-SQL task, such as attention mechanism [25], graph representation [17, 22, 36, 48, 51, 56], syntax parsing [15, 21, 41, 49], etc. One of the most

representative is BERT [10], which has been widely used in Text-to-SQL and achieved SOTA performances at that time [4, 52]. Besides, to narrow the gap between Text-to-SQL research and its real-world deployment, numerous large-scale benchmark datasets have been released, including WikiSQL [58], Spider [53], KaggleDBQA [20], BIRD [23] etc. With these great efforts, the research communities have made impressive progress in Text-to-SQL.

Recently, large language models (LLMs), such as GPT-4 [28] from OpenAI and LLaMA [45] from Meta, have emerged as a milestone for natural language processing and machine learning. Different from general machine learning model, LLMs are pre-trained on massive text corpus, which can perform various natural language tasks. The basic operating principle is to gradually produce the next word that has the highest probability based on the input prompt [54]. Therefore, to tackle Text-to-SQL task with LLMs, the core is to find the optimal prompt, also known as prompt engineering [24, 27].

Specifically, according to number of examples provided in prompt, prompt engineering are classified into two scenarios: zero-shot scenario and few-shot scenario. In zero-shot scenario, no example is provided, and the main challenge is to represent the natural language question effectively, including incorporating relevant information such as the corresponding database schema [6, 12, 24, 47]. In this paper, the process of representing natural language questions and relevant information is referred to as **question representation**. While in few-shot scenario, a limited number of examples are available, thus besides question representation, we also need to study how to select the most helpful examples and organize them in the prompt appropriately. In natural language processing, the above progress that LLMs learn from contextual examples is named as **in-context learning** [11]. It enables LLMs to identify explicit or inexplicit patterns from the input prompt, and generate corresponding outputs. In this way, LLMs are capable of new tasks during inference without any explicit task-specific training phase. Recent studies [14, 26, 35] confirm the significant role of including examples for effective in-context learning.

Although LLMs are demonstrated to be effective in both zero-shot and few-shot scenarios in prior studies [6, 19, 24, 27, 43], their performances can be further enhanced by **supervised fine-tuning (SFT)**, which enhances LLMs using additional task-specific training data to make it more suitable for specific downstream tasks. In recent researches, supervised fine-tuning is used as a training paradigms of **Alignment**, which aligns LLMs' behavior to avoid generating offensive, biased responses and hallucinations [29]. In this paper, we will focus on enhancing LLMs' Text-to-SQL capabilities with supervised fine-tuning. It is worth noting that despite the extensive research on prompt engineering for Text-to-SQL, there is a scarcity of studies exploring the supervised fine-tuning of LLMs for Text-to-SQL [43], leaving this area as an open question.

In summary, question representation, in-context learning, together with supervised fine-tuning are three essential knobs in large language model based Text-to-SQL. In this paper, we will provide a systematical study and discussion about them.

3 METHODOLOGY

As stated above, in this paper we focus on question representation, in-context learning and supervised fine-tuning. In this section, we provide formal definitions for these three problems, survey their existing solutions systematically, and point out the potential issues in existing techniques. To address these issues, we propose a new Text-to-SQL prompt engineering method, named DAIL-SQL, which refreshes the best performance in Spider leaderboard with 86.6% execution accuracy.

Table 1: Question representations in existing works, as well as their reported execution accuracy (EX) in zero-shot scenario. The Instruction (INS), Rule Implication (RI) and Foreign Key (FK) are possible components in a prompt. INS is the task description, such as "Write a SQL to answer the question". RI is the guiding statement, such as "Complete sqlite SQL query only and with no explanation". FK is the foreign key information of the database.

Question Representation	INS	RI	FK	Ref.	LLMs	EX (%)
BS $_P$	✗	✗	✗	[35]	-	-
TR $_P$	✓	✗	✗	[27]	CODE-DAVINCI-002	69.0
OD $_P$	✓	✓	✗	[24]	GPT-3.5-TURBO	70.1
				[35]	GPT-4	64.9
CR $_P$	✓	✗	✓	[27]	CODE-DAVINCI-002	75.6
				[6]	CODE-DAVINCI-002	71.8
				[6]	GPT-3.5-TURBO	70.7
AS $_P$	✓	✗	✗	[44]	-	-

```
Table continents, columns = [ContId, Continent]
Table countries, columns = [CountryId, CountryName,
↳ Continent]
Q: How many continents are there?
A: SELECT
```

Listing 1: Example of Basic Prompt

```
Given the following database schema:
continents: ContId, Continent
countries: CountryId, CountryName, Continent

Answer the following: How many continents are there?
SELECT
```

Listing 2: Example of Text Representation Prompt

3.1 Question Representation

In this section, we first discuss question representations under zero-shot scenario for Text-to-SQL. Considering a target question q in natural language on certain database $\mathcal{D}$, the target of question representation is to maximize the possibility of LLM $\mathcal{M}$ generating the correct SQL s^* as follows:

$$\max_{\sigma} \quad \mathbb{P}_{\mathcal{M}}(s^* | \sigma(q, \mathcal{D})),$$

where function $\sigma(\cdot, \cdot)$ decides representation for target question q, with the useful information from the schema of database $\mathcal{D}$. Besides, $\sigma(\cdot, \cdot)$ also can include information such as instruction statement, rule implication and foreign key.

Follow the above definition, we survey different choices of σ in zero-shot scenario and choose four most representative ones from

```
### Complete sqlite SQL query only and with no explanation
### SQLite SQL tables, with their properties:
#
# continents(ContId, Continent)
# countries(CountryId, CountryName, Continent)
#
### How many continents are there?
SELECT
```

Listing 3: Example of OpenAI Demostration Prompt

literature. In addition, we also include the question representation used in Alpaca [44] since it's popular in supervised fine-tuning. Table 1 summarizes these five representation methods and lists their reported details from their original papers.

- **Basic Prompt** (BS $_P$). Basic Prompt [35] is a simple representation shown in Listing 1. It is consisted of table schemas, natural language question prefixed by "*Q:* " and a response prefix "*A: SELECT*" to prompt LLM to generate SQL. In this paper we named it as Basic Prompt due to its absence of instructions.
- **Text Representation Prompt** (TR $_P$). As shown in Listing 2, Text Representation Prompt [27] represents both schema and question in natural language. Compared with Basic Prompt, it adds instruction at the very beginning of prompt to guide LLMs. In [27], it achieves 69.0% execution accuracy on Spider-dev in zero-shot scenario.
- **OpenAI Demostration Prompt** (OD $_P$). The OpenAI Demostration Prompt (Listing 3) is first used in OpenAI's official Text-to-SQL demo [31], and evaluated in [24, 35]. It's consisted of instruction, table schemas, and question, where all information are commented by pound sign "#". Compared with Text Representation Prompt, the instruction in OpenAI Demostration Prompt is more specific with a rule, "*Complete sqlite SQL query only and with no explanation*", which we will further discuss in the Sec. 4.3 along with experimental results.
- **Code Representation Prompt** (CR $_P$). The Code Representation Prompt [6, 27] presents Text-to-SQL task in SQL syntax. Specifically, as shown in Listing 4, it directly presents "*CREAT TABLE*" SQLs, and prompts LLM with natural language question in comments. Compared with other representations, CR $_P$ stands out due to its ability to provide comprehensive information necessary for database creation, such as column types and primary/foreign keys. With such a representation, [27] correctly predicts about 75.6% SQLs with LLM CODE-DAVINCI-002.
- **Alpaca SFT Prompt** (AS $_P$). The Alpaca SFT Prompt is a prompt designed for supervised fine-tuning [44]. As shown in Listing 5, it prompts LLM to follow instruction and finish task according to the input context in Markdown format. We include it to examine its effectiveness and efficiency in both prompt engineering and supervised fine-tuning scenarios.

```
/* Given the following database schema: */
CREATE TABLE continents(
    ContId int primary key,
    Continent text,
    foreign key(ContId) references countries(Continent)
);

CREATE TABLE countries(
    CountryId int primary key,
    CountryName text,
    Continent int,
    foreign key(Continent) references continents(ContId)
);

/* Answer the following: How many continents are there? */
SELECT
```

Listing 4: Example of Code Representation Prompt

```
Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:
Write a sql to answer the question "How many continents are there?"

### Input:
continents(ContId, Continent)
countries(CountryId, CountryName, Continent)

### Response:
SELECT
```

Listing 5: Example of Alpaca SFT Prompt

As shown in Table 1, different representations are experimented with different LLMs, and integrated in different frameworks, making it difficult to compare them fairly and effectively. Additionally, the specific roles played by individual components such as foreign key information and rule implication remain unclear. Consequently, it is essential to conduct a systematical study to better understand question representations, and investigate their advantages and disadvantages through a fair comparison.

