# Archer: A Human-Labeled Text-to-SQL Dataset with Arithmetic, Commonsense and Hypothetical Reasoning

**Danna Zheng**[1], **Mirella Lapata**[1], **Jeff Z. Pan**[1,2]
[1] School of Informatics, University of Edinburgh, UK
[2] Huawei Edinburgh Research Centre, CSI, UK
dzheng@ed.ac.uk, mlap@inf.ed.ac.uk, http://knowledge-representation.org/j.z.pan/

## Abstract

We present Archer, a challenging bilingual text-to-SQL dataset specific to complex reasoning, including arithmetic, commonsense and hypothetical reasoning. It contains 1,042 English questions and 1,042 Chinese questions, along with 521 unique SQL queries, covering 20 English databases across 20 domains. Notably, this dataset demonstrates a significantly higher level of complexity compared to existing publicly available datasets. Our evaluation shows that Archer challenges the capabilities of current state-of-the-art models, with a high-ranked model on the Spider leaderboard achieving only 6.73% execution accuracy on Archer test set. Thus, Archer presents a significant challenge for future research in this field.

## 1 Introduction

The text-to-SQL task is an important NLP task, which maps input questions to meaningful and executable SQL queries, enabling users to interact with databases in a more intuitive and user-friendly manner. State-of-the-art methods (Pourreza and Rafiei, 2024; Li et al., 2023a,b; Scholak et al., 2021) relying on large language models have achieved execution accuracy above 75% on the Spider dataset(Yu et al., 2018), which encompasses complex SQL grammar and cross-domain settings. Recently, Pourreza and Rafiei (2024) achieved remarkable results with an impressive 85.3% execution accuracy on the Spider dataset, leveraging the enhanced capabilities of GPT-4.

However, previous text-to-SQL datasets (Yu et al., 2018; Finegan-Dollak et al., 2018; Yaghmazadeh et al., 2017; Iyer et al., 2017; Zhong et al., 2017; Li and Jagadish, 2014; Giordani and Moschitti, 2012; Popescu et al., 2003; Tang and Mooney, 2000; Dahl et al., 1994), have limitations that prevent them from capturing complex reasoning effectively. For example, Spider (Yu et al., 2018) purposely excludes questions that would require external knowledge (Pan et al., 2023, 2017a,b), like that from common-sense knowledge graphs or mathematical calculations. This exclusion limits Spider's ability to properly test how well models can handle real-world scenarios, which often require a deeper level of reasoning capabilities.

In this paper, we present Archer, an innovative dataset designed to incorporate three distinct types of reasoning: arithmetic, commonsense, and hypothetical reasoning. By including such varied reasoning skills, Archer seeks to challenge and expand the capabilities of text-to-SQL models, equipping

***Arithmetic Reasoning***

How much higher is the maximum power of a BMW car than the maximum power of a Fiat car?
宝马汽车的最高功率比飞雅特汽车的最高功率高多少?

```
SELECT MAX(horsepower) - (SELECT MAX (horsepower)
FROM cars_data A JOIN car_names B ON A.id=B.makeid
WHERE B.model="fiat") AS diff FROM cars_data A JOIN
car_names B ON A.id=B.makeid WHERE B.model="bmw"
```

***Commonsense Reasoning***

Which 4-cylinder car needs the most fuel to drive 300 miles? List how many gallons it needs, and its make and model.
开300英里耗油最多的四缸车的品牌和型号分别是什么，它需要多少加仑的油?

**Commonsense Knowledge:** Fuel used is calculated by divding distance driven by fuel consumption.

```
SELECT B. Make, B.Model, 1.0 * 300 / mpg AS n_gallon
FROM cars_data A JOIN car_names B ON A.Id=B.MakeId
WHERE cylinders="4" ORDER BY mpg ASC LIMIT 1
```

***Hypothetical Reasoning***

If all cars produced by the Daimler Benz company have 4-cylinders, then in all 4-cylinder cars, which one needs the most fuel to drive 300 miles? Please list how many gallons it needs, along with its make and model.
假如生产自奔驰公司的车都是四缸，开300英里耗油最多的四缸车的品牌和型号分别是什么，它需要多少加仑的油?

```
SELECT B.Make, B.Model, 1.0 * 300 / mpg AS n_gallon
FROM cars_data A JOIN car_names B ON A.id=B.makeid
JOIN model_list C ON B.model=C.model JOIN car_makers
D on C.maker=D.id WHERE D.fullname="Daimler Benz" or
A.cylinders="4" ORDER BY mpg ASC LIMIT 1
```

Figure 1: Archer examples with three reasoning types: arithmetic, commonsense, and hypothetical reasoning. (See more examples in Appendix D)



*Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics*
*Volume 1: Long Papers*, pages 94–111

them to manage more intricate and nuanced queries. Figure 1 showcases data examples from Archer that demonstrate these three reasoning abilities.

To evaluate the challenge posed by Archer, we conducted experiments with both large language models (LLMs) and fine-tuned models. However, all models demonstrated inferior performance when dealing with Archer. Even the model that achieved a high place on the Spider leaderboard managed only 6.73% execution accuracy on Archer test sets. These findings highlight substantial potential for improvement, indicating that Archer indeed provides a significant challenge to current models.

## 2 Reasoning Types

In this section, we present the three different types of reasoning in Archer: arithmetic, commonsense, and hypothetical reasoning.

