Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs

Jinyang Li[1,♣ ‡] **Binyuan Hui**[2,♣], **Ge Qu**[1,♣], **Binhua Li**[2], **Jiaxi Yang**[2], **Bowen Li**[3], **Bailin Wang**[4], **Bowen Qin**[2], **Rongyu Cao**[2], **Ruiying Geng**[2], **Nan Huo**[1], **Xuanhe Zhou**[3], **Chenhao Ma**[5], **Guoliang Li**[3], **Kevin C.C. Chang**[6 †] **Fei Huang**[2], **Reynold Cheng**[1 †] **Yongbin Li**[2 †]
[1] The University of Hong Kong [2] DAMO Academy, Alibaba Group
[3] Tsinghua University
[4] Massachusetts Institute of Technology
[5] The Chinese University of Hong Kong (Shenzhen)
[6] University of Illinois at Urbana-Champaign
jl0725@connect.hku.hk, ckcheng@cs.hku.hk
binyuan.hby@alibaba-inc.com

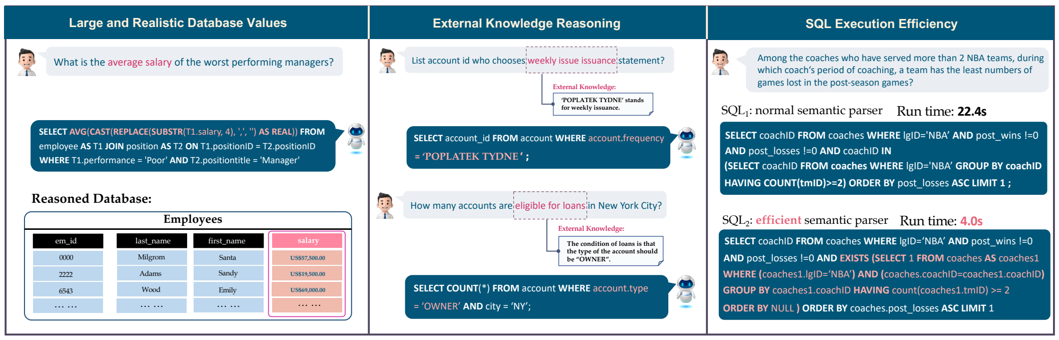
Abstract

Text-to-SQL parsing, which aims at converting natural language instructions into executable SQLs, has gained increasing attention in recent years. In particular, Codex and ChatGPT have shown impressive results in this task. However, most of the prevalent benchmarks, i.e., Spider, and WikiSQL, focus on database schema with few rows of database contents leaving the gap between academic study and real-world applications. To mitigate this gap, we present **BIRD**, a **BI**g benchmark for la**R**ge-scale **D**atabase grounded in text-to-SQL tasks, containing **12,751** pairs of text-to-SQL data and **95** databases with a total size of **33.4 GB**, spanning **37** professional domains. Our emphasis on database values highlights the new challenges of dirty database contents, external knowledge between NL questions and database contents, and SQL efficiency, particularly in the context of massive databases. To solve these problems, text-to-SQL models must feature database value comprehension in addition to semantic parsing. The experimental results demonstrate the significance of database values in generating accurate text-to-SQLs for big databases. Furthermore, even the most popular and effective text-to-SQL models, i.e. ChatGPT, only achieve 40.08% in execution accuracy, which is still far from the human result of 92.96%, proving that challenges still stand. Besides, we also provide an efficiency analysis to offer insights into generating text-to-efficient-SQLs that are beneficial to industries. We believe that BIRD will contribute to advancing real-world applications of text-to-SQL research. The leaderboard and source code are available: https://bird-bench.github.io/.

1 Introduction

Text-to-SQL parsing [57, 52, 53, 2, 54, 37], which focuses on transforming natural language into SQL queries, has attracted significant research interests from both academia and industry. This attention stems from its potential to empower non-expert data analysts in automatically extracting desired information from ubiquitous relational databases using natural language. Recent advances in neural

♣ Equal contribution.
‡Work done during an intern at Alibaba DAMO Academy.
†Corresponding authors.


Large and Realistic Database Values
External Knowledge Reasoning
SQL Execution Efficiency

What is the average salary of the worst performing managers?

```
SELECT AVG(CAST(REPLACE(SUBSTR(T1.salary, 4), ',', '') AS REAL)) FROM
employee AS T1 JOIN position AS T2 ON T1.positionID = T2.positionID
WHERE T1.performance = 'Poor' AND T2.positiontitle = 'Manager'
```

Reasoned Database:

Employees

em_id	last_name	first_name	salary
0000	Milgrom	Santa	US$57,500.00
2222	Adams	Sandy	US$19,500.00
6543	Wood	Emily	US$69,000.00
… …	… …	… …	… …

List account id who chooses weekly issue issuance statement?

External Knowledge: 'POPLATEK TYDNE' stands for weekly issuance.

```
SELECT account_id FROM account WHERE account.frequency
= 'POPLATEK TYDNE' ;
```

How many accounts are eligible for loans in New York City?

External Knowledge: The condition of loans is that the type of the account should be "OWNER".

```
SELECT COUNT(*) FROM account WHERE account.type
= 'OWNER' AND city = 'NY';
```

Among the coaches who have served more than 2 NBA teams, during which coach's period of coaching, a team has the least numbers of games lost in the post-season games?

SQL$_1$: normal semantic parser Run time: **22.4s**

```
SELECT coachID FROM coaches WHERE lgID='NBA' AND post_wins !=0
AND post_losses !=0 AND coachID IN
(SELECT coachID FROM coaches WHERE lgID='NBA' GROUP BY coachID
HAVING COUNT(tmID)>=2) ORDER BY post_losses ASC LIMIT 1 ;
```

SQL$_2$: efficient semantic parser Run time: **4.0s**

```
SELECT coachID FROM coaches WHERE lgID='NBA' AND post_wins !=0
AND post_losses !=0 AND EXISTS (SELECT 1 FROM coaches AS coaches1
WHERE (coaches1.lgID='NBA') AND (coaches.coachID=coaches1.coachID)
GROUP BY coaches1.coachID HAVING count(coaches1.tmID) >= 2
ORDER BY NULL ) ORDER BY coaches.post_losses ASC LIMIT 1
```

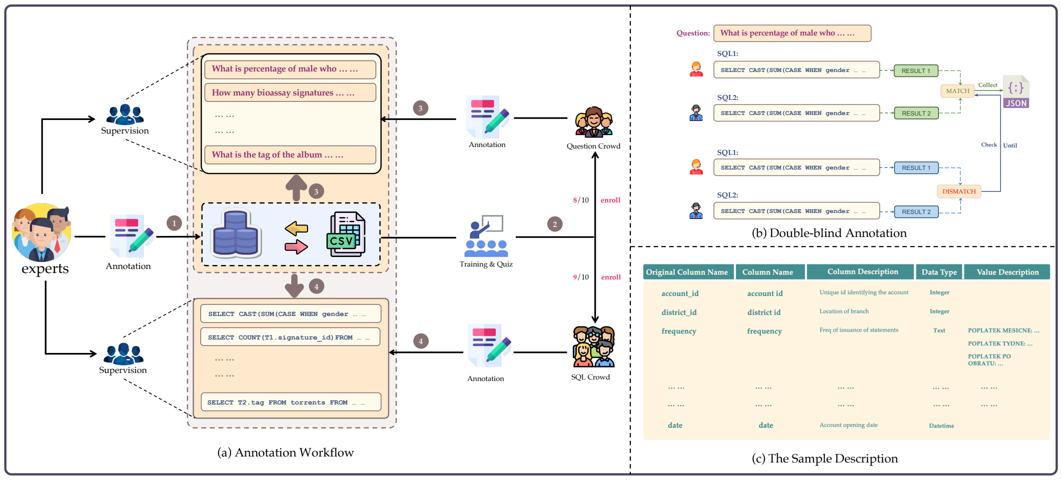
Figure 1: Examples of challenges in our **BIRD** benchmark. 1) databases contain values of noisy data types [14, 24, 19, 32]. In the left example, the average salary could be fetched by processing the data type from string (`TEXT` in SQLite) to float (`REAL` in SQLite) after deleting the special tokens, `"US$"` and `","`. 2) external knowledge and reasoning are required. In the middle example, models must handle that only `"OWNER"` accounts are eligible for loans. 3) query execution efficiency needs to be considered. In the right example, the adoption of more efficient SQL queries leads to significant gains in speed, which is of great value in industries.

models, including those based on large language models (LLMs), have led to impressive performance on existing benchmarks such as Spider [55] and WikiSQL [60]. For instance, the execution accuracy of the top-performing model in Spider leaderboard has increased from 53.5% [61] to 85.3% [35] over the past three years. The latest SOTA parser [35] in Spider benefits from the powerful understanding and coding capabilities of the large language model (LLM), and such excellent performance leads us to ask a question: ***Can LLM already serve as a database interface ?***

The answer is no, as shown in Figure.1, we discovered that current state-of-the-art models still struggle to generalize to more realistic situations characterized by large database sizes and noisy content. Besides, the mysteries hidden behind the huge database values require external knowledge and reasoning to reveal. Furthermore, existing benchmarks do not account for SQL execution efficiency, which holds significant practical importance in real-life applications, notably in the case of large databases. Motivated by these observations, we aim to develop a new text-to-SQL benchmark that better represents real-life scenarios and narrows the gap between experimental and practical settings.