3.2 In-Context Learning for Text-to-SQL

The above question representation methods enable LLMs to directly output desired SQLs by zero-shot learning. However, LLMs can perform better for Text-to-SQL through in-context learning, in which only a few examples are provided in the input prompts. Therefore, in this subsection, we discuss the keys of in-context learning, that are example selection and example organization. We first give a formulation of in-context learning to ease the further discussions.

In Text-to-SQL, given a set of triples $Q = \{(q_i, s_i, \mathcal{D}_i)\}$, where q_i and s_i are natural language question and its corresponding SQL query on database $\mathcal{D}_i$, the target of in-context learning for Text-to-SQL is to maximize the possibility of LLM $\mathcal{M}$ generating the

correct SQL s^* on the target question q and database $\mathcal{D}$ as follows:

$$\max_{\mathcal{Q}',\sigma} \quad \mathbb{P}_{\mathcal{M}}(s^*|\sigma(q,\mathcal{D},\mathcal{Q}')),$$
$$\text{s.t.} \quad |\mathcal{Q}'| = k \quad \text{and} \quad \mathcal{Q}' \subset \mathcal{Q},$$

where function $\sigma(\cdot,\cdot,\cdot)$ decides representation for target question q, with the useful information from the schema in database $\mathcal{D}$ and k examples selected from $\mathcal{Q}$. In this paper, we focus on *cross-domain Text-to-SQL*, which means the target database $\mathcal{D}$ is not included among the databases $\mathcal{D}_i$ mentioned in $\mathcal{Q}$., i.e., $\mathcal{D} \notin \{\mathcal{D}_i|(q_i, s_i, \mathcal{D}_i) \in \mathcal{Q}\}$.

In-context learning for Text-to-SQL involves selecting the most helpful examples $\mathcal{Q}'$ and deciding how to organize the information of these selected examples into prompt. Next we discuss these two sub-tasks: example selection and example organization.

3.2.1 Example Selection. We summarize various example selection strategies in prior studies as follows.

- **Random**. This strategy randomly samples k examples from the available candidates. Previous works [14, 26, 27] have adopted it as a baseline for example selection.
- **Question Similarity Selection** (QTS$_S$). QTS$_S$ [26] chooses k examples with the most similar questions. Specifically, it embeds both example questions in $\mathcal{Q}$ and the target question q with a pre-trained language model. Then it applies a pre-defined distance measure, such as the Euclidean distance or negative cosine similarity, to each example-target pair. Finally kNN algorithm is leveraged to select k examples from $\mathcal{Q}$ that closely match the target question q.
- **Masked Question Similarity Selection** (MQS$_S$). For cross-domain Text-to-SQL, MQS$_S$ [14] eliminates the negative influence of domain-specific information by replacing table names, column names, and values in all questions with a mask token, and then compute the similarities of their embedding with kNN algorithm.
- **Query Similarity Selection** (QRS$_S$). Instead of using the target question q, QRS$_S$ [27] aims to select k examples that are similar to target SQL query s^*. Specifically, it employs a preliminary model to generate SQL query s' using target question q and database D, where this generated s' can be regarded as an approximation of s^*. Then it encodes queries from examples into binary discrete syntax vectors according to their keywords. After that, it chooses k examples by considering both similarity to the approximated query s' and diversity among selected examples.

Above strategies focus on selecting examples using only target question or query. However, according to prior studies [11], in-context learning is essentially learning from analogy. In the case of Text-to-SQL, the objective is to generate queries that match the given questions, thus LLMs are supposed to learn the mapping from questions to SQL queries. Therefore, we point out that during example selection, taking both question and SQL queries into consideration may benefit Text-to-SQL task. We will further discuss it in Sec. 3.3.

3.2.2 Example Organization. The example organization plays a pivotal role in determining what information of the above selected examples will be organized into the prompt. We summarize existing strategies in prior studies into two categories, Full-Information Organization and SQL-Only Organization, as demonstrated in Listing 6 and Listing 7. In these examples, *${DATABASE_SCHEMA}* represents the database schema, and *${TARGET_QUESTION}* stands for the question representation in Listing 4.

```
/* Given the following database schema: */
${DATABASE_SCHEMA}
/* Answer the following: How many authors are there? */
SELECT count(*) FROM authors

/* Given the following database schema: */
${DATABASE_SCHEMA}
/* Answer the following: How many farms are there? */
SELECT count(*) FROM farm

${TARGET_QUESTION}
```

Listing 6: Example of Full-Information Organization.

```
/* Some SQL examples are provided based on similar
↳ problems: */
SELECT count(*) FROM authors

SELECT count(*) FROM farm

${TARGET_QUESTION}
```

Listing 7: Example of SQL-Only Organization.

```
/* Some example questions and corresponding SQL queries
↳ are provided based on similar problems: */
/* Answer the following: How many authors are there? */
SELECT count(*) FROM authors

/* Answer the following: How many farms are there?. */
SELECT count(*) FROM farm

${TARGET_QUESTION}
```

Listing 8: Example of DAIL Organization.

- **Full-Information Organization** (FI$_O$). FI$_O$ [6, 27] organizes examples in the same representation with the target question. As shown in Listing 6, examples are structured identically to the target question, and the only difference is that instead of the "*SELECT*" token at the end, the selected examples have the corresponding SQL queries after "*SELECT*".
- **SQL-Only Organization** (SO$_O$). SO$_O$ [14] includes only SQL queries of the selected examples with a prefix instruction in the prompt, as demonstrated in Listing 7. Such organization aims at maximizing the number of examples with limited token length. However, it removes the mapping information between questions and corresponding SQL queries, and such information can be useful, which we will demonstrate later.

In summary, FI$_O$ includes the full information of examples, which ensures the quality; while SO$_O$ only keeps SQL queries to accommodate more examples, which prefers the quantity. We wonder if there exists a better trade-off between quality and quantity in example organization , which can further benefit the Text-to-SQL task.

3.3 DAIL-SQL

To address the aforementioned issues in example selection and organization, in this subsection, we present a novel Text-to-SQL method named DAIL-SQL. Due to limited space please refer to our full version [13] for the pseudocode.

For example selection, inspired by MQS_S and QRS_S, we proposed **DAIL Selection** (DAIL_S), considering both questions and queries to select candidates. Specifically, DAIL Selection first masks domain-specific words in both target question q and example questions q_i in the candidate set Q. It then ranks the candidate examples based on the Euclidean distance between the embeddings of masked q and q_i. Simultaneously, it calculates the query similarity between the pre-predicted SQL query s' and s_i in Q. Finally, the selection criterion prioritizes the sorted candidates by question similarity with a query similarity greater than a predefined threshold τ. In this way, the selected top k examples have good similarity with both question and query.

To preserve the mapping information between questions and SQL queries and also improve the token efficiency, we propose a new example organization strategy **DAIL Organization** (DAIL_O) to trade-off in terms of quality and quantity. Specifically, DAIL_O presents both questions q_i and corresponding SQL queries s_i, as illustrated in Listing 8. As a compromise between FI_O and SO_O, DAIL_O reserves the question-SQL mapping, and reduces the token length of examples by removing token-cost database schema.

In DAIL-SQL, we adopt CR_P as our question representation. The reason is that compared with other representations, CR_P contains full information of the database, including primary and foreign keys, which may offers more useful information for LLMs, such as foreign keys for the prediction of "JOIN" clauses. Besides, pre-trained on extensive coding corpora, LLMs could better understand the prompt in CR_P without too much additional effort.