**Arithmetic reasoning** Arithmetic reasoning pertains to the act of resolving mathematical problems through logical and analytical thought processes, involving datatype values (Pan and Horrocks, 2003, 2005) and arithmetic operators. According to an analysis of SQL queries from practical applications like the Baidu search engine and customer service and data analysis robots by Wang et al. (2020), mathematical calculations account for a significant portion across SQL applications. However, previous high-quality datasets contain very few questions that involve calculations, and such questions are typically auto-generated with simple grammar. In contrast, all question-SQL pairs included in Archer necessitate arithmetic reasoning and are manually annotated to ensure high quality.

**Commonsense reasoning** Commonsense reasoning refers to the capacity to make logical deductions based on implicit (and possibly uncertain) commonsense knowledge (Romero et al., 2019; Arnaout et al., 2022; He et al., 2023; Wan et al., 2021; Wang et al., 2010, 2014; Stoilos et al., 2006; Pan et al., 2005), including, e.g., a broad understanding of how things function in the world. Commonsense knowledge can be useful for both zero-shot learning (Chen et al., 2023a, 2021a,b, 2023b; Geng et al., 2023) and model explanations (Guan et al., 2024; Chen et al., 2018). Archer includes questions that necessitate models to comprehend the database, infer missing details, and generate logical inferences to create accurate SQL queries. As illustrated in Figure 1, for the question *"Which 4-cylinder car needs the most fuel to drive 300 miles? List how many gallons it needs, and its make and model."*, the database does not provide an explicit schema about the fuel used to travel 300 miles for each car. It only provides each car's fuel consumption in MPG. Solving this question requires commonsense knowledge, specifically the understanding of *"Fuel used is calculated by dividing distance driven by fuel consumption"* to derive the correct SQL.

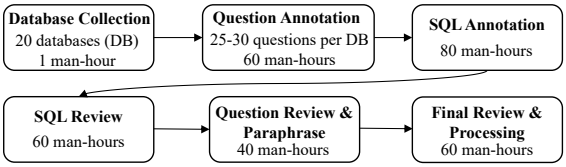


Figure 2: The annotation process of our Archer.

**Hypothetical reasoning** Hypothetical reasoning takes the complexity a step further, requiring models to have counterfactual thinking ability, which is the ability to imagine and reason over unseen cases based on the seen facts and counterfactual assumptions. Archer includes questions that involve hypothetical situations, requiring the model to understand and reason about conditional relationships. As illustrated in Figure 1, consider the hypothetical question *"If all cars produced by the Daimler Benz company have 4-cylinders, then in all 4-cylinder cars, which one needs the most fuel to drive 300 miles? Please list how many gallons it needs, along with its make and model."*. In this question, the underlying assumption contradicts the factual information stored in the database. The model must comprehend this assumption and convert it into the SQL condition *d.fullname = "Daimler Benz" or a.cylinders = "4"*.

## 3 Corpus Construction

As illustrated in Figure 2, we create Archer in the following six steps, spending around 300 hours of human labor in total: §3.1 Database Collection, §3.2 Question Annotation, §3.3 SQL Annotation, §3.4 SQL review, §3.5 Question Review and Paraphrase, §3.6 Final Review and Process.

### 3.1 Database Collection

In a noteworthy research study conducted by Yu et al. (2018), a total of 200 high-quality databases across various domains were meticulously collected and created, requiring approximately 150 man-hours. Out of these, 166 databases were made

publicly available.

Since not all Spider databases support the proposed reasoning types, we carefully selected 20 databases across 20 domains from the Spider 166 databases based two criteria. Firstly, we applied a script to keep only databases with a minimum of 3 tables and 20 columns within each database, as well as a minimum of 6 columns with time or numeric data types. Secondly, we manually checked the filtered databases. These two steps ensure that each selected database contains sufficient information to support complex reasoning.

### 3.2 Question Annotation

Two bilingual (English and Chinese) Ph.D. students with SQL experience were assigned the task of generating questions based on 20 databases. The annotators were required to propose 25-30 questions for each database, ensuring that the questions met the following four requirements:

**1) Arithmetic Reasoning:** Each question should incorporate arithmetic reasoning. The annotators were expected to include a minimum of five questions for each arithmetic reasoning type (addition, subtraction, multiplication, division).

**2) Hypothetical Reasoning:** At least five questions should involve hypothetical reasoning. For each question using hypothetical reasoning, the annotators were also required to propose a corresponding factual question.

**3) Commonsense Reasoning:** The annotators were encouraged to propose questions that involve commonsense reasoning. However, the number of questions with commonsense reasoning was not strictly limited. This flexibility acknowledged that not all databases support commonsense reasoning, and not all arithmetic calculations necessitate it.

**4) Complex SQL Grammar:** The annotators were encouraged to propose questions that require the utilization of complex SQL grammar, such as GROUP BY, ORDER BY, and JOIN.

The annotators were asked to write each question in both English and Chinese. Besides, they were instructed to indicate the reasoning types involved (arithmetic: addition, subtraction, multiplication, division; hypothetical; commonsense), and provide the relevant knowledge or formulation if the question incorporated commonsense reasoning.

### 3.3 SQL Annotation

In order to mitigate cognitive bias, we employed a diverse set of annotators for the tasks of generating questions and writing SQL queries. Two Ph.D. students, who possess strong SQL skills, were specifically chosen to translate the natural language questions into SQL queries. Their responsibilities encompassed the following:

**1) Clarity Ensuring:** The annotators reviewed both English and Chinese questions to identify any ambiguity and restructure them accordingly.