In this work, we propose **BIRD**, a **BI**g Bench for La**R**ge-Scale **D**atabase Grounded in Text-to-SQLs for real-world applications. BIRD contains complex **12,751** examples of querying information over **95** big databases with a total size of **33.4 GB** spanning **37** professional domains. For training, we collected 80 open-source relational databases from real analysis platforms (Kaggle, Relation.vit); for evaluation, we curated 15 additional relational databases. Given these databases, we rely on crowdsourcing to collect natural language instructions and the corresponding SQLs. First, our database experts create a description file explaining all column names, abbreviated values, value types, and external knowledge for each database to help annotators better understand the database contents. Then we hire and train native speakers to ask questions facing these databases on one side; on the other side, a SQL annotation team consisting of data engineers and database students is recruited to generate SQLs to answer questions. To accommodate efficiency, we propose a new metric Valid Efficiency Score (VES) to evaluate the efficiency of generated SQLs in addition to the standard execution accuracy. To the best of our knowledge, BIRD is the first text-to-SQL benchmark to incorporate efficiency, promoting more efficient query methods within the context of massive and noisy database contents.

We evaluate the performance of state-of-the-art text-to-SQL parsers using two popular methodologies: fine-tuning with T5 [38], and in-context learning with large language models (LLMs) such as Codex [6] (`code-davinci-002`) and ChatGPT [33] (`gpt-3.5-turbo`). Our experimental results reveal that the current models struggle to generalize well. Specifically, the Spider SOTA model, which depends solely on the database schema, achieves execution accuracies of only 25.88% and 28.95% on the development and test sets, respectively. In comparison, the performance still lags far behind human performance, which we also provide in this benchmark. We encourage further research to address the more realistic settings presented in this benchmark.


experts
Annotation
1
Supervision
Supervision
What is percentage of male who … …
How many bioassay signatures … …
… …
… …
What is the tag of the album … …
3
CSV
4
SELECT CAST(SUM(CASE WHEN gender … …
SELECT COUNT(T1.signature_id) FROM … …
… …
… …
SELECT T2.tag FROM torrents FROM … …
3
Annotation
Question Crowd
Training & Quiz
2
8/10 enroll
9/10 enroll
4
Annotation
SQL Crowd
(a) Annotation Workflow
Question: What is percentage of male who … …
SQL1:
SELECT CAST(SUM(CASE WHEN gender … …
RESULT 1
SQL2:
SELECT CAST(SUM(CASE WHEN gender … …
RESULT 2
MATCH
Collect
JSON
Check
Until
SQL1:
SELECT CAST(SUM(CASE WHEN gender … …
RESULT 1
DISMATCH
SQL2:
SELECT CAST(SUM(CASE WHEN gender … …
RESULT 2
(b) Double-blind Annotation
(c) The Sample Description

Original Column Name	Column Name	Column Description	Data Type	Value Description
account_id	account id	Unique id identifying the account	Integer	
district_id	district id	Location of branch	Integer	
frequency	frequency	Freq of issuance of statements	Text	POPLATEK MESICNE: … POPLATEK TYDNE: … POPLATEK PO OBRATU: …
… …	… …	… …	… …	… …
… …	… …	… …	… …	… …
date	date	Account opening date	Datetime	

Figure 2: An Overview of the Text-to-SQL Annotation Workflow in (a). This figure depicts a four-step procedure. (1) The workflow begins with specialists assembling and producing databases and description files. (2) Experts then teach and evaluate crowdsourcing people, keeping only those who pass the evaluation. (3) Question annotators create a corpus of questions using databases and their corresponding description files. (4) SQL annotators produce SQL files, equipped with databases, descriptions, and questions. (b) and (c) also depict the Double-blind annotation procedure and an example of database descriptions.

2 Task Formulation & Annotations

2.1 Task Definition.

Text-to-SQL refers to the process of converting a natural language question $\mathcal{Q}$ into a SQL query $\mathbf{Y}$ capable of retrieving relevant data from a database. The database $\mathcal{D} = \langle \mathcal{C}, \mathcal{T} \rangle$, where $\mathcal{C}$ and $\mathcal{T}$ represent columns and tables respectively. External knowledge evidence $\mathcal{K}$ can be employed to improve the model's comprehension of the database contents. Finally, the text-to-SQL could be formulated as:

$$\mathbf{Y} = f(\mathcal{Q}, \mathcal{D}, \mathcal{K} \mid \boldsymbol{\theta}), \tag{1}$$

where $f(\cdot \mid \boldsymbol{\theta})$ is a model or neural network with the parameter $\boldsymbol{\theta}$. It is possible for external knowledge to be structured, as json or csv files in KaggleDBQA [25], and KnowSQL [10], or it could be unstructured, as evidence texts in [21]. In some cases, such as in Spider [55], no external knowledge is needed, which is represented as $\mathcal{K} = \{\varnothing\}$. It's important for models to determine whether or not to select knowledge by themselves when facing new questions [40], thus we provide all three formats of knowledge as bank in BIRD.

3 Dataset Construction

3.1 Database Source

It is difficult to collect databases with complex schemas and sufficient value. Due to privacy protection, the majority of synthetic databases are not available to individual users as open-source data. Earlier works [46, 55] that choose to self-design database schemas and value production can reduce this issue. Nonetheless, the value distribution and schemas may differ from real-world scenarios in this way. we obtain databases from three different sources to enrich real-world attributes. All of these databases contain real and large value distributions and are easily accessible with the appropriate licenses. 32% of our databases are sourced from Kaggle[*], a platform renowned for holding data science competitions with difficult, noisy content and schemas. Another 48% come from CTU Prague Relational Learning Repository[†], an open platform for machine learning research with multi-

[*] https://www.kaggle.com/
[†] https://relational.fit.cvut.cz/

relational data. The remaining 20% are built with a similar procedure to DuSQL [46], which entails acquiring open tables, synthesizing and standardizing schemas, and generating database constraints. Finally, we present 95 databases consisting of 69, 11, and 15 databases for training, development, and testing respectively. We classified our databases into 37 domains, covering blockchain, sports, health care, politics, etc. Similar to Spider, to safeguard the integrity of the databases, we have manually set foreign key and primary key constraints and calibrated integrated names for abbreviated table or column names. Our databases encompass schemas and values from a variety of domains. We anticipate that it will be a significant resource for researchers to explore domain generalization in semantic parsing tasks with large database contents.

3.2 Question Annotation

Entrance. We hire a group of native speakers of English with degrees above the bachelor's level and database-related knowledge to ask a variety of natural language questions regarding the contents of databases. To fulfill this objective, we have adopted the following procedure: (1). ER diagrams and database description files are documented to assist the annotators in understanding the databases; (2). we present the annotators with three databases from different domains and require them to generate 10 questions for each database; (3). these questions are then assessed by 3 text-to-SQL experts applying predefined rules. Those questions earning at least two votes are marked as valid. Only annotators capable of generating more than 8 valid questions per database are preserved. As a result, 11 native speakers contribute questions to BIRD.

Database Description File. The Database Description File is a crucial resource designed to aid annotators in comprehending the structure of databases, thereby allowing them to ask insightful questions about the contents. It offers two primary pieces of information regarding the database. (1). Full schema names: Database table and column names are frequently abbreviated, which are difficult to understand. (2). Value description: This is especially beneficial when phrases or tokens in a question are not string-matched to values in the database, as Figure 2 shows, it's very hard for the model to determine the meanings of `"POJISTNE"` stands for `insurance payment`. The Database Description File includes information on values via accessing original description files or websites in order to gather this information.

External Knowledge. In our study of professional data analysis, we find that external knowledge is required to map the natural language instructions into counterpart database elements. Therefore, we collect and classify this knowledge into four categories: (1). Numeric Reasoning Knowledge: this category refers to the mathematical computation required for certain SQL operations. In our benchmark, we present 8 basic math operations[‡], including 4 operations with additional inference as [7]: `MINUS`, `ADDITION`, `DIVISION`, `MULTIPLY`. BIRD also contains compositional operations over basic ones, such as the computation of percentages, ratios, etc. (2). Domain Knowledge: this category consists of domain-specific knowledge that is utilized to generate SQL operations [10, 59]. For instance, a business analyst in the banking business may require knowledge of financial indicators such as return on investment and net income in order to generate effective SQL queries. (3). Synonym Knowledge: this category includes words or expressions that have the same or similar meanings. In the context of our research, synonym knowledge is used to ensure that questions can be correctly converted to SQLs, regardless of how they are phrased differently [11]. (4). Value Illustration: we also provide detailed descriptions of databases values, including value types, value categories; and the mapping combinations of columns and values that correspond to entities, for example: `"center"` can be represented by `"pos = C"` in the database `professional_basketball`. This extensive description will assist annotators in comprehending the database's structures and contents.

3.3 SQL Annotation

Entrance. With the purpose of enhancing the quality of our SQL queries, we assemble a team of skilled data engineers and database students. The team undergoes rigorous testing through the text-to-SQL evaluation process, which assesses their capability of generating SQL queries for a

[‡]`AVERAGE`, `MIN`, `MAX`, `SUM` belong to basic Aggregation operation in our taxonomy, and `COUNT` is a specific category as shown in 3.

Table 1: An overview comparison between BIRD and other cross-domain text-to-SQL benchmarks. In SQL, `Function` pertains to the window function. `Knowledge` refers to whether or not this dataset necessitates knowledge reasoning from the model. `Efficiency` refers to whether or not this dataset takes into consideration execution efficiency.

Dataset	# Example	# DB	# Table/DB	# Row/DB	Function	Knowledge	Efficiency
WikiSQL [60]	80,654	26,521	1	17	✗	✗	✗
Spider [55]	10,181	200	5.1	2K	✗	✗	✗
KaggleDBQA [25]	272	8	2.3	280K	✗	✓	✗
BIRD	12,751	95	7.3	549K	✓	✓	✓

variety of questions facing different domains of databases. Each annotator is asked to answer 10 questions, and only those who score at least 9 out of 10 will be qualified to annotate SQL queries for BIRD.