In summary, DAIL-SQL utilizes CR_P as the question representation, selects examples based on information from both question and query, and organizes them to keep question-to-SQL mappings. In such prompt design, LLMs could work better for Text-to-SQL task, and in Spider leaderboard, the proposed DAIL-SQL refresh the performance with 86.2% execution accuracy.

Note DAIL-SQL is a flexible LLM-based Text-to-SQL solution, which can be further extended and integrated with other components easily. For example, to improve the performance, we equip DAIL-SQL with self-consistency [50], which achieves a performance of 86.6% execution accuracy. Although self-consistency improves the execution accuracy by 0.4%, it is very time consuming and yields many times the cost of original DAIL-SQL. Therefore, in this paper we still focus on DAIL-SQL.

3.4 Supervised Fine-Tuning for Text-to-SQL

To enhance the performance of LLMs in zero-shot scenario, the popular option for existing Text-to-SQL methods is in-context learning, which is discussed in above subsections. As an alternative yet promising option, supervised fine-tuning is less explored so far. Similar to supervised fine-tuning for various language task, we can adopt it to the field of Text-to-SQL, and improve LLMs' performance on this downstream task. To further understand how supervised fine-tuning works for Text-to-SQL, we first provide a brief formulation as follows.

For Text-to-SQL, given a large language model $\mathcal{M}$, a set of Text-to-SQL training data $\mathcal{T} = \{(q_i, s_i, \mathcal{D}_i)\}$, where q_i and s_i are the natural language question and its corresponding query on database $\mathcal{D}_i$, the objective of supervised fine-tuning is to minimize the following empirical loss:

$$\min_{\sigma, \mathcal{M}^*} \sum_{i=1}^{|\mathcal{T}|} \mathcal{L}_{\mathcal{M}^*}(\sigma(q_i, \mathcal{D}_i), s_i),$$

where $\mathcal{L}$ is the loss function to measure the difference between the generated query and the groundtruh query. Similar to question representation, σ decides question representation with useful information from the schema in database $\mathcal{D}$. In this definition, supervised fine-tuning for Text-to-SQL covers two sub-tasks, including fine-tuning the given LLM $\mathcal{M}$ using supervised data $\mathcal{T}$ in order to get the optimal LLM $\mathcal{M}^*$, and searching for the optimal question representation σ. Since question representations have been discussed in Sec. 3.1, this section will primarily focus on data preparation $\mathcal{T}$ and fine-tuning.

For general domain, each item in supervised data $\mathcal{T} = \{(p_i, r_i)\}$ contains an input prompt p_i and an expected respond r_i from LLM. To ensure consistency with the inference process, we employ a supervised fine-tuning and generate prompt-response pairs from a given Text-to-SQL dataset. Specifically, given a Text-to-SQL data set $\mathcal{T} = \{(q_i, s_i, \mathcal{D}_i)\}$, we fine-tune the LLMs using the generated tuning data by using target question and the given database as prompt, and treating the desired query as response from LLM, i.e., $\mathcal{T} = \{(p_i = \sigma(q_i, \mathcal{D}_i), r_i = s_i)\}$. Once the data is ready, we can use existing package to fine-tune the given LLM $\mathcal{M}$ through either full fine-tuning [32] or parameter-efficient fine-tuning [16] depending on the available computational resources. After fine-tuning, the optimized LLM $\mathcal{M}^*$ can be used to do inference, that is asking it to generate queries through natural language questions. Note that we utilize the same question representation σ in both fine-tuning and inference processes. We will conduct a series of experiments and discuss the great potential of supervised fine-tuning for Text-to-SQL.

4 EXPERIMENT

In this section, we first introduce our experimental settings. Then we conduct extensive comparisons with existing solutions in question representation, in-context learning and supervised fine-tuning respectively. After that, we further compare them in terms of token efficiency to inspire more efficient solutions.

4.1 Setting

Dataset. We evaluate Text-to-SQL methods on two well recognized datasets, **Spider** [53] and **Spider-Realistic** [9]. Spider is a large-scale cross-domain Text-to-SQL dataset, which contains 8659 instances in training split and 1034 instances in development split over 200 databases. Each instance is consisted of a natural language question on a specific database and its corresponding SQL query. In this paper, we use the development split *Spider-dev* for the purpose of evaluation as the test split is not released. Spider-Realistic [9] is a more challenging variant of Spider. It selects a subset of 508

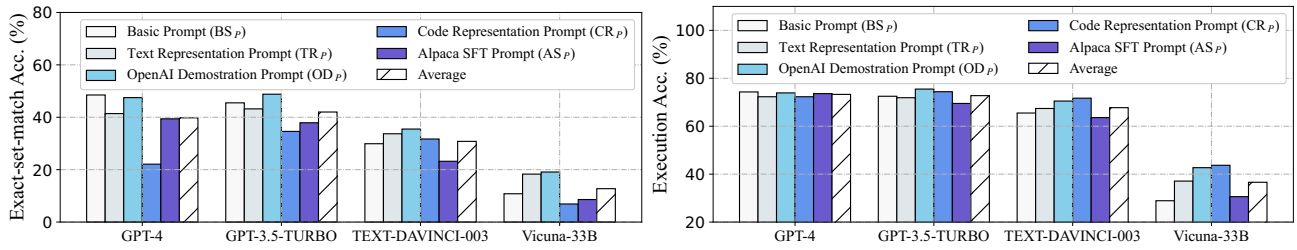


Figure 1: Results of different question representations on Spider-dev under zero-shot scenario.

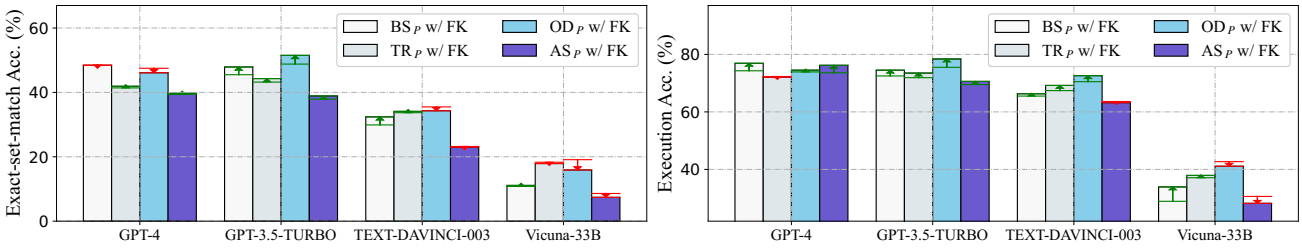


Figure 2: Ablation studies of foreign keys information on Spider-dev. The green arrow indicates an increase, and red arrow indicates a decrease.

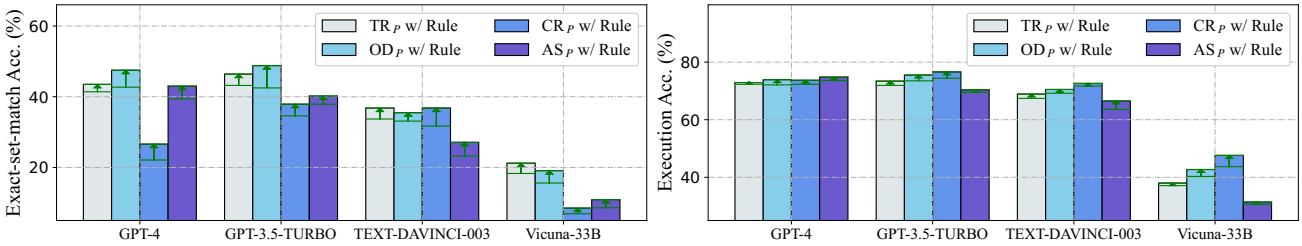


Figure 3: Ablation studies of "with no explanation" rule implication on Spider-dev. The green arrow indicates an increase, and red arrow indicates a decrease.

examples from Spider-dev and manually revises the questions while keeping the SQL queries unchanged. For few-shot scenarios, we utilize the training split of Spider as the example candidates when testing with both Spider-dev and Spider-Realistic.