**2) SQL Writing:** The annotators were instructed to use consistent SQL patterns when multiple equivalent queries are applicable for similar questions.

**3) Verification and Correction:** The annotators were also responsible for reviewing the annotations pertaining to reasoning types and the common knowledge necessary to solve each question.

### 3.4 SQL Review

To ensure the correctness of the annotated SQL for each question, we employed a professional SQL expert to review all the SQL queries and rectify any incorrect ones. Subsequently, the original SQL annotators were responsible for verifying the SQL queries corrected by the expert. In cases where there are differences of opinion between the expert and the annotators regarding the corrected queries, they were required to engage in a discussion and reach a consensus to finalize the SQL annotation.

### 3.5 Question Review and Paraphrase

We employed two native English speakers and two native Chinese speakers to review and paraphrase English and Chinese questions, respectively. Initially, their task was to assess the naturalness and grammatical accuracy of the questions. Subsequently, the annotators were requested to provide a paraphrased version of each question in order to enhance the dataset's robustness.

### 3.6 Final Review and Processing

In the final stage of our process, we assigned the task of reviewing the English and Chinese questions, SQL, and annotations relating to reasoning types and commonsense knowledge to our most seasoned annotator. Once this comprehensive review was completed, we ran a script to ensure that all SQL queries are executable.

## 4 Dataset Statistics and Comparison

In Table 1, we present a summary of the statistics for Archer as well as other publicly available text-to-SQL datasets. We conducted a comparative analysis of Archer and other datasets based on four key perspectives: scale, complexity, reasoning distribution, and language.

### 4.1 Scale

Archer consists of 1,042 Chinese questions, 1,042 English questions, and 521 corresponding SQL queries, covering a wide range of 20 distinct databases spanning 20 domains. Each database in Archer, on average, consists of 7.55 tables and 45.25 columns. Archer stands out for its inclusion of multiple domains and a higher average number of tables and columns.

It is worth noting that WikiSQL (Zhong et al., 2017) and DuSQL (Wang et al., 2020) are exceptionally large databases generated automatically. Inspired by them, Archer has the potential to serve as a valuable resource for summarizing SQL templates and training SQL-to-text generators to create large-scale datasets in line with our reasoning setting. In this project, we do not utilize Archer for automatic question-SQL pairs generation. This possibility is a potential future direction.

### 4.2 Complexity

Archer distinguishes itself by its considerably higher level of complexity compared to existing text-to-SQL datasets. Several factors contribute to this complexity:

Firstly, the average question length in Archer is significantly longer than that in other datasets. This poses a challenge to models because longer inputs increase the likelihood of misunderstandings or misinterpretations of specific question details.

Secondly, the average SQL length in Archer stands at 79.71, which is significantly longer than that of other datasets except for ATIS, which contains only one table. Longer SQL statements increase the likelihood of generating incorrect code.

Thirdly, value prediction, which is crucial in SQL generation, is often undervalued in current research. Interestingly, Pourreza and Rafiei (2024) achieved an execution accuracy of 85.3% on the Spider dataset without utilizing database content. This is primarily because Spider SQL queries typically contain an average of only 0.93 value slots, with most values explicitly quoted in the question. In contrast, Archer emphasizes the importance of values, with an average of 6.21 value slots per SQL. Furthermore, Archer questions do not explicitly quote exact values; instead, they naturally mention value information, mirroring real-world scenarios.

Fourthly, SQL queries in Archer refer to an average of 2.17 tables, suggesting that a substantial number of the questions require the use of information from multiple tables to derive SQLs.

Fifthly, the level of SQL statement nesting in Archer is higher than that in other datasets, indicating a greater degree of reasoning complexity required to answer Archer questions, which often necessitates the use of multiple subqueries.

Finally, Archer exhibits a high usage rate of complex SQL grammar features such as `GROUP BY` and `ORDER BY` in each SQL, surpassing the frequency of usage seen in nearly all other datasets.

### 4.3 Reasoning Distribution

All questions in Archer require arithmetic reasoning. This means that mathematical calculations and operations are essential in understanding and answering these questions effectively. Additionally, 44.0% of the questions involve hypothetical reasoning, where the model needs to reason about hypothetical scenarios to derive the correct SQL. Furthermore, 51.4% of the questions require commonsense reasoning, where the model needs to utilize general knowledge and commonsense understanding to produce the correct SQL.

It is worth noting that the majority of previous text-to-SQL datasets do not incorporate arithmetic and commonsense reasoning. Moreover, none of the previous datasets contain questions that involve hypothetical reasoning. Therefore, the inclusion of these types of reasoning tasks in Archer sets it apart from previous datasets and presents new challenges for models in the field of text-to-SQL understanding and generation.