Double-Blind Annotation. As shown in Figure 2 (b), we employ a double-blind technique [43] for SQL annotation. This approach involves two independent SQL annotators who generate SQLs for the same question without discussion. The annotated SQLs are executed in databases, and those yielding identical results are gathered[§]. Otherwise, the SQLs are checked with experts until a consensus is reached. Double-blind procedures can dramatically reduce the SQL annotation error rate, as there is a small probability for two annotators to generate the same incorrect results when working with databases with large contents. The most well-formed and efficient SQL[¶] selected by experts for each question is picked for the BIRD corpus, and the knowledge evidence sentences are recorded for each SQL if utilized.

Examination. After a meticulous double-blind procedure, experts evaluate each text-to-SQL pair to ensure the highest quality of data. The evaluation process includes examining each data in two dimensions: SQL validness, and text-knowledge-SQL alignment. Firstly, the SQL validness will be confirmed that each SQL is executable and can return a valid result from the database. The "valid result" refers to the set of results that is not `"NULL"`. If the executed result set is `"NULL"`, experts will make slight changes to the conditions of the questions until the associated SQLs can provide a valid result set. Secondly, text-knowledge-SQL alignment is involved to ensure that each SQL can be generated with the given texts and knowledge evidence. If the evidence is insufficient to generate the SQL or contains errors, experts will be in charge of correcting them.

Difficulty. In order to help researchers deeply analyze model performance in various text-to-SQL case levels, we class all examples as `simple`, `moderate`, and `challenging`. Spider computed difficulty mainly based on SQL complexity. However, we find that additional factors, such as question comprehension, schema linking, external knowledge acquisition, and reasoning, also influence model and human performance when translating texts to SQLs. Therefore, each SQL annotator is required to evaluate examples based on these factors, and experts conclude the ratings to divide examples into the three aforementioned difficulty levels, providing a more comprehensive difficulty analysis on text-to-SQL tasks.

4 Data Statistics

4.1 Overall Statistics

Table 1 presents an overview comparison between BIRD and other text-to-SQL benchmarks. As the statistics demonstrate, BIRD is a large-scale cross-domain benchmark, covering a variety of domains. BIRD is also a SQL playground, which incorporates the usage of window functions, knowledge

[§]To reduce the ambiguity of use of `DISTINCT`, we compare HashMap outcomes instead of lists.

[¶]Definitely, this procedure can also serve to enhance the diversity of SQL queries corresponding to individual questions, thereby reflecting the diverse perspectives on the same questions. Considering the burden of quality verification on pairwise SQLs, we just open-source the optimal SQLs as the ground truth at this time. We will open-source full pairwise SQLs if this benchmark receives enough attention.

reasoning, and efficiency evaluation in SQL generation. Moreover, the databases included in BIRD are the most extensive, making it an ideal testing base for bridging the gap between academic use and real-world applications of text-to-SQL.

4.2 Question Statistics

Figure 3 displays the extensive question type statistics of BIRD. To facilitate a deeper understanding of the distinctions between questions in BIRD and those from other datasets, we classify questions into two macro-categories: Fundamental Type and Reasoning Type, and each contains 4-5 micro-categories in detail. The `Fundamental Type` contains similar questions to those found in other text-to-SQL benchmarks. The `Reasoning Type` contains questions that require external knowledge grounding, which is exclusive to BIRD. As shown in Figure 3, the majority of questions are marked as match-based, indicating that schema linking, referring to aligning entities in questions to tables or columns [44, 1], continues to be the most important topic of text-to-SQL problems, even in the harder scenarios. Among the `Reasoning Type` of questions, there are ample examples that require domain knowledge and numeric reasoning, making BIRD questions especially challenging. In addition, 70% of the questions need value illustrations. This indicates that more real-world questions in text-to-SQL applications demand a thorough understanding of database values, which is consistent with our motivation for creating the BIRD benchmark.

4.3 Database Statistics

In the BIRD, we investigate the distribution of database domains, database size, and value types. Figure 4 (a) presents a detailed distribution of domains and their counterpart databases in a sunburst diagram for both training and development sets. The width of each semi-circle corresponds to the number of text-to-SQL pairs using the respective database. Figure 4 (a) also shows the size distributions of databases. The deeper color means a larger size of databases, and vice versa. For example, the database `Donor` is the largest database with 4.5 GB in this dataset. Furthermore, we observe from Figure 4 (b) that a considerable proportion of BIRD's data comprises date-related values. Considering that real-world applications often rely on time-sensitive data [26], the prevalence of datetime expressions and their related questions in the dataset we collected highlights the practical purposes.

4.4 SQL Statistics

Our study provides insights into the complexity and diversity of SQLs used in the BIRD. As illustrated in Figure 5, we present a comprehensive distribution analysis of SQLs across four dimensions. `No. Toks / SQL` and `No. JOINs / SQL` demonstrate the intricacy and requirements of multi-table reasoning of the SQLs in BIRD. `No. of Keywords` and `No. n-grams / SQL (n=3)` serve as the support for the diverse patterns of SQLs contained within the dataset since we decouple the question and SQL annotation procedures to make the situation more realistic [5].

5 Evaluation Metrics

In contexts of practical data analysis, text-to-SQL models are prioritized for delivering expected results accurately and efficiently. Thus we provide two metrics in BIRD, execution accuracy (EX) and valid efficiency score (VES) to evaluate text-to-SQL parsers confronted with real-world scenarios with large database contents.

(1) Execution Accuracy (EX): EX is defined as the proportion of questions in the evaluation set for which the execution results of both the predicted and ground-truth inquiries are identical, relative to the overall number of queries [37]. Considering the result set as V_n executed by the n^{th} ground-truth SQL Y_n, and the result set $\hat{V}_n$ executed by the predicted SQL $\hat{Y}_n$, EX can be computed by:

$$\text{EX} = \frac{\Sigma_{n=1}^{N} \mathbb{1}(V_n, \hat{V}_n)}{N}, \tag{2}$$

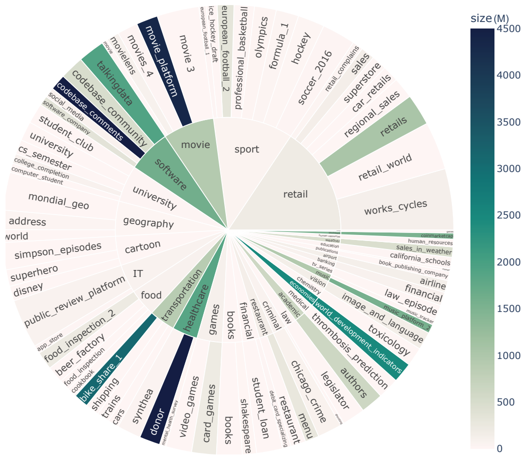
where $\mathbb{1}(\cdot)$ is an indicator function, which can be represented as:

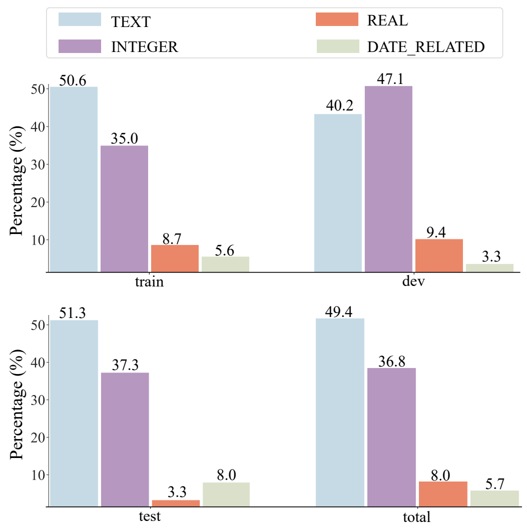
$$\mathbb{1}(V, \hat{V}) = \begin{cases} 1, & V = \hat{V} \\ 0, & V \neq \hat{V} \end{cases} \tag{3}$$

Question Type	Sub Type	Question / SQL	Percentage
Fundamental Type	Match-based	How many gas stations in CZE has Premium gas? `SELECT COUNT(GasStationID) FROM gasstations WHERE Country = 'CZE' AND Segment = 'Premium'`	83.9 %
	Ranking	What are the titles of the top 5 posts with the highest popularity? `SELECT Title FROM posts ORDER BY ViewCount DESC LIMIT 5`	20.3 %
	Comparison	How many color cards with no borders have been ranked higher than 12000 on EDHRec? `SELECT COUNT(id) FROM cards WHERE edhrecRank > 12000 AND borderColor = 'borderless'`	16.7 %
	Counting	How many of the members' hometowns are from Maryland state? `SELECT COUNT(T2.member_id) FROM zip_code AS T1 INNER JOIN member AS T2 ON T1.zip_code = T2.zip WHERE T1.state = 'Maryland'`	30.4 %
	Aggregation	What is the average height of the superheroes from Marvel Comics? `SELECT AVG(T1.height_cm) FROM superhero AS T1 INNER JOIN publisher AS T2 ON T1.publisher_id = T2.id WHERE T2.publisher_name = 'Marvel Comics'`	15.7 %
Reasoning Type	Domain Knowledge	Name the ID and age of patient with two or more laboratory examinations which show their hematoclit level exceeded the normal range. `SELECT T1.ID, STRFTIME('%Y', CURRENT_TIMESTAMP) - STRFTIME('%Y', T1.Birthday) FROM Patient AS T1 INNER JOIN Laboratory AS T2 ON T1.ID = T2.ID WHERE T1.ID IN ( SELECT ID FROM Laboratory WHERE HCT > 52 GROUP BY ID HAVING COUNT(ID) >= 2 )`	23.6 %
	Numeric Computation	Among the posts with a score of over 20, what is the percentage of them being owned by an elder user? `SELECT CAST(SUM(IIF(T2.Age > 65, 1, 0)) AS REAL) * 100 / count(T1.Id) FROM posts AS T1 INNER JOIN users AS T2 ON T1.OwnerUserId = T2.Id WHERE T1.Score > 20`	24.5 %
	Synonym	How many clients opened their accounts in Jesenik branch were women ? (female) `SELECT COUNT(T1.client_id) FROM client AS T1 INNER JOIN district AS T2 ON T1.district_id = T2.district_id WHERE T1.gender = 'F' AND T2.A2 = 'Jesenik'`	7.2 %
	Value Illustration	Among the weekly issuance accounts, how many have a loan of under 200000? `SELECT COUNT(T1.account_id) FROM loan AS T1 INNER JOIN account AS T2 ON T1.account_id = T2.account_id WHERE T2.frequency = 'POPLATEK TYDNE' AND T1.amount < 200000`	70.1 %

Figure 3: Questions in the BIRD contain two main categories. The `Fundamental Type` of questions are comparable to other text-to-SQL benchmarks. The `Reasoning Type` of questions requires external knowledge grounding to answer.