Metric. To make a fair comparison, we follow prior study [57] to use exact-set-match accuracy (**EM**) and execution accuracy (**EX**). The exact-set-match accuracy measures the matched SQL keywords between the predicted SQL query and its corresponding ground truth. The execution accuracy, on the other hand, compares the execution output of the predicted SQL query with that of the ground truth SQL query on some database instances. This metric provides a more precise estimate of the model's performance since there may be multiple valid SQL queries for a single given question. We use the existing released evaluation scripts at https://github.com/taoyds/test-suite-sql-eval.

LLM. To ensure a fair comparison, for all the methods, we use the same maximal context length, that is 4096 for OpenAI LLM and 2048 for open-source LLM. During evaluation, we leave 200 tokens for response generation. By default, we set the argument temperature as 0 to eliminate the influence of randomness. Regarding post-processing, we follow existing work to extract the first SQL query in response and remove additional output. For more implementation details, please refer to our full version [13].

4.2 Question Representations

In this subsection, we evaluate the question representations presented in Sec. 3.1 under zero-shot scenario, employing four LLMs: GPT-4, GPT-3.5-TURBO, TEXT-DAVINCI-003, and Vicuna-33B.

Fig. 1 presents the comparison of different question representations over Spider-dev. By comparing different representations, we can observe that OD $_P$ fits to all four LLMs and achieves 75.5% execution accuracy with GPT-3.5-TURBO. In contrast, AS $_P$ exhibits poor performance with GPT-3.5-TURBO, TEXT-DAVINCI-003, and Vicuna-33B, necessitating a suitable LLM to work well with. Unexpectedly, GPT-4 exhibits a preference for the simple BS $_P$ derived from Din-SQL [35], indicating that a powerful LLM can mitigate the complexities associated with representation design. Besides, by comparing the average performance for four LLMs, GPT-4 and GPT-3.5-TURBO are more capable in the zero-shot scenario. Due to

Table 2: Evaluation on Spider-dev with different example selections. The organization is fixed to Full-Information Organization.

Few-shot	Selection	Question Similarity	Query Similarity	GPT-4		GPT-3.5-TURBO		TEXT-DAVINCI-003		Vicuna-33B	
				EM	EX	EM	EX	EM	EX	EM	EX
0-shot	-	-	-	22.1	72.3	34.6	74.4	31.7	71.7	6.9	43.7
1-shot	Random	0.23	0.47	41.7	77.4	45.9	73.9	38.2	70.6	14.4	47.9
	Question Similarity Selection	0.39	0.65	53.3	78.8	51.9	74.3	44.1	72.3	16.5	48.5
	Masked Question Similarity Selection	0.57	0.80	58.2	79.1	57.4	76.0	47.9	75.0	21.4	48.7
	DAIL Selection	0.56	0.95	62.1	80.2	59.5	75.5	51.9	76.9	22.8	49.2
	Upper Limit	0.56	0.98	63.7	81.0	61.4	77.2	53.1	77.5	22.7	49.4
3-shot	Random	0.23	0.48	48.9	79.4	49.0	73.6	41.7	71.6	16.8	46.9
	Question Similarity Selection	0.37	0.63	56.3	79.2	53.8	74.7	52.2	74.1	21.1	47.1
	Masked Question Similarity Selection	0.54	0.78	66.1	81.5	61.1	77.3	59.7	77.0	27.7	52.3
	DAIL Selection	0.53	0.94	69.1	81.7	63.9	77.8	64.4	79.5	30.7	53.6
	Upper Limit	0.53	0.98	71.5	83.4	66.2	79.2	66.7	81.1	31.2	54.4
5-shot	Random	0.23	0.48	51.6	79.5	52.9	75.7	49.0	72.1	-	-
	Question Similarity Selection	0.36	0.61	58.2	79.9	55.9	75.1	54.8	73.2	-	-
	Masked Question Similarity Selection	0.52	0.77	66.8	82.0	62.3	77.9	64.7	78.6	-	-
	DAIL Selection	0.52	0.94	71.9	82.4	66.7	78.1	67.7	80.5	-	-
	Upper Limit	0.51	0.97	74.4	84.4	68.8	79.6	70.7	82.4	-	-

the expensive cost of GPT-4, GPT-3.5-TURBO together with OD $_P$ maybe a better choice for the zero-shot scenario. For less powerful LLMs like TEXT-DAVINCI-003 and Vicuna-33B, OD $_P$ and CR $_P$ are preferred.

To further investigate the different question representations, we conduct ablation study to explore the effects of their invidual components.

Foreign Key (FK). Foreign Key implies the relation among different relational tables, which might be helpful in Text-to-SQL task. In our evaluation, only CR $_P$ contains foreign key information. To examine its effect, we add foreign key information into other representations and evaluate them in Fig. 2. For OpenAI LLMs, we observe that foreign key significantly improves the execution accuracy of LLMs by $0.6\% - 2.9\%$, except the combinations of TR $_P$ with GPT-4 (-0.2%) and AS $_P$ with TEXT-DAVINCI-003 (-0.4%). However, the impact of foreign key for Vicuna-33B tends to be unstable. Notably, the inclusion of foreign keys leads to a surprising improvement of 5.0% for the BS $_P$, but adversely affects the performance of the OD $_P$ and AS $_P$.

Rule Implication (RI). Inspired by the outperformance of OD $_P$, we explore the effect of rule implication. Specifically, OD $_P$ implicate LLMs to generate SQL queries "*with no explanation*" To examine the effect of "*with no explanation*" rule in question representation, we present an ablation study in Fig. 3. Specifically, we plot the performance of different representations after including the "*with no explanation*" implication and the change of accuracy. From Fig. 3 we observe adding this rule consistently booms the performance of all LLMs in both exact-set-match and execution accuracy, with the most significant improvements exceeding 6% and 3%, respectively. While for OD $_P$, removing this rule incurs about $2.4\% - 6.2\%$ drop in exact-set-match accuracy, and $1.3\% - 2.4\%$ drop in execution accuracy, indicating the importance of this rule implication.

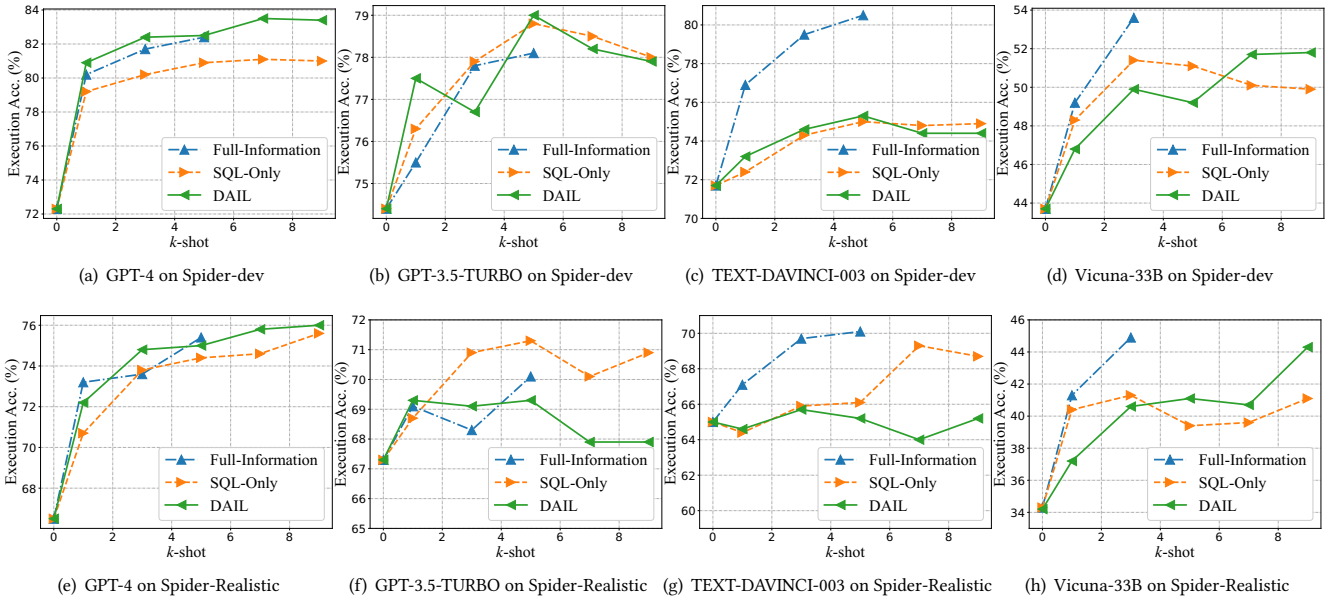
In summary, both the foreign key and the "*with no explanation*" rule implication are beneficial for Text-to-SQL task. In our evaluation, OD $_P$ with foreign keys and GPT-3.5-TURBO are the most effective and economic combination, which achieves 51.5% exact-set-match accuracy and 78.4% execution accuracy.