### 4.4 Language

Unlike most previous text-to-SQL datasets that focus solely on English, Archer provides both English and Chinese questions. This bilingual feature of Archer enhances the evaluation and training capabilities of text-to-SQL models, catering to the needs of users in both English and Chinese languages, while forming a solid base for potential support of more languages for Archer, which is left as a future work.

| Dataset | Scale | | | | | | Complexity | | | | | | | Reasoning Distribution | | | | | | | Lang |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #Q | #SQL | #DB | #Dom | T/DB | C/DB | QL | SQLL | VS | TM | NL | GB | OB | A(+) | A(-) | A(*) | A(/) | H | C | C+H | |
| ATIS | 5280 | 947 | 1 | 1 | 25 | 131 | 10.53 | **99.75** | 3.14 | **4.66** | 0.39 | 0.01 | 0.00 | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | en |
| GeoQuery | 877 | 246 | 1 | 1 | 8 | 31 | 7.48 | 26.76 | 0.82 | 1.46 | 1.04 | 0.18 | 0.07 | ✗ | ✗ | ✗ | 0.2% | ✗ | ✗ | ✗ | en |
| Scholar | 817 | 193 | 1 | 1 | 12 | 28 | 6.59 | 38.03 | 1.36 | 3.26 | 0.02 | 0.37 | 0.28 | ✗ | 0.5% | ✗ | ✗ | ✗ | ✗ | ✗ | en |
| Academic | 196 | 185 | 1 | 1 | 15 | 42 | 13.33 | 36.85 | 1.30 | 3.23 | 0.04 | 0.21 | 0.12 | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | en |
| IMDB | 131 | 89 | 1 | 1 | 16 | 65 | 10.23 | 29.51 | 1.20 | 2.84 | 0.01 | 0.07 | 0.11 | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | en |
| Yelp | 128 | 120 | 1 | 1 | 7 | 38 | 9.87 | 28.33 | 1.68 | 2.25 | 0.00 | 0.10 | 0.08 | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | en |
| Advising | 4387 | 205 | 1 | 1 | 18 | 124 | 10.90 | 48.08 | 3.06 | 3.13 | 0.17 | 0.03 | 0.07 | 3.4% | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | en |
| Restaurant | 378 | 23 | 1 | 1 | 3 | 12 | 10.13 | 29.57 | 2.26 | 2.26 | 0.17 | 0.00 | 0.00 | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | en |
| WikiSQL | 80654 | 51159 | 26531 | - | 1 | 6.33 | 12.46 | 13.32 | 0.53 | 1.00 | 0.00 | 0.00 | 0.00 | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | en |
| DuSQL | 25003 | 20308 | 208 | - | 4.04 | 21.38 | 19.20 | 20.63 | 1.16 | 1.33 | 0.20 | 0.42 | 0.30 | 2.4% | 9.5% | 1.0% | 4.4% | ✗ | - | ✗ | zh |
| BIRD | 10962 | 10841 | 80 | - | 7.68 | 54.71 | 15.81 | 23.85 | 1.16 | 2.20 | 0.08 | 0.10 | 0.19 | 0.8% | 5.0% | 7.9% | 10.0% | ✗ | - | ✗ | en |
| Cspider | 9693 | 5275 | 166 | 99 | 5.28 | 27.13 | 11.90 | 24.37 | 0.93 | 1.69 | 0.10 | 0.23 | 0.21 | 0.1% | 0.1% | ✗ | 0.0% | ✗ | ✗ | ✗ | zh |
| Spider | 9693 | 5275 | 166 | 99 | 5.28 | 27.13 | 13.29 | 24.37 | 0.93 | 1.69 | 0.10 | 0.23 | 0.21 | 0.1% | 0.1% | ✗ | 0.0% | ✗ | ✗ | ✗ | en |
| KaggleDBQA | 272 | 249 | 8 | 8 | 2.13 | 22.38 | 9.83 | 13.80 | 0.54 | 1.18 | 0.00 | 0.44 | **0.50** | 0.0% | 0.0% | ✗ | 0.0% | ✗ | ✗ | ✗ | en |
| **Archer (Ours)** | 1042 | 521 | 20 | 20 | 7.55 | 45.25 | en-**29.94** zh-**25.99** | 79.71 | **6.21** | 2.17 | **1.08** | **0.59** | 0.26 | 34.0% | 47.8% | 62.0% | 40.7% | 44.0% | 51.4% | 22.1% | en zh |

Table 1: Comparison of public text-to-SQL datasets. The abbreviations used are as follows: #Q for the number of unique questions, #SQL for the number of unique SQLs, #DB for the number of databases, #Dom for the number of domains, T/DB for the number of tables per database, C/DB for the number of columns per database, QL for the average question length, SQLL for the average SQL length, VS for the average number of value slots per question, TM for the average number of tables mentioned in each SQL, NL for the average nested level per SQL, GB and OB for the average number of `GROUP BY` and `ORDER BY` clauses per SQL respectively. A, H, C, and Lang represent arithmetic, hypothetical, commonsense, and language, respectively. The cross mark, - denote absence and presence respectively. The statistics for BIRD, CSpider, and Spider is based on training and dev sets as their test sets are unavailable. Language is represented as en for English databases and questions, zh for Chinese databases and questions, and zh for English databases and Chinese questions.

## 5 Experiments

### 5.1 Baseline Models

We benchmark the performance of two types of presentative text-to-SQL models on Archer: LLMs and finetuned Models.