(2) Valid Efficiency Score (VES): VES is designed to measure the efficiency of valid SQLs generated by models. It is worth noting that the term "valid SQLs" refers to predicted SQL queries whose result sets align with those of the ground-truth SQLs. Any SQL queries that fail to execute with the correct values will be declared invalid since they are totally useless if they cannot fulfill the user requests, regardless of their efficiency. In this case, the VES metric could consider both the efficiency and accuracy of execution results, providing a comprehensive evaluation of a model's performance.


a) Database domain distribution w/ size



b) Database value type distribution

Figure 4: This is a comprehensive database distribution in the BIRD. a) shows the domain and size distribution of each database. And b) shows the data type distribution of databases.

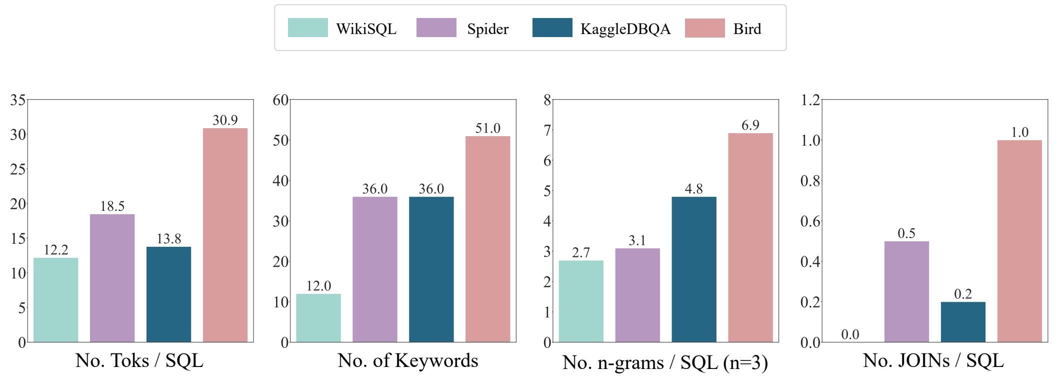


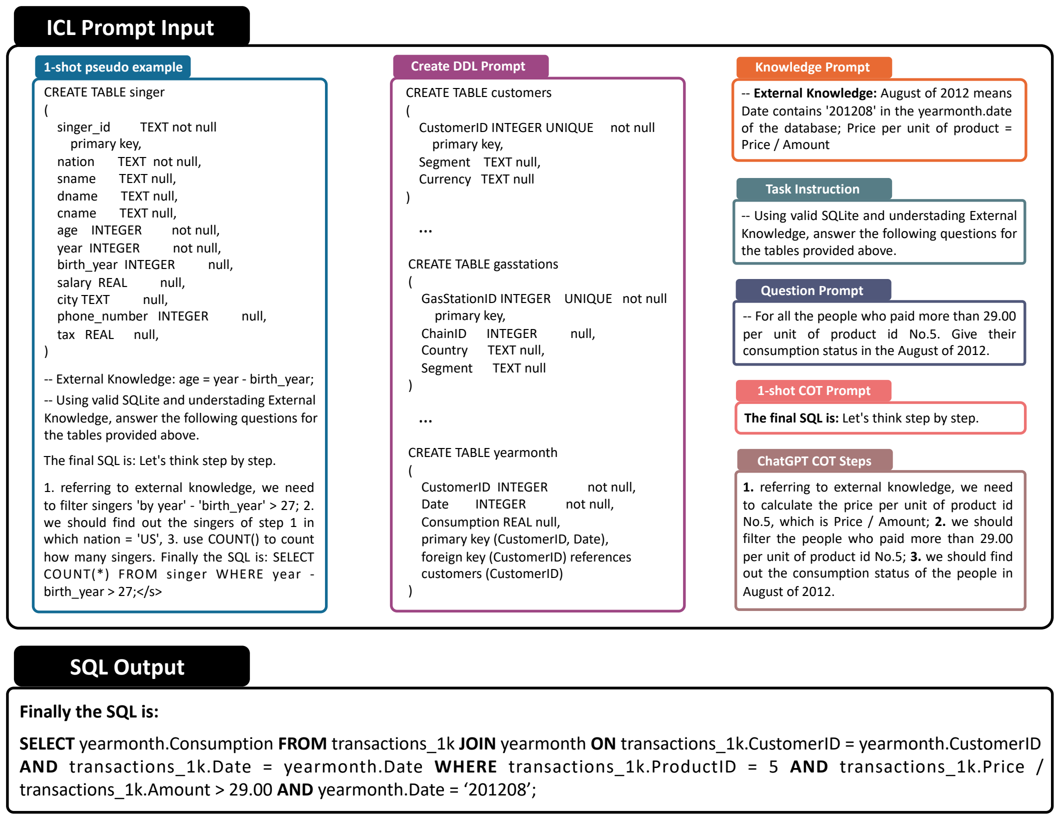
Figure 5: A comparative statistical analysis of SQL queries in the BIRD dataset and other cross-domain text-to-SQL benchmarks.

Formally, the VES can be expressed as:

$$\text{VES} = \frac{\Sigma_{n=1}^{N} \mathbb{1}(V_n, \hat{V}_n) \cdot \mathbf{R}(Y_n, \hat{Y}_n)}{N}, \quad \mathbf{R}(Y_n, \hat{Y}_n) = \sqrt{\frac{\mathbf{E}(Y_n)}{\mathbf{E}(\hat{Y}_n)}} \tag{4}$$

where $\mathbf{R}(\cdot)$ denotes the relative execution efficiency of predicted SQL in comparison to ground-truth SQL, allowing for machine status-related uncertainty. $\mathbf{E}(\cdot)$ is a function to measure the absolute execution efficiency for each SQL in a given environment[‖]. Furthermore, we incorporate the square root function to minimize random instances which are super more rapid or slower than the ground-truth SQLs, i.e. more than 500 times faster or slower. This operation aims to promote the model that can consistently generate efficient SQL predictions by learned expertise of database knowledge, rather than relying on arbitrary attempts and guesses for specific scenarios. Here, the efficiency can refer to running time, throughput, memory cost, or merged metrics. In BIRD, we consider the running time as the main metric of efficiency at this time.

[‖]In BIRD evaluation, we run 100 times for each SQL in the same CPU and evaluate average results after dropping the outliers.



```
CREATE TABLE singer
(
  singer_id     TEXT not null
     primary key,
  nation     TEXT  not null,
  sname      TEXT null,
  dname      TEXT null,
  cname      TEXT null,
  age   INTEGER       not null,
  year  INTEGER       not null,
  birth_year  INTEGER        null,
  salary  REAL       null,
  city TEXT       null,
  phone_number  INTEGER       null,
  tax   REAL     null,
)
```

-- External Knowledge: age = year - birth_year;

-- Using valid SQLite and understading External Knowledge, answer the following questions for the tables provided above.

The final SQL is: Let's think step by step.

1. referring to external knowledge, we need to filter singers 'by year' - 'birth_year' > 27; 2. we should find out the singers of step 1 in which nation = 'US', 3. use COUNT() to count how many singers. Finally the SQL is: SELECT COUNT(*) FROM singer WHERE year - birth_year > 27;</s>

```
CREATE TABLE customers
(
  CustomerID INTEGER UNIQUE    not null
     primary key,
  Segment   TEXT null,
  Currency  TEXT null
)

...

CREATE TABLE gasstations
(
  GasStationID INTEGER   UNIQUE   not null
     primary key,
  ChainID     INTEGER         null,
  Country     TEXT null,
  Segment     TEXT null
)

...

CREATE TABLE yearmonth
(
  CustomerID  INTEGER         not null,
  Date       INTEGER         not null,
  Consumption REAL null,
  primary key (CustomerID, Date),
  foreign key (CustomerID) references
  customers (CustomerID)
)
```

-- **External Knowledge:** August of 2012 means Date contains '201208' in the yearmonth.date of the database; Price per unit of product = Price / Amount

-- Using valid SQLite and understading External Knowledge, answer the following questions for the tables provided above.

-- For all the people who paid more than 29.00 per unit of product id No.5. Give their consumption status in the August of 2012.

The final SQL is: Let's think step by step.

1. referring to external knowledge, we need to calculate the price per unit of product id No.5, which is Price / Amount; **2.** we should filter the people who paid more than 29.00 per unit of product id No.5; **3.** we should find out the consumption status of the people in August of 2012.