4.3 In-Context Learning for Text-to-SQL

In few-shot scenario, we examine different example selection and organization strategies with GPT-4, GPT-3.5-TURBO, TEXT-DAVINCI-003, and Vicuna-33B. To ensure a fair comparison, we adopt CR $_P$ as the question representation for all the experiments in this subsection, due to its superior performance in one-shot preliminary experiment as shown in our full version [13].

4.3.1 Example Selection. To verify the importance of both question and query for example selection, we calculate question's and query's Jaccard similarities between chosen examples and the target instance, and report the averaged numbers under column *question similarity* and *query similarity* in Table 2. Specifically, we remove database-specific information from questions [48] and queries [21], and calculate the Jaccard similaritis of the remained tokens. Besides, we introduce **Upper Limit** for reference, which is similar with DAIL Selection but utilizes the ground truth query s^* rather than the query generated by preliminary predictor. Notably, we do not directly provide the ground truth to the LLMs, but just use the ground truth query as a reference for selecting examples. To some extent, Upper Limit indicates the upper bound of performance for similarity based selection methods.

Table 2 shows the comparisons of different example selection strategies in 1-, 3- and 5-shot scenarios on Spider-dev. By comparing different selection strategies, it is demonstrated that DAIL $_S$ generally outperforms other strategies. In 5-shot scenario, equipped with GPT-4, DAIL-SQL achieves 82.4% execution accuracy. Besides, in Table 2 we observe the increasing question and query similarity corresponds to higher execution accuracy mostly, indicating the importance of considering both question and query similarity. Note DAIL $_S$'s execution accuracy is still lower than Upper Limit. This



(a) GPT-4 on Spider-dev (b) GPT-3.5-TURBO on Spider-dev (c) TEXT-DAVINCI-003 on Spider-dev (d) Vicuna-33B on Spider-dev

(e) GPT-4 on Spider-Realistic (f) GPT-3.5-TURBO on Spider-Realistic (g) TEXT-DAVINCI-003 on Spider-Realistic (h) Vicuna-33B on Spider-Realistic

Figure 4: Evaluation on Spider-dev with different example organizations. The selection is fixed to DAIL Selection.

discrepancy can be attributed to the lower query similarity, indicating the gap between the ground truth query and that generated by the preliminary model.

4.3.2 Example Organization. To compare different example organization strategies, we evaluate Full-Information Organization, SQL-Only Organization and DAIL Organization in few-shot scenario on both Spider-dev and Spider-Realistic. Fig. 4 shows the comparison results.

From Fig. 4(a) and Fig. 4(e), we can observe that GPT-4 benefits from contextual examples steadily on both Spider-dev and Spider-Realistic. With DAIL Organization, its execution accuracy increases from 72.3% to 83.5% on Spider-dev and from 66.5% to 76.0% on Spider-Realistic. While for GPT-3.5-TURBO and TEXT-DAVINCI-003, adding examples may incur drop in execution accuracy due to limited in-context learning capability. Regarding Vicuna-33B, its performance consistently improves as the number of examples increases in DAIL Organization. By comparing different organization strategies, we observe that GPT-4 shows preference for DAIL Organization in both Spider-dev and Spider-Realistic, suggesting it can effectively learn the mapping from question-SQL pairs. For GPT-3.5-TURBO (Fig. 4(b) and Fig. 4(f)), compared with its zero-shot performance in Fig. 1, its enhancement in in-context learning is the smallest among four LLMs, due to its weakness in in-context learning. For TEXT-DAVINCI-003, Full-Information Organization is far beyond the other two strategies, especially with increasing example number, as depicted in Fig. 4(c) and Fig. 4(g). Figures 4(d) and 4(h) illustrate that in the case of Vicuna-33B, DAIL Organization outperforms SQL-Only Organization but falls short of the performance achieved by Full-Information Organization. By comparing different LLMs, we infer that for LLM with greater in-context learning capability, like GPT-4, benefits from DAIL Organization the most, while the weaker LLMs require more information to learn from examples. However, we emphasize DAIL Organization can be a good choice to achieve higher performance, and the best execution accuracy in our evaluation is achieved by DAIL Organization with GPT-4.

In summary, for example selection, our findings emphasize the importance of the mapping from question to SQL query. Considering both question and query similarities simultaneously, DAIL $_S$ outperforms other selection strategies in our evaluation. For example organization, we show the effectiveness of DAIL $_O$, and point out its demands for potent LLMs. Finally, in our evaluation, we observe that our approach, DAIL-SQL, equipped with GPT-4, achieves the highest performance with an execution accuracy of 83.5% on Spider-dev and 76.0% on Spider-Realistic.

4.4 Supervised Fine-Tuning for Text-to-SQL

In this section, we investigate supervised fine-tuning in Text-to-SQL. Due to the unaffordable cost of fine-tuning OpenAI LLMs, we focus on open-source LLMs. Given the fact that very few existing work adopt open-source LLMs and their performance remain unknown, we first undertake a thorough evaluation for open-source LLMs, employing various question representation, example selection and organization strategies. After that, we fine-tune open-source LLMs in Text-to-SQL and observe their enhancement in both zero-shot and few-shot scenarios.

4.4.1 Open-source LLM. To investigate the potential of open-source LLM, we choose LLaMA [45], and its aligned variants in varying scales. They are detailed as follows. Note the aligned variants means the LLM is aligned to be more helpful, harmless and honest [1], and the suffix "-7B" means the LLM has 7 billions parameters, the same meaning for "-13B" and "-33B".

Table 3: Zero-shot evaluation results on Spider-dev with different open-source LLMs. The best performances of pre-trained and aligned LLM are in bold.

	LLM	BS $_P$		TR $_P$		OD $_P$		CR $_P$		AS $_P$		Average	
		EM	EX	EM	EX	EM	EX	EM	EX	EM	EX	EM	EX
Pre-trained	LLaMA-7B	6.5	9.6	3.1	4.9	3.6	9.0	4.8	16.3	1.3	5.9	3.9	9.1
	LLaMA-13B	8.8	18.4	4.5	15.2	8.2	21.8	5.6	25.0	8.9	26.9	7.2	21.5
	LLaMA-33B	9.6	26.7	12.0	25.9	13.6	36.4	12.2	**42.8**	**13.8**	38.1	12.2	34.0
	Falcon-40B	0.3	11.7	0.2	0.9	0.3	7.6	0.1	21.9	0.0	5.0	0.2	9.4
Aligned	Vicuna-7B	7.5	15.6	1.2	9.9	6.2	21.5	5.6	24.0	0.9	5.4	4.3	15.3
	Vicuna-13B	8.2	21.7	10.1	24.4	11.2	31.4	5.8	33.5	4.7	20.0	8.0	26.2
	Vicuna-33B	10.8	28.9	18.3	37.1	19.1	42.7	6.9	43.7	8.6	30.6	12.7	36.6
	LLaMA-2-CHAT-7B	14.3	23.4	7.2	15.5	6.3	12.3	12.2	25.5	5.0	20.5	9.0	19.4
	LLaMA-2-CHAT-13B	18.8	32.6	15.4	30.5	11.1	22.3	20.7	40.0	16.9	36.2	16.6	32.3
	LLaMA-2-CHAT-70B	21.8	46.2	11.9	33.9	21.4	45.5	12.4	44.0	8.4	28.6	15.2	39.6
	CodeLLaMA-34B	**27.8**	65.5	15.9	40.3	25.8	65.3	24.3	**68.5**	22.4	61.5	23.2	60.2

- **LLaMA-7B/13B/33B** [45] is a collection of widely recognized open-source LLMs, which are pre-trained on massive corpus by Meta.
- **Falcon-40B** [33] is pre-trained solely on massive corpus of refined web data.
- **LLaMA-2-CHAT-7B/13B/70B** [46] are up-to-date version of LLaMA. They are both pre-trained and aligned, and outperform the previous version on most benchmarks.
- **Vicuna-7/13/33B** [7, 55] is a collection of open-source chatbot aligned from LLaMA with user-shared conversations. Vicuna-13B [7] is declared to perform similar to OpenAI ChatGPT and Google Bard, and outperforms LLaMA and Alpaca in most scenarios.
- **CodeLLaMA-34B** [40] is fine-tuned from LLaMA-2-34B with about 500B tokens of code data.