**LLMs** LLMs have shown strong performance on commonly used text-to-SQL benchmarks, such as Spider. To analyze the difficulty of the whole Archer, we provide the zero-shot results of GPT-3.5 (`gpt-3.5-turbo`) with different prompt settings: *API Doc*, *CT-3*, *CT-3+COT*. *API Doc* follows the style of the Text-to-SQL example provided by OpenAI, which includes the schema information in a comment style. *CT-3*, introduced by Rajkumar et al. (2022), includes the `CREATE TABLE` commands for each table and the results of executing a `SELECT * FROM T LIMIT 3` query on each table. Compared with API Doc, CT-3 provides more information like declarations of column types and foreign keys, and a small amount (3) of content examples. *CT-3+COT* implement the Chain-Of-Thought (COT) technique on top of the CT-3 prompt by appending the prompt sentence `"Let's think step by step."` before the SQL generation. Following the work of Li et al. (2023c), we provide a 1-shot pseudo example for LLMs to learn the procedure of thinking and output format. Furthermore, we evaluate the performance of *GPT-4+DIN-SQL* (Pourreza and Rafiei, 2024) on Archer. As a highly-ranked solution on the Spider leaderboard at the time of writing, it consists of four modules: (1) schema linking, (2) query classification and decomposition, (3) SQL generation, and (4) self-correction. The initial three modules exploit the in-context learning ability of GPT-4 with ten shots, while the self-correction is conducted by GPT-4 in a zero-shot setting. Note that we do not evaluate GPT-4+DIN-SQL on Archer Chinese questions because it is designed for English datasets. More details on the prompts can be found in Appendix A.

**Fine-tuned Models** T5-based fine-tuned models have shown promising results on the Spider leaderboard. It is, however, worth mentioning that many top-tier models on the leaderboard are customized specifically for the limited SQL grammars present in the Spider dataset. Given that our dataset contains more complex grammatical structures compared to Spider, these specialized models may not be suitable for our needs. As a result, we select vanilla T5 models as our baselines instead of the aforementioned variants. We evaluate English questions using T5-base, T5-large, T5-3B, and evaluate Chinese questions using mT5-base, mT5-large, mT5-xl. We concatenate the natural question $Q$ and database schema into a sequence as input in a format as below:

$$x = [q_1, ..., q_{|Q|}|t_1 : c_1^{t_1}, ..., c_{|t_1|}^{t_1}|...|t_{|\mathcal{T}|} : c_1^{t_{|\mathcal{T}|}}, ..., c_{|t_{|\mathcal{T}|}|}^{t_{|\mathcal{T}|}}] \quad (1)$$

| Models | EN Questions, EN databases | | | | ZH Questions, EN databases | | | |
|---|---|---|---|---|---|---|---|---|
| | Full | | Test | | Full | | Test | |
| | VA | EX | VA | EX | VA | EX | VA | EX |
| ***LLMs*** | | | | | | | | |
| GPT-3.5 + API Doc | 82.63 | 13.24 | 83.65 | 3.85 | 86.18 | 10.65 | 85.58 | **3.85** |
| GPT-3.5 + CT-3 | **84.17** | **13.34** | 80.77 | 3.85 | **91.17** | **12.86** | **91.35** | 1.92 |
| GPT-3.5 + CT-3 + COT | 75.14 | 13.24 | 74.04 | 4.81 | 72.84 | 12.19 | 65.38 | **3.85** |
| GPT-4 + DIN-SQL | - | - | **96.15** | **6.73** | - | - | - | - |
| ***Fine-tuned Models*** | | | | | | | | |
| T5-base/mT5-base | - | - | 11.54 | 0.00 | - | - | 9.62 | 0.00 |
| T5-large/mT5-large | - | - | 15.38 | 0.00 | - | - | 14.42 | 0.00 |
| T5-3B/mT5-xl | - | - | 19.23 | 0.00 | - | - | 17.31 | 0.00 |
| T5-base/mT5-base + Aug | - | - | 25.00 | 0.00 | - | - | 24.03 | 0.00 |
| T5-large/mT5-large + Aug | - | - | 33.65 | 3.84 | - | - | 30.77 | 0.96 |
| T5-3B/mT5-xl + Aug | - | - | **50.00** | **4.81** | - | - | **61.54** | **1.92** |

Table 2: Baseline performance on Archer. GPT-4+DIN-SQL was tested only on the English set due to cost and its English-specific design. We only report the fine-tuned model's performance on the test set.

where $q_i$ is the $i^{th}$ question token, $t_j$ is the $j^{th}$ table, and $c_k^{t_j}$ is the $k^{th}$ in the $j^{th}$ table. Following the works of Li et al. (2023a); Lin et al. (2020), we extract the potential database cell values and append them to their corresponding columns.

### 5.2 Evaluation Metrics

We employ two evaluation metrics: VAlid SQL (VA) and EXecution accuracy (EX). VA is the proportion of the predicted SQL statements that can be executed successfully, no matter with correct or incorrect answers. EX is the proportion of the predicted SQL statements where the execution results match those of the gold SQL statements. We computed EX of each instance use a new evaluation script as shown in Algorithm 1, which mitigates the false-negative issue in present publicly available evaluation scripts caused by permutations of columns and rows.

### 5.3 Experiments Setup

**Data Split** Among the 20 databases, we split 16, 2, and 2 databases as training, dev, and test sets, respectively. The databases for Archer training set are collected from the Spider training set, and the databases for Archer dev set and test set are collected from the Spider dev set. We strive to introduce as few new SQL keywords as possible during SQL annotation to facilitate the integration of our dataset with the Spider and CSpider datasets. We also report the performance of T5 finetuned on the augmented training set which consists of Archer training set and Spider/CSpider training set.

For LLM baselines, we assess the zero-shot performance of GPT-3.5 on the full Archer to evaluate the dataset's overall difficulty. As for GPT-4+DINSQL, due to its high cost and extended response time, we only test it on Archer test sets.