Finally the SQL is:

SELECT yearmonth.Consumption **FROM** transactions_1k **JOIN** yearmonth **ON** transactions_1k.CustomerID = yearmonth.CustomerID **AND** transactions_1k.Date = yearmonth.Date **WHERE** transactions_1k.ProductID = 5 **AND** transactions_1k.Price / transactions_1k.Amount > 29.00 **AND** yearmonth.Date = '201208';

Figure 6: The detailed prompt design for implementation of ChatGPT + KG + COT.

6 Experiments

6.1 Baseline Models

We present the performance of two types of baseline models in BIRD. The first type of model is based on fine-tuning (FT) techniques, which outputs SQL tuning all parameters of language models via annotated data. On the other hand, in-context learning (ICL) is employed in the second type of model. In FT models, we select T5 [38] as the main baseline models. While seq2AST-based methods [44, 4] are also effective in text-to-SQL, actually their grammar rules utilized during decoding are constrained on specific datasets [26]. For ICL-based models, we provide zero-shot results of Codex, ChatGPT and its Chain-Of-Thought (COT) demonstration, which are very good at zero-shot text-to-SQL generation.

FT-based Models. T5 is a strong and versatile pre-trained language model (PLM) for text-to-text generation that has achieved state-of-the-art performance in a variety of semantic parsing tasks, including text-to-SQL. We concatenate the question with serialized database schema as input [41, 51, 42]. And SQL can be fetched in an end-to-end fashion by easily fine-tuning.

ICL-based Models. Codex (`code-davinci-002`[**][††]) and ChatGPT (`gpt-3.5-turbo`) are popular and powerful large-scale pre-trained language models (LLMs) for code generation driven by ICL. It can produce multiple types of codes, including SQL, from human instructions without additional training. We employ programming-based prompts, as described in [39], to collect results by calling the API. Given that models are not allowed to unseen databases and ground-truth SQLs in the evaluation set, a zero-shot generation strategy is the most appropriate. Moreover, to investigate the impact of multi-step reasoning of LLMs on BIRD, we implement the Chain-Of-Thought (COT)

[**]https://openai.com/blog/openai-api
[††]It's still valid at the time of writing.

technique [50] by easily adding the prompt sentence "`Let's think step by step.`" before the generation of SQLs [22]. However, we find out the output of ChatGPT is too uncertain with many unexpected explanations, thus we provide a 1-shot pseudo example for ChatGPT to learn the procedure of thinking and output format. The detailed prompt design is shown in Figure 6

Knowledge Fusion. In the baseline implementation, we naively concatenate the knowledge evidence sentences with questions and database schemas, but we can observe a significant improvement by this easy method. More complicated and effective strategy of knowledge grounding for ChatGPT and T5 would be an important future topic. The knowledge evidence sentences are concluded to the external knowledge provided by annotators as described in Section 3.2.

6.2 Execution Accuracy Analysis

As shown in the Table 2, we implement each baseline model for both two scenarios. The first is **NOT** to provide the ground truth knowledge evidence sentence for each text-to-SQL data. The other testing bed is to provide knowledge evidence and make tex-to-SQL models do knowledge grounding by themselves, which is more challenging.

Results w/o Knowledge Evidence. As shown in the Table 3, the performance of FT models T5 perform worse obviously than ICL-based LLMs in both EX and VES. This indicates that LLM with a huge number of parameters and emergent capability [49] can have a better self-knowledge grounding capability and pre-trained SQL knowledge than FT smaller models with less than 5B parameters. After being equipped with COT, ChatGPT can perform better, since multi-step reasoning is beneficial to it when the knowledge and data is low-resource.

Results w/ Knowledge Evidence. After being easily fed with the external knowledge about the database contents, all models have a clear improvement across the different difficulty levels as shown in Table 2 and Table 4. This indicates that knowledge evidence in BIRD is effective and instructive for model to better understand the database contents. All these performance gains can illustrate that the database contents are very important to text-to-SQL models when facing more real cases. Despite this, we observe a decline or limited improvements in performance for ChatGPT + knowledge grounding of database contents annotated by human experts for COT version. We hypothesis that the internal multi-step knowledge reasoning of LLMs is not compatible with the way of external knowledge in this situation. Therefore, the development of a method that effectively combines the strong multi-step self-reasoning capabilities of LLMs with external knowledge reasoning coherently presents a promising future direction [30]. This innovation may pave the way for the LLM plugin that facilitates integration of different sources of knowledge for more reliable results.

Table 2: The Execution Accuracy (EX) of SOTA text-to-SQL models in BIRD. The human performance is also provided.

Models	**Development Data**		**Testing Data**	
	w/o knowledge	**w/ knowledge**	**w/o knowledge**	**w/ knowledge**
		FT-based		
T5-Base	6.32	11.54 (+5.22)	7.06	12.89 (+5.83)
T5-Large	9.71	19.75 (+10.04)	10.38	20.94 (+10.56)
T5-3B	10.37	23.34 (+12.97)	11.17	24.05 (+12.88)
		ICL-based		
Codex	25.42	34.35 (+8.93)	24.86	36.47 (+11.61)
ChatGPT	24.05	**37.22** (+13.17)	26.77	39.30 (+12.53)
ChatGPT + COT	**25.88**	36.64 (+10.76)	28.95	40.08 (+11.24)
Human Performance	-	-	**72.37**	**92.96** (+20.59)

6.3 Efficiency Analysis

As discussed in Section 5, we introduced a novel sub-task, text-to-efficient-SQL, and proposed the VES metric for its evaluation. In this section, we at first present overall efficiency performance of

Table 3: The Valid Efficiency Score (VES) of SOTA text-to-SQL models in BIRD. The human performance is also presented.

Models	**Development Data**		**Testing Data**	
	w/o knowledge	**w/ knowledge**	**w/o knowledge**	**w/ knowledge**
FT-based				
T5-Base	7.78	12.90 (+5.12)	8.97	14.71 (+5.74)
T5-Large	9.90	22.74 (+12.84)	12.25	25.00 (+12.75)
T5-3B	13.62	25.57 (+11.95)	15.17	27.80 (+12.63)
ICL-based				
Codex	**33.37**	43.41 (+10.04)	35.40	41.60 (+6.20)
ChatGPT	27.97	**43.81** (+15.84)	36.68	51.40 (+14.72)
ChatGPT + COT	32.33	42.30 (+9.97)	49.69	56.56 (+6.87)
Human Performance	-	-	**70.36**	**90.27** (+19.91)

Model	Dev Set				Test Set			
	simple	moderate	challenging	total	simple	moderate	challenging	total
(EX) ChatGPT	31.08	13.29	12.08	24.05	35.41	19.46	12.28	26.77
(EX) ChatGPT + KG	**45.44**	**26.14**	**19.01**	**37.22**	**49.21**	**31.89**	**20.70**	**39.30**
(VES) ChatGPT	36.20	15.43	14.42	27.97	50.09	24.71	15.39	36.68
(VES) ChatGPT + KG	**54.71**	**28.16**	**22.80**	**43.81**	**65.06**	**41.21**	**25.81**	**51.40**

Table 4: The Execution Accuracy (EX) and Valid Efficiency Score (VES) are presented for both the ChatGPT model and its version with knowledge grounding (KG), taking into consideration development and testing datasets.

baseline models. Afterwards, we investigate two prospective approaches for enhancing text-to-SQL systems to generate more efficient SQL queries.

General Results. According to the Table 3, we can observe that the text-to-SQL models with higher EX can more possibly achieve higher VES. This relationship can be explained by the prerequisite that text-to-SQL models must accurately predict results in order to attain a higher VES, which fulfills the pratical purpose.

Two-Stage Optimization. Intuitively, we can decompose the goal of text-to-efficient-SQL into two sub-procedures. The first is semantic parsing procedure as previous text-to-SQL task, which only considers how to convert questions into SQLs accurately. The second procedure is SQL optimization to rewrite the SQLs to be more efficient while keeping the same results [63]. In order to demonstrate the effectiveness of this workflow, we randomly pick 10 examples from the development set in which ChatGPT accurately predicts the results, then our experts rewrite them into more efficient SQLs by query optimization rules [29, 34, 64].

Given that all these SQLs can fetch the correct results generated by ChatGPT, then we only present the time saving percentage to indicate the effectiveness of query optimization. The detailed analysis with examples is shown in the Figure 7. It includes the optimized SQLs, running time comparison, and reasons to describe the counterpart rule. The future work also could be a one-stage learning that can take into consideration the value descriptions of databases, while generating SQLs.

Embodied Database. In BIRD, We pioneer the concept of an embodied database allowing models to be aware of data types and distributions by producing SQL queries that interact with databases. This method establishes the groundwork for creating more effective and efficient SQL queries. For example, based on the cases from the Figure 7, the SQL effciency could also be improved obviously after setting indexes in the database.

Query Rewriting

Ex1.1 Question:
List out the age of users who located in Vienna, Austria obtained the badge?

ChatGPT SQL:
```
SELECT Age FROM users WHERE Location = 'Vienna, Austria' AND Id IN (SELECT UserId FROM
badges)
```

Optimized SQL: (time-saving percentage: **99.92%**)
```
SELECT u.Age FROM users AS u INNER JOIN badges AS b ON u.Id = b.UserId WHERE u.Location
= 'Vienna, Austria'
```

Take Away:
By applying a **JOIN** operation instead of a subquery with **IN** can improve efficiency, as the database may execute the **JOIN** and filtering processes concurrently in just one operation **without** the need to store the intermediate results to filter primary query.

Ex1.2 Question:
How many of the members' hometowns are from Maryland state?