4.4.2 Zero-shot Scenario with Open-source LLM. Table 3 shows their zero-shot performances on Spider-dev with different question representations. Due to limited space, the performance on Spider-Realistic refers to our full version [13]. Next, we provide several analysis from aspects of question representations, model scale and alignment as follows.

Effect of Question Representation. We can observe that the best performances is achieved by CR $_P$ with 68.5% execution accuracy on Spider-dev. One possible reason is that CR $_P$ tends to stimulate the coding capability of LLMs. This effect is particularly evident in CodeLLaMA-34B, which only achieve 40.3% execution accuracy with natural language-based TR $_P$.

Effect of Model Scale. From the results we observe a positive correlation between model scale and performance on Text-to-SQL for both LLaMA and Vicuna. Specifically, the average execution match accuracy of LLaMA shows a notable progression from 9.1% to 34.0% on Spider-dev, and Vicuna shows a similar upward trend from 15.3% to 36.6%. With the most parameter size, LLaMA-2-CHAT-70B improves the average performance to 39.6%.

Effect of Alignment. From the results we observe that LLM alignment can benefit Text-to-SQL. Specifically, with the same model scale, Vicuna outperforms LLaMA by about 5% in execution accuracy on both Spider-dev and Spider-Realistic. For Falcon-40B, it performs poorly with all representations, attributable to the absence of dedicated code data in its training dataset. As a comparison, with carefully collected code data in the alignment stage, CodeLLaMA-34B exhibits a significant improvement in Text-to-SQL task with similar model scale. Note that, CodeLLaMA-34B also outperforms LLaMA-2-CHAT-70B by an average of 20.6% despite having only half the parameter of LLaMA-2-CHAT-70B. This highlights the crucial importance of the training corpus in LLMs.

In conclusion, having more parameters in LLMs may hold certain potential benefits to Text-to-SQL, but the training corpus (e.g., having task-specific training data) plays a more crucial role.

4.4.3 Few-shot Scenario with Open-source LLM. For few-shot scenario, Fig. 5 shows the performance of LLaMA-33B and Vicuna-33B with CR $_P$. We use DAIL Selection to select example as it is reported as the best strategy in Sec. 4.3. From this Figure, we can see that LLaMA-33B benefits more than Vicuna-33B, and achieves 36.4% exact-set-match accuracy with 5-shot Full-Information Organization examples. Regarding execution match accuracy, increasing number of examples benefits Text-to-SQL in most cases. Besides, among different organization strategies, Full-Information Organization outperforms other strategies in different k-shot scenarios, which achieves 51.5% execution accuracy with Vicuna-33B. Please refer to our full version [13] for more analysis in few-shot scenario.

Notably, in both zero-shot and few-shot scenarios, the open-source LLMs are far behind OpenAI LLMs. We will try to further enhance their performance with supervised fine-tuning.

4.4.4 Supervised Fine-tuning with Open-source LLM. To further enhance Open-source LLMs' performances, we explore supervised fine-tuning for Text-to-SQL. Similar to in-context learning, it may prefer different representations. Thus, we first fine-tune open-source LLMs on zero-shot training samples with different representations. Following the setting of supervised fine-tuning [32, 44], we block the gradients from prompt and only update weights with those from response (SQL queries). We use the train split in Spider,

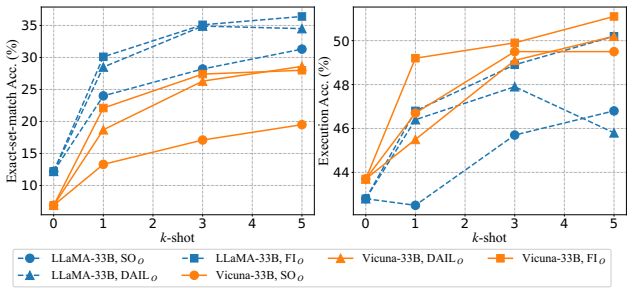


Figure 5: Few-shot evaluation with open-source LLMs on Spider-dev.

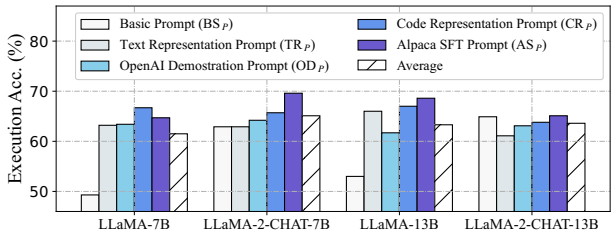


Figure 6: Zero-shot evaluation results on Spider-dev with different fine-tuned open-source LLMs.

which contains 8659 training samples. More training details refer to our full version paper [13].

Zero-shot Scenario. Fig. 6 shows the performance of supervised fine-tuning with various LLMs and question representations in zero-shot scenario. Compared with zero-shot performance before fine-tuning in Table 3 , their performances are greatly enhanced. By comparing different representations, Alpaca SFT Prompt show obvious advantages in supervised fine-tuning as it is designed for such scenario.

We also observe the gap among different representations and model scales becomes narrow. The possible reason is that after fine-tuning, LLMs learn to answer new Text-to-SQL questions without task instruction and foreign keys. In this experiment, the best performance on Spider is achieved by the combination of LLaMA-13B and Alpaca SFT Prompt, whose execution accuracy is 68.6%. As for larger LLM, the combination of LLaMA-33B and Code Representation Prompt achieves 69.1% execution accuracy.

In summary, supervised fine-tuning is quite beneficial for open-source LLMs in Text-to-SQL. In zero-shot scenario, fine-tuned LLaMA-13B and 30B are comparable to TEXT-DAVINCI-003.

Few-shot Scenario. After supervised fine-tuning, an important issue is: *Can we continue to enhance the performance of open-source LLM by adding contextual examples?* To answer this question, we evaluate fine-tuned LLaMA-7B and 13B with 0, 1, 3 and 5-shot prompts as shown in Table 4. We also add the evaluation results of original LLaMA-7B and 13B for clear comparison. Unexpectedly, the fine-tuned LLMs fail to learn from examples. Specifically, adding contextual examples in test prompts incurs sudden decrease in both exact-set-match and execution match accuracy, and adding more

Table 4: Few-shot evaluation results of supervised fine-tuned LLMs on Spider-dev.

LLM	Org.	0-shot		1-shot		3-shot		5-shot	
		EM	EX	EM	EX	EM	EX	EM	EX
LLaMA -7B	FI_O	3.1	13.0	23.4	30.1	23.7	30.3	24.7	30.9
	SO_O	3.1	13.0	13.3	21.4	15.2	24.1	15.3	25.0
	$DAIL_O$	3.1	13.0	18.5	25.4	22.1	28.1	22.6	29.3
+ SFT	FI_O	63.9	66.7	59.6	61.4	58.7	61.4	59.4	61.5
	SO_O	63.9	66.7	59.8	62.3	58.8	61.1	59.5	62.2
	$DAIL_O$	63.9	66.7	58.5	61.9	59.8	61.7	58.9	60.9
LLaMA -13B	FI_O	2.4	20.3	21.6	33.8	27.3	38.1	28.5	38.8
	SO_O	2.4	20.3	20.7	33.6	23.2	35.9	27.4	36.9
	$DAIL_O$	2.4	20.3	13.2	30.0	15.5	32.3	16.2	32.4
+ SFT	FI_O	62.7	67.0	61.9	67.1	60.5	65.0	60.9	65.0
	SO_O	62.7	67.0	61.9	66.2	60.1	64.6	60.2	65.2
	$DAIL_O$	62.7	67.0	62.5	66.5	60.6	66.0	61.3	66.4

examples is also unhelpful. A possible reason is that LLM overfits to zero-shot prompt, which makes examples unuseful.