**Hyper-Parameters** For GPT-3.5 baselines, we set stop sequence to `['--', ';', '#']` and the temperature to 0. In the case of GPT-4+DIN-SQL, we adhere to the default setting as outlined in Pourreza and Rafiei (2024). For T5 baselines, we employ the Adafactor optimizer with a learning rate of 5e-5. For T5-base/mT5-base and T5-large/mT5-large, we adopt a batch size of 6 and a gradient descent step of 5. For T5-3b and mT5-xl, we use a batch size of 2 and a gradient descent step of 16. To adjust the learning rate, we utilize linear warm-up with a warm-up rate of 0.1, followed by cosine decay. During inference, we set the beam size to 8. We set the maximum epoch to 128, having checkpoints every 10 epochs as well as the last epoch. We then select the optimal checkpoints based on their EX performance on the development set.

## 6 Results and Discussion

### 6.1 Overall Evaluation

We summarize the performance of LLMs and fine-tuned models in Table 2. The low performance of these models on Archer suggests that Archer presents a significant challenge. This underscores the considerable potential for future improvement in this domain.

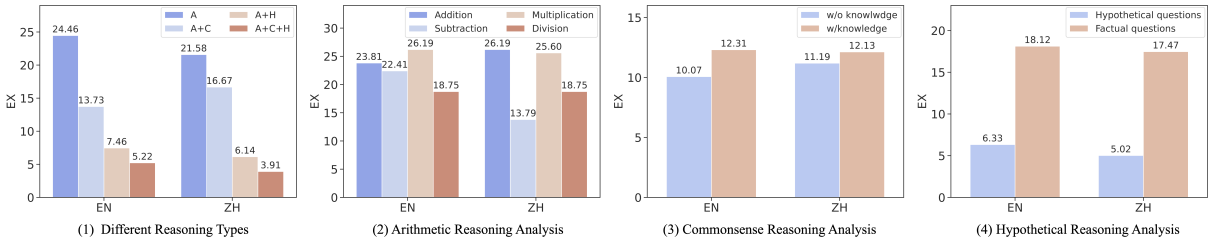


Figure 3: GPT-3.5 + CT-3 execution accuracy comparison across and within different reasoning types. A refers to arithmetic. H refers to hypothetic. C refers to commonsense.

**LLM** GPT-4+DIN-SQL obtain EX score of 6.73% on Archer test set, while it is able to achieve 85.3% test-suite execution performance on Spider test set (Pourreza and Rafiei, 2024). To evaluate the overall difficulty of Archer, we test the zero-shot performance of GPT-3.5 with API Doc, CT-3, CT-3+COT prompts on the full Archer data. Among the three kinds of prompts, CT-3 achieves the highest EX scores on both English data (EX: 13.34%) and Chinese data (EX: 12.86%). As expected, CT-3 performed slightly better than API Doc, likely due to its inclusion of more useful information, such as declarations of column types and foreign keys. However, the addition of COT in CT-3+COT did not outperform CT-3 on the complete Archer. On the other hand, for the Test set only, CT-3+COT slightly outperform CT-3. From Table 2, we observe a significant decrease in VA when using COT, suggesting that COT suffers from having more syntax errors in the generated SQL. Although CT-3+COT achieved a higher EX score than CT-3 and API Doc specifically for questions involving arithmetic and commonsense reasoning, it performed less effectively on questions that require hypothetical reasoning (cf. Table 3 in Appendix C).

**Finetuned Models** From Table 2. we observe that T5 from scale base to 3B (XL) trained on Archer training set achieve 0.00% EX scores. This outcome could be attributed to the small-scale nature of Archer combined with its high complexity. However, when Archer training set was augmented with the Spider/CSpider training set, the VA scores of T5 models exhibited a substantial improvement. Specifically, the T5-3B model trained on the augmented training set achieved an EX score of 4.81% on the English test (matching the performance of GPT-3.5+CT-3+COT) set and 1.92% on the Chinese test set (matching the performance of GPT-3.5+CT-3).

These results suggest that Archer has the potential to advance the development of text-to-SQL systems with complex reasoning.

### 6.2 Different Reasoning Analysis

To gain a comprehensive understanding of the difficulty levels within the complete Archer across various reasoning types, we conducted a thorough analysis using the GPT-3.5 model with the CT-3 prompt, which demonstrated the highest performance on the full dataset. Additional results for GPT-3.5 with alternative prompts can be found in Appendix C.

**Overall Comparison** Figure 3-(1) shows the performance on questions with different kinds of reasoning. The results reveal that questions solely based on arithmetic reasoning exhibit significantly higher performance compared to those involving additional forms of reasoning. Specifically, hypothetical reasoning presents a greater challenge than commonsense reasoning. Moreover, questions that require the integration of all three reasoning types exhibit the poorest performance.

**Arithmetic Reasoning** The performance on questions that exclusively require arithmetic reasoning across various arithmetic operations is presented in Figure 3-(2). The findings indicate that subtraction and division pose greater difficulty compared to addition and multiplication.