ChatGPT SQL:
```
SELECT COUNT(*) FROM member INNER JOIN zip_code ON member.zip = zip_code.zip_code WHERE
zip_code.state = 'Maryland'
```

Optimized SQL: (time-saving percentage: **67.93%**)
```
SELECT COUNT(member.member_id) FROM member INNER JOIN zip_code ON member.zip =
zip_code.zip_code WHERE zip_code.state = 'Maryland'
```

Take Away:
Utilizing the **COUNT** function on a **NOT-NULL** column, as opposed to **COUNT(*)**, can increase time efficiency. This rewritten SQL enables the database to count **NOT-NULL** values within a single column, rather than compute all rows including those with **NULL** values. Usually, the primary key column is selected as this **NOT-NULL** column.

Ex1.3 Question:
Who is the owner of the account with the largest loan amount?

ChatGPT SQL:
```
SELECT c.client_id FROM client c INNER JOIN disp d ON c.client_id = d.client_id INNER
JOIN loan l ON d.account_id = l.account_id ORDER BY l.amount DESC LIMIT 1
```

Optimized SQL: (time-saving percentage: **62.39%**)
```
SELECT c.client_id FROM client c INNER JOIN disp d ON c.client_id = d.client_id
INNER JOIN loan l ON d.account_id = l.account_id WHERE l.amount = ( SELECT MAX(amount)
FROM loan)
```

Take Away:
In an unindexed environment, employing the **MAX** function can potentially yield faster results since it avoids the need for **sorting**, which could run against a large table.

Adding Indexes to Database

Ex2.1 Question:
How many accounts are there in the district of \"Pisek\"?

ChatGPT SQL:
```
SELECT COUNT(*) FROM account a INNER JOIN district d ON a.district_id = d.district_id
WHERE d.A2 = 'Pisek'
```

Added Indexes: (time-saving percentage: **87.27%**)
```
CREATE INDEX account_district_id_index ON account(district_id);
CREATE UNIQUE INDEX district_district_id_uindex ON district(district_id);
```

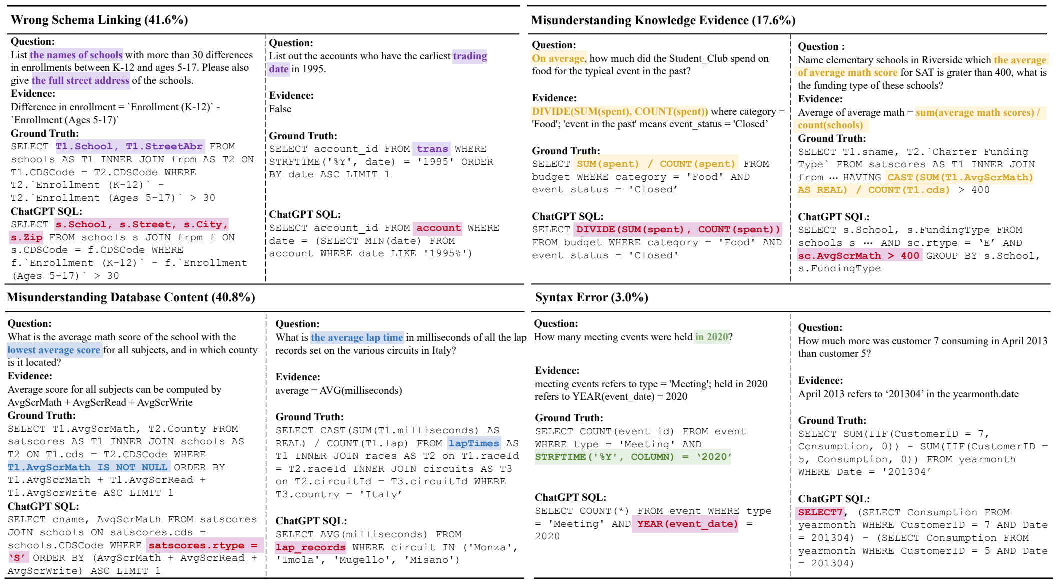
Take Away:
Adding **indexes** into a database can significantly increase the speed of SQL queries because it creates a data structure that enables the database engine to quickly locate rows that match specific criteria instead of **scanning** the entire table.

Figure 7: Two possible solutions and explanations to improve efficiency are presented. The first batch of examples shows how to optimize SQL efficiency by rewriting SQL based on rules. The last example is to show that adding indexes to databases can also improve SQL efficiency without rewriting them.

6.4 Human Performance

In order to active the efforts of text-to-SQL studies to achieve an application-level performance in the challenging but closer to the real-world scenarios, we also provide human performance in BIRD. The Table 2, Table 3 shows that there's still a huge gap between even SOTA text-to-SQL models and human performance.

The procedure of collecting human performance is still rigorous. During the annotation, all data is divided into 10 batches for better management and error tracks by experts. The first 8 batchs of



Wrong Schema Linking (41.6%)

Question:
List the names of schools with more than 30 differences in enrollments between K-12 and ages 5-17. Please also give the full street address of the schools.
Evidence:
Difference in enrollment = `Enrollment (K-12)` - `Enrollment (Ages 5-17)`
Ground Truth:
```
SELECT T1.School, T1.StreetAbr FROM
schools AS T1 INNER JOIN frpm AS T2 ON
T1.CDSCode = T2.CDSCode WHERE
T2.`Enrollment (K-12)` -
T2.`Enrollment (Ages 5-17)` > 30
```
ChatGPT SQL:
```
SELECT s.School, s.Street, s.City,
s.Zip FROM schools s JOIN frpm f ON
s.CDSCode = f.CDSCode WHERE
f.`Enrollment (K-12)` - f.`Enrollment
(Ages 5-17)` > 30
```

Question:
List out the accounts who have the earliest trading date in 1995.
Evidence:
False
Ground Truth:
```
SELECT account_id FROM trans WHERE
STRFTIME('%Y', date) = '1995' ORDER
BY date ASC LIMIT 1
```
ChatGPT SQL:
```
SELECT account_id FROM account WHERE
date = (SELECT MIN(date) FROM
account WHERE date LIKE '1995%')
```

Misunderstanding Knowledge Evidence (17.6%)

Question:
On average, how much did the Student_Club spend on food for the typical event in the past?
Evidence:
DIVIDE(SUM(spent), COUNT(spent)) where category = 'Food'; 'event in the past' means event_status = 'Closed'
Ground Truth:
```
SELECT SUM(spent) / COUNT(spent) FROM
budget WHERE category = 'Food' AND
event_status = 'Closed'
```
ChatGPT SQL:
```
SELECT DIVIDE(SUM(spent), COUNT(spent))
FROM budget WHERE category = 'Food' AND
event_status = 'Closed'
```

Question :
Name elementary schools in Riverside which the average of average math score for SAT is grater than 400, what is the funding type of these schools?
Evidence:
Average of average math = sum(average math scores) / count(schools)
Ground Truth:
```
SELECT T1.sname, T2.`Charter Funding
Type` FROM satscores AS T1 INNER JOIN
frpm ⋯ HAVING CAST(SUM(T1.AvgScrMath)
AS REAL) / COUNT(T1.cds) > 400
```
ChatGPT SQL:
```
SELECT s.School, s.FundingType FROM
schools s ⋯ AND sc.rtype = 'E' AND
sc.AvgScrMath > 400 GROUP BY s.School,
s.FundingType
```

Misunderstanding Database Content (40.8%)

Question:
What is the average math score of the school with the lowest average score for all subjects, and in which county is it located?
Evidence:
Average score for all subjects can be computed by AvgScrMath + AvgScrRead + AvgScrWrite
Ground Truth:
```
SELECT T1.AvgScrMath, T2.County FROM
satscores AS T1 INNER JOIN schools AS
T2 ON T1.cds = T2.CDSCode WHERE
T1.AvgScrMath IS NOT NULL ORDER BY
T1.AvgScrMath + T1.AvgScrRead +
T1.AvgScrWrite ASC LIMIT 1
```
ChatGPT SQL:
```
SELECT cname, AvgScrMath FROM satscores
JOIN schools ON satscores.cds =
schools.CDSCode WHERE satscores.rtype =
'S' ORDER BY (AvgScrMath + AvgScrRead +
AvgScrWrite) ASC LIMIT 1
```

Question:
What is the average lap time in milliseconds of all the lap records set on the various circuits in Italy?
Evidence:
average = AVG(milliseconds)
Ground Truth:
```
SELECT CAST(SUM(T1.milliseconds) AS
REAL) / COUNT(T1.lap) FROM lapTimes AS
T1 INNER JOIN races AS T2 on T1.raceId
= T2.raceId INNER JOIN circuits AS T3
on T2.circuitId = T3.circuitId WHERE
T3.country = 'Italy'
```
ChatGPT SQL:
```
SELECT AVG(milliseconds) FROM
lap_records WHERE circuit IN ('Monza',
'Imola', 'Mugello', 'Misano')
```

Syntax Error (3.0%)

Question:
How many meeting events were held in 2020?
Evidence:
meeting events refers to type = 'Meeting'; held in 2020 refers to YEAR(event_date) = 2020
Ground Truth:
```
SELECT COUNT(event_id) FROM event
WHERE type = 'Meeting' AND
STRFTIME('%Y', COLUMN) = '2020'
```
ChatGPT SQL:
```
SELECT COUNT(*) FROM event WHERE type
= 'Meeting' AND YEAR(event_date) =
2020
```

Question:
How much more was customer 7 consuming in April 2013 than customer 5?
Evidence:
April 2013 refers to '201304' in the yearmonth.date
Ground Truth:
```
SELECT SUM(IIF(CustomerID = 7,
Consumption, 0)) - SUM(IIF(CustomerID =
5, Consumption, 0)) FROM yearmonth
WHERE Date = '201304'
```
ChatGPT SQL:
```
SELECT7, (SELECT Consumption FROM
yearmonth WHERE CustomerID = 7 AND Date
= 201304) - (SELECT Consumption FROM
yearmonth WHERE CustomerID = 5 AND Date
= 201304)
```

Figure 8: 4 major types of error cases are presented. Some cases are shortcut for better presentation.

data is the final training data and dev data for public use, and the remaining 2 batches of data is used for testing. We consider the annotation of first 8 batches of data as a learning processes for SQL annotators since their erroneous SQLs could be fixed by experts and learn how to generate good-quality SQLs for this task. Then their first scores on an examination, conducted by testing set from the final two batches can be viewed as the human performance since we don't interrupt and assist them during examination and all errors are preserved. After testing, we proceed with the following double-blind SQL annotation procedures as Section 3.3 to correct SQLs for these data by discussion with experts. And SQLs after second round double-blind annotation are collected as ground truth.