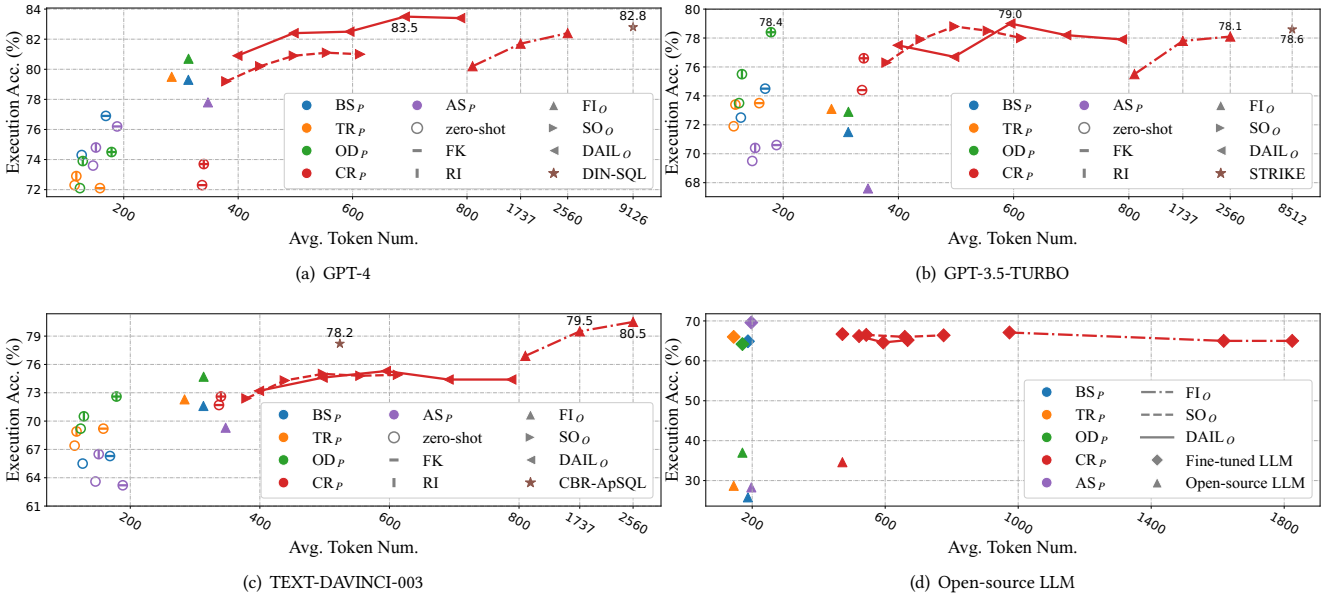
In summary, open-source LLMs demonstrate significant potential for Text-to-SQL tasks, particularly in supervised fine-tuning. Specifically, after fine-tuning, their performances are comparable to TEXT-DAVINCI-003 in zero-shot scenario. However, unlike OpenAI LLMs, fine-tuned LLMs fail to learn from contextual examples. The question of preserving in-context learning ability after fine-tuning remains to be explored in future studies.

4.5 Token Efficiency

Considering OpenAI LLMs are charged by token numbers, and LLMs' running time are proportional to token lengths, we underscore token efficiency in prompt engineering, which aims to achieve higher accuracy with less tokens. In this section, we review our experiments on Spider-dev in terms of token efficiency (For more efficiency analysis, please refer to our full version [13]). Specifically, for both OpenAI and open-source LLMs, we experimentally study the trade-off between execution accuracy and token numbers, and the token number is mainly affected by question representation and example organization. For example selection, we fix it as $DAIL_S$. Besides, we also include several state-of-the-art Text-to-SQL methods in our comparison, including DIN-SQL [35], STRIKE [27] and CBR-ApSQL [14].

Fig. 7 shows the comparison in terms of token efficiency. In zero-shot scenario, compared with rule implication, prompt with foreign keys generally achieve higher execution accuracy at the expense of more tokens. In few-shot scenario, comparing different organization strategies, FI_O are very inefficient, whose tokens numbers are several times that of $DAIL_O$ and SO_O. Comparing $DAIL_O$ and SO_O, $DAIL_O$ together with GPT-4 achieve the highest accuracy of 83.5%, yet having similar token cost with SO_O. Therefore, $DAIL_O$ are more efficient than SO_O and FI_O in terms of token.

Compared with other state-of-the-art solutions, DAIL-SQL outperforms DIN-SQL and STRIKE in terms of both accuracy and efficiency. While for CBR-ApSQL, it achieves 78.2% accuracy with TEXT-DAVINCI-003, but still lower than the optimal performance achieved by $DAIL_S$ + FI_O. Besides, for open-source LLM in Fig. 7(d),



(a) GPT-4 (b) GPT-3.5-TURBO

(c) TEXT-DAVINCI-003 (d) Open-source LLM

Figure 7: Token efficiency of different representations in Spider-dev for LLMs. We utilize different colors to represent different question representations and different shapes to denote different example organizations as well as the usage of foreign key information and rule implication. In particular, the overlap of shapes is used to indicate the usage of both foreign key information and rule implication. The rings stand for the prompts in zero-shot scenario and the stars stand for the previous SOTA results of few-shot methods in LLMs.

the LLMs fine-tuned on Text-to-SQL are much more efficient. However, as discussed in Sec. 4.4, adding examples is unhelpful for open-source LLMs, and even reduces their token efficiency.

In summary, token efficiency is a critical metric for real-world applications of LLMs on Text-to-SQL. In light of this, our approach, DAIL-SQL, offers a compelling solution that combines high execution accuracy with improved token efficiency. This makes it highly practical and suitable for real-world applications.

5 DISCUSSION

Based on our experiments, we can have some empirical insights:

- For question representation, Code Representation Prompt and OpenAI Demostration Prompt are recommended, and other information such as foreign key and rule implication can be very helpful.
- For example selection, the similarities of both natural language question and SQL query are important. These two similarities together are a good indicator for designing effective selection strategy.
- For example organization, if the adopted LLM is powerful enough, like GPT-4, presenting them question and SQL query pairs is an effective yet efficient choice. Otherwise, presenting them full information examples is suggested.
- For open-source LLM, having more parameters in LLMs benefits to Text-to-SQL task, but the training corpus plays a more crucial role. Besides, supervised fine-tuning is necessary and has considerable potential in Text-to-SQL task.

There are also some limitations in this paper. We fine-tune open-source LLMs with only the Spider training set, and additional Text-to-SQL data would further enhance LLMs. Besides, the databases in Spider and Spider-Realistic may be not large enough, and we believe some new challenges in effectiveness and efficiency will emerge if there are a mass of tables in Text-to-SQL task. Furthermore, the current evaluation metric prioritizes correctness over efficiency, and promoting LLM to generate efficient SQL among correct alternatives remains an important, unexplored question. We will keep working on these limitations and open questions.

6 CONCLUSIONS

In this paper, we conduct a systematical study on LLM-based Text-to-SQL from aspects of prompt engineering and supervised fine-tuning. We point out that existing in-context learning techniques for Text-to-SQL neglect the mapping between questions and queries, as well as the trade-off between example quality and quantity. To address these issues, we proposed DAIL-SQL, which refreshes the Spider leaderboard with 86.6% execution accuracy and ranks the first place. Regarding supervised fine-tuning, we demonstrate the great potentials of open-source LLMs for Text-to-SQL, underline the importance of training corpus and model scaling, and point out the degeneracy of in-context learning capability after fine-tuning. Further, we conduct an observation over existing solutions in terms of efficiency.All of these are open challenges and opportunities for future study. We hope that our work can provide a comprehensive study about Text-to-SQL, give some guidelines for real-world applications, and help people advance its frontiers.

REFERENCES

[1] Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Benjamin Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. 2021. A General Language Assistant as a Laboratory for Alignment. *CoRR* abs/2112.00861 (2021).