**Commonsense Reasoning** On commonsense reasoning, in Figure 3-(3), we compare the performance of GPT-3.5+CT-3 on such questions under two settings. The first setting involves directly inputting the question itself, while the second setting involves inputting the concatenation of the knowledge and the question. The results reveal that explicitly stating the knowledge within the question can aid in generating correct SQL queries. This suggests that leveraging external knowledge bases could be beneficial in solving similar questions. However, incorporating external knowledge into

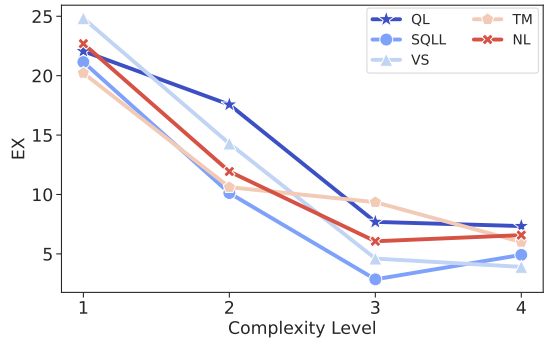


Figure 4: GPT-3.5 + CT-3 execution accuracy performance w.r.t different complexity level. The abbreviations used are as follows: QL for the average question length (1: [0,15)], 2: [15,20), 3:[30,45), 4: [45,)), SQLL for the average SQL length (1: [0,50)], 2: [50,100), 3:[100,150), 4: [150,)), VS for the average number of value slots per question (1: [0,3)], 2: [3,6), 3:[6,9), 4: [9,)), TM for the average number of tables mentioned in each SQL (1: [0,2)], 2: [2,3), 3:[3,5), 4: [5,)), NL for the average nested level per SQL (1: [0,1)], 2: [1,2), 3:[2,3), 4: [3,)).

text-to-SQL tasks presents significant challenges in general. Firstly, models need to compare information from natural language questions with the relational database to determine if external knowledge is required. Secondly, models need to extract the most relevant knowledge from external knowledge bases. Last but not least, the process of integrating this knowledge into the text-to-SQL generation process remains largely unexplored.

**Hypothetical reasoning** On hypothetical reasoning, in Figure 3-(4), we compare the performance of these questions and observe a significant performance gap. The EX performance on factual questions exceeds 17%, whereas the performance on hypothetical questions falls below 7%, confirming the difficulties involved.

### 6.3 Complexity Factors Analysis

To gain insights into the SQL complexity within Archer, Figure 4 illustrates the relationship between the EX score and various factors, including question length, SQL length, number of value slots, number of tables mentioned in SQL, and SQL nested level. The performance demonstrates a decreasing trend as the question becomes longer, the SQL length increases, the number of value slots rises, the number of tables mentioned in the SQL grows, or the SQL nested level escalates. As shown in Table 1, Archer exhibits considerably higher complexity across these factors when compared to other publicly available text-to-SQL datasets.

### 6.4 Bad Case Analysis

We randomly selected 50 executable but incorrect examples generated by GPT-4 + DIN-SQL and identified the following common error types:

**Incorrect Logic** : GPT-4 sometimes struggles with hypothetical questions that involve complex logic. For instance, when asked "If all cars produced by Daimler Benz company are 4-cylinders, which 4-cylinder car needs the most fuel to drive 300 miles?", the model might generate SQL queries like `WHERE T1.Cylinders = 4 AND T4.Maker = 'Daimler Benz'`. However, the correct query should be `WHERE T1.Cylinders = 4 OR T4.Maker = 'Daimler Benz'` as there could be other 4-cylinder cars aside from Mercedes-Benz. This reveals a limitation in comprehending the hypothetical nature of the question.

**Incorrect Knowledge** : GPT-4 may make commonsense errors when generating the SQL, such like unit conversions. For example, if a question requests fuel consumption in liters per hundred kilometers, but the database only contains fuel efficiency data in miles per gallon, the accurate conversion formula is `liters_per_hundred_kilometers = 235.2145 / MPG`. However, GPT-4 employs an incorrect formula like `(100 * 3.78541) / MPG`.

**Incorrect Schema Understanding** : GPT-4 sometimes struggles to correctly link query entities to the corresponding database columns. For example, when asked about the "average single cylinder displacement of an 8-cylinder car", GPT-4 might generate a query like `SELECT avg(Edispl) FROM cars_data WHERE Cylinders = 8`. However, in this case, the query should calculate the average single cylinder displacement, like `SELECT AVG(1.0 * Edispl / Cylinders) AS avg_displ FROM cars_data WHERE Cylinders = 8`. This error highlights the need for the model to understand database column names, especially when they involve abbreviations commonly used in real-world databases. (Note that in the Spider dataset, annotators tend to use exact column names in their queries, e.g., What is the average edispl for all Volvos?)

**Other Detail Errors** : For example, GPT-4 may also exhibit minor errors such as forgetting to multiply 1.0 for float calculations.

## 7 Related Work

The earliest text-to-SQL datasets, including ATIS (Dahl et al., 1994; Iyer et al., 2017), GeoQuery (Zelle and Mooney, 1996; Iyer et al., 2017), Scholar (Iyer et al., 2017), Academic (Li and Jagadish, 2014), IMDB (Yaghmazadeh et al., 2017), Yelp (Yaghmazadeh et al., 2017), Advising (Finegan-Dollak et al., 2018) and Restaurants (Giordani and Moschitti, 2012; Tang and Mooney, 2000; Popescu et al., 2003), were limited to a single database. Consequently, models trained on these datasets struggled to generalize to unseen databases as they were tested on the same database used for training. To address such limitations, Zhong et al. (2017) introduced WikiSQL in which the databases in the test set were not present in the training set. However, the SQL queries in WikiSQL were generated automatically using simplified assumptions, which may not fully capture the complexity of real-world queries.