6.5 Error Analysis

ChatGPT (`GPT-3.5-turbo`) is widely acknowledged as one of the most potent code-based large language models in existence and can achieve the SOTA result in BIRD. Therefore, in this error analysis as shown in the Figure 8, the performance of ChatGPT is concentrated. We observe 500 randomly sampled error cases, providing a detailed assessment in the following categories. **Wrong Schema Linking (41.6%)** pertains to the scenario where ChatGPT can accurately comprehend the structure of the database, but erroneously associates it with inappropriate columns and tables. This demonstrates that the task of schema linking [44, 59], even in intricate and practical situations, continues to be a significant obstacle for text-to-SQL models. **Misunderstanding Database Content (40.8%)** occurs when ChatGPT either fails to recall the correct database structure (e.g., `rtype` doesn't belong to the `satscores` table) or generates fake schema items (e.g., `lap_records` is not appearing in the `formula_1` database and many values are predicted incorrectly) especially when the database is very large. In this case, how to make ChatGPT really understand database structure and contents [28] is still a pain point topic in LLMs. **Misunderstanding Knowledge Evidence (17.6%)** refers to cases in which ChatGPT does not accurately interpret human-annotated evidence. An instance is that ChatGPT directly copies the formula `DIVIDE(SUM(spent), COUNT(spent))`. This finding demonstrates that ChatGPT exhibits a lack of robustness in response to unfamiliar prompts or knowledge, causing it to directly replicate formulas without considering SQL syntax. Additionally, this vulnerability may lead to security concerns. For example, if a poison knowledge evidence is introduced through an unfamiliar formula on purpose, ChatGPT may inadvertently copy it into the output without verification, thus exposing the system to potential data security risks [15]. Instead, humans can understand it and generalize it to SQL via their knowledge and experiences. The final groups of error cases are **Syntax Error (3.0%)** with a small portion, suggesting that the ChatGPT is well-performed zero-shot semantic parser. However, we observe that ChatGPT occasionally employs incorrect keywords (e.g., misusing the MySQL `Year()` function instead of an SQLite function `STRFTIME()` or exhibits decoding errors (e.g., `SELECT7`)).

7 Related Work

Text-to-SQL Dataset A high-quality dataset is essential for driving the development of various natural language processing tasks, including text-to-SQL. The earlier single-domain text-to-SQL datasets such as GeoQuery [57], ATIS [9], and Restaurant [20] were designed for specific information retrieval tasks. The WikiSQL [60] and Spider [55], comprising of more than 10,000 data, are more appropriate for the development of models that prioritize SQL and database knowledge as opposed to the only memorization of domain knowledge or values.

Despite these improvements, the majority of cross-domain text-to-SQL datasets continue to focus on the database schema as opposed to the database values, thereby resulting in a large gap from the real-world scenarios. KaggleDBQA [25] attempted to mitigate this issue by constructing 272 text-to-SQL pairs from eight databases on the open-source data analysis platform Kaggle. Other recent datasets, such as EHRSQL [26], SEDE [13], and MIMICSQL [47], collected databases with diverse and large values. Additionally, the SQL queries in these datasets have become more professional, with features like window functions and time-aware grounding. However, these datasets still focus on a single domain, such as healthcare or software platforms.

Another group of recent work starts to concentrate on creating knowledge-intensive text-to-SQL benchmarks [10, 59]. These datasets assist experts in making informed decisions via knowledge grounding or commonsense reasoning in real-world analysis scenarios. To our best knowledge, the BIRD represents the first large-scale benchmark to incorporate all of the aforementioned real-world features, with a particular emphasis on prioritizing the database values.

Text-to-SQL Models The fundamental principle of a cross-domain text-to-SQL parser involves the construction of an encoder to learn representations of questions and schemas, followed by a decoder to generate SQLs [37]. For example, IRNET [12] designs an encoder consisting of attention-based Bi-LSTM for learning question and schema representations, and a decoder to predict SQLs based on the encoded intermediate representations. RATSQL [44], SDSQL [17], LGESQL [4], and S^{2}SQL [18], Proton [45] enhance the representation learning of natural language questions and database schema via relational graph neural network. R^{2}SQL [16], SCoRe [56], and Star [3] enhance contextual learning for conversational text-to-SQL tasks.

Later, sequence-to-sequence pre-trained language models (PLMs) such as T5 [38] become popular in text-to-SQL tasks due to their portability and capability of generation across different datasets. These models achieve impressive results by fine-tuning with minimal effort. Furthermore, RASAT [36] enhances T5's structural information encoding via schema alignment into the encoder, while Graphix [28] equip T5 with multi-hop reasoning to achieve state-of-the-art results on complicated cross-domain text-to-SQL tasks.

In recent years, LLMs such as ChatGPT [33], Palm [8], OPT [58], have attracted considerable attention due to their powerful zero-shot reasoning and domain generalization capabilities. ChatGPT can perform exceptionally well on semantic parsing tasks, including text-to-SQL tasks, with minimal or no input data. In fact, in the BIRD project, ChatGPT even performs more impressively than initially expected.

SQL Efficiency Efficient execution of SQL queries on big databases has been a significant topic in both academia and industries. Many techniques are proposed to improve SQL query efficiency, by index selection [23], SQL optimization [27, 63], etc.

SQL optimiztion is a common method for enhancing the efficacy of SQL queries. Several SQL optimization algorithms [29, 31, 48], such as rule-based optimization and cost-based optimization, are proved effective. Rule-based optimization employs a set of principles to transform the SQL query into a form that can be executed more efficiently. On the other hand, cost-based optimization estimates the execution cost of various query plans and selects the one with the lowest cost through analyzing statistic distribution of database contents. Similar to NLP community, there are also recent works utilizing artificial intelligence for query optimization such as [63].

Index prediction is another important technique for improving SQL execution efficiency. Researchers propose many algorithms of index prediction [62] based on various optimization criteria, such as minimizing SQL execution time, maximizing index utilization.

However, the evaluation and improvement of SQL queries embedded in the text-to-SQL tasks is more challenging since the accuracy of question & answering (QA) should also be taken into consideration. In this work, we provide VES to measure efficiency of text-to-SQL generators to mitigate this problem. In addition, we suggest that exploring the trade-off between efficiency and execution accuracy could be a promising avenue for future research, as it is beneficial to real-world analysis scenarios.

8 Conclusion

In this paper, we introduce BIRD, an English large-scale cross-domain, text-to-SQL benchmark with a particular focus on large database contents. BIRD mitigates the gap between text-to-SQL research and real-world applications by exploring three additional challenges: 1) handling large and dirty database values, 2) external knowledge reasoning, and 3) optimizing SQL execution efficiency. Our experimental results demonstrate that BIRD presents a more daunting challenge compared to existing benchmarks since even the most powerful codeLM, ChatGPT, falls significantly short of human performance. This leaves plenty of room for improvement and innovation in the text-to-SQL tasks. Moreover, our thorough efficiency and error analyses provide valuable insights and directions for future research, paving the way for the development of more advanced and practical text-to-SQL solutions in real-world scenarios.

References

[1] Ben Bogin, Jonathan Berant, and Matt Gardner. Representing schema structure with graph neural networks for text-to-SQL parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4560–4565, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1448. URL https://aclanthology.org/P19-1448.

[2] Ruichu Cai, Boyan Xu, Zhenjie Zhang, Xiaoyan Yang, Zijian Li, and Zhihao Liang. An encoder-decoder framework translating natural language to database queries. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*, page 3977–3983, 2018.

[3] Zefeng Cai, Xiangyu Li, Binyuan Hui, Min Yang, Bowen Li, Binhua Li, Zheng Cao, Weijie Li, Fei Huang, Luo Si, and Yongbin Li. STAR: SQL guided pre-training for context-dependent text-to-SQL parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1235–1247, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.findings-emnlp.89.

[4] Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. LGESQL: Line graph enhanced text-to-SQL model with mixed local and non-local relations. 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 2541–2555, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.198. URL https://aclanthology.org/2021.acl-long.198.

[5] Shuaichen Chang, Jun Wang, Mingwen Dong, Lin Pan, Henghui Zhu, Alexander Hanbo Li, Wuwei Lan, Sheng Zhang, Jiarong Jiang, Joseph Lilien, Steve Ash, William Yang Wang, Zhiguo Wang, Vittorio Castelli, Patrick Ng, and Bing Xiang. Dr.spider: A diagnostic evaluation benchmark towards text-to-SQL robustness. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Wc5bmZZU9cy.

[6] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor

Babuschkin, S. Arun Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021.

[7] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.300. URL https://aclanthology.org/2021.emnlp-main.300.