[2] LILY Group at Yale University. 2018. Spider 1.0, Yale Semantic Parsing and Text-to-SQL Challenge. https://yale-lily.github.io/spider.

[3] Christopher Baik, Zhongjun Jin, Michael J. Cafarella, and H. V. Jagadish. 2020. Duoquest: A Dual-Specification System for Expressive SQL Queries. In *Proceedings of the 2020 International Conference on Management of Data*. 2319–2329.

[4] Ursin Brunner and Kurt Stockinger. 2021. ValueNet: A Natural Language-to-SQL System that Learns from Database Information. In *37th IEEE International Conference on Data Engineering*. 2177–2182.

[5] 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*. 3977–3983.

[6] 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. *CoRR* abs/2305.11853 (2023).

[7] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. https://lmsys.org/blog/2023-03-30-vicuna/

[8] Naihao Deng, Yulong Chen, and Yue Zhang. 2022. Recent Advances in Text-to-SQL: A Survey of What We Have and What We Expect. In *Proceedings of the 29th International Conference on Computational Linguistics*. 2166–2187.

[9] 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*. 1337–1350.

[10] 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*. 4171–4186.

[11] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A Survey for In-context Learning. *CoRR* abs/2301.00234 (2023).

[12] 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. *CoRR* abs/2307.07306 (2023).

[13] 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. *CoRR* abs/2308.15363 (2023).

[14] Chunxi Guo, Zhiliang Tian, Jintao Tang, Pancheng Wang, Zhihua Wen, Kang Yang, and Ting Wang. 2023. A Case-Based Reasoning Framework for Adaptive Prompting in Cross-Domain Text-to-SQL. *CoRR* abs/2304.13301 (2023).

[15] 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 Conference of the Association for Computational Linguistics*. 4524–4535.

[16] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *The 10th International Conference on Learning Representations*.

[17] Binyuan Hui, Ruiying Geng, Lihan Wang, Bowen Qin, Yanyang Li, Bowen Li, Jian Sun, and Yongbin Li. 2022. S^2SQL: Injecting Syntax to Question-Schema Interaction Graph Encoder for Text-to-SQL Parsers. In *Findings of the Association for Computational Linguistics*. 1254–1262.

[18] George Katsogiannis-Meimarakis and Georgia Koutrika. 2023. A Survey on Deep Learning Approaches for Text-to-SQL. *VLDB J.* 32, 4 (2023), 905–936.

[19] Anirudh Khatry, Joyce Cahoon, Jordan Henkel, Shaleen Deep, K. Venkatesh Emani, Avrilia Floratou, Sumit Gulwani, Vu Le, Mohammad Raza, Sherry Shi, Mukul Singh, and Ashish Tiwari. 2023. From Words to Code: Harnessing Data for Program Synthesis from Natural Language. *CoRR* abs/2305.01598 (2023).

[20] Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. KaggleDBQA: Realistic Evaluation of Text-to-SQL Parsers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. 2261–2273.

[21] Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023. RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL. In *37th AAAI Conference on Artificial Intelligence*. 13067–13075.

[22] Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, and Yongbin Li. 2023. Graphix-T5: Mixing Pre-trained Transformers with Graph-Aware Layers for Text-to-SQL Parsing. In *37th AAAI Conference on Artificial Intelligence*. 13076–13084.

[23] Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, 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. *CoRR* abs/2305.03111 (2023).

[24] Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S. Yu. 2023. A Comprehensive Evaluation of ChatGPT's Zero-Shot Text-to-SQL Capability. *CoRR* abs/2303.13547 (2023).

[25] Hu Liu, Yuliang Shi, Jianlin Zhang, Xinjun Wang, Hui Li, and Fanyu Kong. 2023. Multi-hop Relational Graph Attention Network for Text-to-SQL Parsing. In *International Joint Conference on Neural Networks*. 1–8.

[26] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What Makes Good In-Context Examples for GPT-3?. In *Proceedings of Deep Learning Inside Out: The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*. 100–114.

[27] Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. 2023. Enhancing Few-shot Text-to-SQL Capabilities of Large Language Models: A Study on Prompt Design Strategies. *CoRR* abs/2305.12586 (2023).

[28] OpenAI. 2023. GPT-4 Technical Report. *CoRR* abs/2303.08774 (2023).

[29] OpenAI. 2023. Introducing ChatGPT. https://openai.com/blog/chatgpt. Last accessed on 2023-07-24.

[30] OpenAI. 2023. Rate limits. https://platform.openai.com/docs/guides/rate-limits/overview. Last accessed on 2023-07-24.

[31] OpenAI. 2023. SQL translate. https://platform.openai.com/examples/default-sql-translate. Last accessed on 2023-07-24.

[32] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training Language Models to Follow Instructions with Human Feedback. In *NeurIPS*.

[33] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only. *CoRR* abs/2306.01116 (2023).

[34] Octavian Popescu, Irene Manotas, Ngoc Phuoc An Vo, Hangu Yeo, Elahe Khorashani, and Vadim Sheinin. 2022. Addressing Limitations of Encoder-Decoder Based Approach to Text-to-SQL. In *Proceedings of the 29th International Conference on Computational Linguistics*. 1593–1603.

[35] Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. *CoRR* abs/2304.11015 (2023).

[36] Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Yu Cheng, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. RASAT: Integrating Relational Structures into Pretrained Seq2Seq Model for Text-to-SQL. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 3215–3229.

[37] Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, Fei Huang, and Yongbin Li. 2022. A Survey on Text-to-SQL Parsing: Concepts, Methods, and Future Directions. *CoRR* abs/2208.13629 (2022).

[38] Abdul Quamar, Vasilis Efthymiou, Chuan Lei, and Fatma Özcan. 2022. Natural Language Interfaces to Data. *Found. Trends Databases* 11, 4 (2022), 319–414.

[39] Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the Text-to-SQL Capabilities of Large Language Models. *CoRR* abs/2204.00498 (2022).

[40] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton-Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code Llama: Open Foundation Models for Code. *CoRR* abs/2308.12950 (2023).

[41] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 9895–9901.

[42] Jaydeep Sen, Chuan Lei, Abdul Quamar, Fatma Özcan, Vasilis Efthymiou, Ayushi Dalmia, Greg Stager, Ashish R. Mittal, Diptikalyan Saha, and Karthik Sankaranarayanan. 2020. ATHENA++: Natural Language Querying for Complex Nested SQL Queries. *Proc. VLDB Endow.* 13, 11 (2020), 2747–2759.

[43] Ruoxi Sun, Sercan Ö. Arik, Hootan Nakhost, Hanjun Dai, Rajarishi Sinha, Pengcheng Yin, and Tomas Pfister. 2023. SQL-PaLM: Improved Large Language Model Adaptation for Text-to-SQL. *CoRR* abs/2306.00739 (2023).

[44] 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.

[45] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *CoRR* abs/2302.13971 (2023).

[46] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael, Smith Ranjan, Subramanian Xiaoqing, Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. LLAMA2: Open Foundation and Fine-Tuned Chat Models. *CoRR* (2023).

[47] Immanuel Trummer. 2022. CodexDB: Synthesizing Code for Query Processing from Natural Language Instructions Using GPT-3 Codex. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2921–2928.

[48] 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*. 7567–7578.

[49] Lihan Wang, Bowen Qin, Binyuan Hui, Bowen Li, Min Yang, Bailin Wang, Binhua Li, Jian Sun, Fei Huang, Luo Si, and Yongbin Li. 2022. Proton: Probing Schema Linking Information from Pre-trained Language Models for Text-to-SQL Parsing. In *The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1889–1898.

[50] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*.

[51] Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018. SQL-to-Text Generation with Graph-to-Sequence Model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 931–936.

[52] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 8413–8426.

[53] 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*. 3911–3921.

[54] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. *CoRR* abs/2303.18223 (2023).

[55] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *CoRR* abs/2306.05685 (2023).

[56] Yanzhao Zheng, Haibin Wang, Baohua Dong, Xingjun Wang, and Changshan Li. 2022. HIE-SQL: History Information Enhanced Network for Context-Dependent Text-to-SQL Semantic Parsing. In *Findings of the Association for Computational Linguistics*. 2997–3007.

[57] 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*. 396–411.

[58] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR* abs/1709.00103 (2017).