[8] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311, 2022.

[9] Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

[10] Longxu Dou, Yan Gao, Xuqi Liu, Mingyang Pan, Dingzirui Wang, Wanxiang Che, Dechen Zhan, Min-Yen Kan, and Jian-Guang Lou. Towards knowledge-intensive text-to-SQL semantic parsing with formulaic knowledge. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5240–5253, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.350.

[11] Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. Towards robustness of text-to-sql models against synonym substitution. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2505–2515. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.195. URL https://doi.org/10.18653/v1/2021.acl-long.195.

[12] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1444. URL https://aclanthology.org/P19-1444.

[13] Moshe Hazoom, Vibhor Malik, and Ben Bogin. Text-to-SQL in the wild: A naturally-occurring dataset based on stack exchange data. In *Proceedings of the 1st Workshop on Natural Language Processing for Programming (NLP4Prog 2021)*, pages 77–87, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.nlp4prog-1.9. URL https://aclanthology.org/2021.nlp4prog-1.9.

[14] Joseph M. Hellerstein. Quantitative data cleaning for large databases. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, page 1197–1200, 2008.

[15] Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. Are large pre-trained language models leaking your personal information? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2038–2047, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.findings-emnlp.148.

[16] Binyuan Hui, Ruiying Geng, Qiyu Ren, Binhua Li, Yongbin Li, Jian Sun, Fei Huang, Luo Si, Pengfei Zhu, and Xiaodan Zhu. Dynamic hybrid relation exploration network for cross-domain context-dependent semantic parsing. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 13116–13124. AAAI Press, 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17550.

[17] Binyuan Hui, Xiang Shi, Ruiying Geng, Binhua Li, Yongbin Li, Jian Sun, and Xiaodan Zhu. Improving text-to-sql with schema dependency learning. In *arXiv:2103.04399*, 2021.

[18] Binyuan Hui, Ruiying Geng, Lihan Wang, Bowen Qin, Yanyang Li, Bowen Li, Jian Sun, and Yongbin Li. S^{2}SQL: Injecting syntax to question-schema interaction graph encoder for text-to-SQL parsers. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1254–1262, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.99. URL https://aclanthology.org/2022.findings-acl.99.

[19] Ihab F. Ilyas and Xu Chu. Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends in Databases*, 5:281–393, 2015.

[20] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1089. URL https://www.aclweb.org/anthology/P17-1089.

[21] Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. QASC: A dataset for question answering via sentence composition. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8082–8090. AAAI Press, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/6319.

[22] Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213, 2022.

[23] Jan Kossmann, Stefan Halfpap, Marcel Jankrift, and Rainer Schlosser. Magic mirror in my hand, which is the best in the land? an experimental evaluation of index selection algorithms. *Proceedings of the VLDB Endowment*, 13(12):2382–2395, 2020.

[24] Prerna S. Kulkarni and Jagdish W. Bakal. Survey on data cleaning. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, page 2361–2366, 2014.

[25] Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 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 (Volume 1: Long Papers)*, pages 2261–2273, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.176. URL https://aclanthology.org/2021.acl-long.176.

[26] Gyubok Lee, Hyeonji Hwang, Seongsu Bae, Yeonsu Kwon, Woncheol Shin, Seongjun Yang, Minjoon Seo, Jong-Yeup Kim, and Edward Choi. Ehrsql: A practical text-to-sql benchmark for electronic health records. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 15589–15601. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/

643e347250cf9289e5a2a6c1ed5ee42e-Paper-Datasets_and_Benchmarks.pdf.

[27] Dandan Li, Lu Han, and Yi Ding. Sql query optimization methods of relational database system. In *2010 Second International Conference on Computer Engineering and Applications*, volume 1, pages 557–560. IEEE, 2010.

[28] Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, and Yongbin Li. Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. *ArXiv*, abs/2301.07507, 2023.

[29] Tanzim Mahmud, KM Azharul Hasan, Mahtab Ahmed, and Thwoi Hla Ching Chak. A rule based approach for nlp based query processing. In *2015 2nd International Conference on Electrical Information and Communication Technologies (EICT)*, pages 78–82. IEEE, 2015.

[30] Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. Augmented language models: a survey. *ArXiv*, abs/2302.07842, 2023.

[31] Vamsi Krishna Myalapalli and ASN Chakravarthy. Revamping sql queries for cost based optimization. In *2016 International Conference on Circuits, Controls, Communications and Computing (I4C)*, pages 1–6. IEEE, 2016.

[32] Paulo H. Oliveira, Daniel dos Santos Kaster, Caetano Traina, and Ihab F. Ilyas. Batchwise probabilistic incremental data cleaning. *ArXiv*, abs/2011.04730, 2020.

[33] Long Ouyang, Jeff 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 E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155, 2022.

[34] Hamid Pirahesh, Joseph M. Hellerstein, and Waqar Hasan. Extensible/rule based query rewrite optimization in starburst. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 39–48, 1992.

[35] Mohammadreza Pourreza and Davood Rafiei. DIN-SQL: decomposed in-context learning of text-to-sql with self-correction. *CoRR*, abs/2304.11015, 2023. doi: 10.48550/arXiv.2304.11015. URL https://doi.org/10.48550/arXiv.2304.11015.

[36] Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Yu Cheng, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 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*, pages 3215–3229, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.211.

[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. A survey on text-to-sql parsing: Concepts, methods, and future directions. In *arXiv:2208.13629*, 2022.

[38] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[39] Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. Evaluating the text-to-sql capabilities of large language models. *ArXiv*, abs/2204.00498, 2022.

[40] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-2124. URL https://aclanthology.org/P18-2124/.

[41] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online

and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.779. URL https://aclanthology.org/2021.emnlp-main.779.

[42] Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? 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 922–938, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.75. URL https://aclanthology.org/2021.acl-long.75.

[43] Alon Talmor, Ori Yoran, Ronan Le Bras, Chandra Bhagavatula, Yoav Goldberg, Yejin Choi, and Jonathan Berant. Commonsenseqa 2.0: Exposing the limits of ai through gamification. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/3ef815416f775098fe977004015c6193-Paper-round1.pdf.

[44] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 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, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.677. URL https://aclanthology.org/2020.acl-main.677.

[45] Lihan Wang, Bowen Qin, Binyuan Hui, Bowen Li, Min Yang, Bailin Wang, Binhua Li, Jian Sun, Fei Huang, Luo Si, and Yongbin Li. Proton: Probing schema linking information from pre-trained language models for text-to-sql parsing. In Aidong Zhang and Huzefa Rangwala, editors, *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 1889–1898. ACM, 2022. doi: 10.1145/3534678.3539305. URL https://doi.org/10.1145/3534678.3539305.

[46] Lijie Wang, Ao Zhang, Kun Wu, Ke Sun, Zhenghua Li, Hua Wu, Min Zhang, and Haifeng Wang. DuSQL: A large-scale and pragmatic Chinese text-to-SQL dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6923–6935, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.562. URL https://aclanthology.org/2020.emnlp-main.562.

[47] Ping Wang, Tian Shi, and Chandan K. Reddy. Text-to-sql generation for question answering on electronic medical records. *Proceedings of The Web Conference 2020*, 2020.

[48] Zhaoguo Wang, Zhou Zhou, Yicun Yang, Haoran Ding, Gansen Hu, Ding Ding, Chuzhe Tang, Haibo Chen, and Jinyang Li. Wetune: Automatic discovery and verification of query rewrite rules. In *Proceedings of the 2022 International Conference on Management of Data*, pages 94–107, 2022.

[49] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022.

[50] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903, 2022.

[51] Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. UnifiedSKG: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 602–631, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.39.

[52] Xiaojun Xu, Chang Liu, and Dawn Song. Sqlnet: Generating structured queries from natural language without reinforcement learning. *ArXiv preprint*, 2017.

[53] Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. Sqlizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA): 1–26, 2017.

[54] Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 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)*, page 588–594, 2018.

[55] 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. 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*, page 3911–3921, 2018.

[56] Tao Yu, Rui Zhang, Alex Polozov, Christopher Meek, and Ahmed Hassan Awadallah. Score: Pre-training for context representation in conversational semantic parsing. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=oyZxhRI2RiE.

[57] John M. Zelle and Raymond J. Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, pages 1050–1055, 1996.

[58] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068, 2022.

[59] Chen Zhao, Yu Su, Adam Pauls, and Emmanouil Antonios Platanios. Bridging the generalization gap in text-to-SQL parsing with schema expansion. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5568–5578, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.381. URL https://aclanthology.org/2022.acl-long.381.

[60] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2SQL: Generating structured queries from natural language using reinforcement learning. In *CoRR abs/1709.00103*, 2017.

[61] Victor Zhong, Mike Lewis, Sida I. Wang, and Luke Zettlemoyer. Grounded adaptation for zero-shot executable semantic parsing. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6869–6882. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.558. URL https://doi.org/10.18653/v1/2020.emnlp-main.558.

[62] Rong Zhou. Research on key performance index prediction of distributed database based on machine learning algorithm. In *Proceedings of the 2nd International Conference on Cognitive Based Information Processing and Applications (CIPA 2022) Volume 2*, pages 563–567. Springer, 2023.

[63] Xuanhe Zhou, Guoliang Li, Chengliang Chai, and Jianhua Feng. A learned query rewrite system using monte carlo tree search. *Proceedings of the VLDB Endowment*, 15(1):46–58, 2021. doi: 10.14778/3485450.3485456.

[64] Xuanhe Zhou, Chengliang Chai, Guoliang Li, and Ji Sun. Database meets artificial intelligence: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(3):1096–1116, 2022. doi: 10.1109/TKDE.2020.2994641.