For a comprehensive cross-domain text-to-SQL dataset, Yu et al. (2018) presented Spider dataset, which is currently the most widely used text-to-SQL datasets. However, Spider excludes questions that require external knowledge, like commonsense reasoning and mathematical calculations, which are often essential for real-world applications.

Wang et al. (2020) proposed DuSQL, a Chinese cross-domain text-to-SQL dataset that includes math-related questions. However, DuSQL's queries and questions are relatively simple due to automatic generation and grammar restrictions. Dou et al. (2022) extended DuSQL with external knowledge in their KnowSQL dataset. Unfortunately, KnowSQL is not publicly unavailable.

In real-life scenarios, databases can be dirtier with abbreviated and obscure naming of tables, columns, and data values. To address this, Lee et al. (2021) proposed KaggleDBQA with realistic databases. Li et al. (2023c) proposed BIRD benchmark for the text-to-SQL task on big and dirty databases with a total size of 33.4 GB.

In contrast to these existing text-to-SQL datasets, Archer focuses specifically on questions involving complex reasoning and offers both English and Chinese questions to query English databases across various domains. Notably, all questions and SQL queries in Archer are manually annotated by humans and thoroughly reviewed by professionals, ensuring high-quality annotations for training and evaluation purposes.

In the solution space, there are both LLM based solutions and solutions based on fine-tuned models. The former solutions, such as DIN-SQL (Pourreza and Rafiei, 2024), tend to perform better in existing text-to-SQL datasets, while the latter ones, particularly FastRAT (Vougiouklis et al., 2023), can offer significant improvements on latency, while keeping decent performance. There can be space combining the above two kinds of solutions for Archer, which is a promising direction for future work.

## 8 Conclusion

In this paper, we present Archer, a complex bilingual text-to-SQL dataset with three distinct reasoning types: arithmetic, commonsense, and hypothetical reasoning. Experimental results on Archer, obtained from both LLMs and fine-tuned models, suggest plenty of space for improvement.

## Acknowledgement

This work is supported by Huawei's Dean's Funding (C-00006589) and the UKRI Centre for Doctoral Training in Natural Language Processing, funded by the UKRI (grant EP/S022481/1).

## Limitations

The evaluation metric used in Archer is execution accuracy. This metric may be perceived as an upper-bound performance measure, as SQL queries producing the same execution results on a single database may still possess different semantic meanings. To overcome this limitation, we plan to release a test suite in the future that evaluates SQL queries on multiple databases, allowing for a more comprehensive assessment of semantic accuracy.

## Ethics Statement

As mentioned in the submission, we select our databases from Spider (Yu et al., 2018), which is public for academic use and does not contain sensitive information. The construction of our dataset involved the active involvement of human participants. We recruited and provided training to five annotators who possessed backgrounds in databases. These annotators were assigned the tasks of generating questions based on the databases, writing SQL queries, and paraphrasing the questions. Importantly, no sensitive personal information was involved throughout this process. Our human annotation study underwent evaluation by the departmental ethics panel, which deemed it exempt from

ethical approval. This exemption was based on the fact that all participants were employees of the University of Edinburgh and were therefore protected by employment law. Furthermore, participants received compensation at the standard hourly rate designated for tutors and demonstrators at the university. To promote academic usage, we intend to freely release the dataset online.

## References


Hiba Arnaout, Simon Razniewski, Gerhard Weikum, and Jeff Z. Pan. 2022. UnCommonSense: Informative Negative Knowledge about Everyday Concepts. In *Proc. of the 31st ACM International Conference on Information and Knowledge Management (CIKM 2022)*.

Jiaoyan Chen, Yuxia Geng, Zhuo Chen, Ian Horrocks, Jeff Z Pan, and Huajun Chen. 2021a. Knowledge-aware Zero-shot Learning: Survey and Perspective. In *Proc. of IJCAI 2021*.

Jiaoyan Chen, Yuxia Geng, Jeff Z Pan Zhuo Chen, Yuan He, Wen Zhang, Ian Horrocks, and Huajun Chen. 2023a. Zero-Shot and Few-Shot Learning With Knowledge Graphs: A Comprehensive Survey. pages 653–685.

Jiaoyan Chen, Freddy Lécué, Jeff Z. Pan, Ian Horrocks, and Huajun Chen. 2018. Knowledge-Based Transfer Learning Explanation. In *Proc of KR*, pages 349–358.

Zhuo Chen, Jiaoyan Chen, Yuxia Geng, Jeff Z Pan, Zonggang Yuan, and Huajun Chen. 2021b. Zero-shot visual question answering using knowledge graph. In *Proc of ISWC*, pages 146–162.

Zhuo Chen, Yufeng Huang, Jiaoyan Chen, Yuxia Geng, Yin Fang Wen Zhang, Jeff Z Pan, and Huajun Chen. 2023b. Duet: Cross-modal Semantic Grounding for Contrastive Zero-shot Learning. In *Proc of AAAI*, pages 405–413.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. 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*.

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

Catherine Finegan-Dollak, Li Zhang Jonathan K. Kummerfeld, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360.

Yuxia Geng, Jiaoyan Chen, Xiang Zhuang, Zhuo Chen, Jeff Z Pan, Juan Li, Zonggang Yuan, and Huajun Chen. 2023. Benchmarking Knowledge-driven Zero-shot Learning.

Alessandra Giordani and Alessandro Moschitti. 2012. Automatic generation and reranking of sql-derived answers to nl questions. In *Proceedings of the Second International Conference on Trustworthy Eternal Systems via Evolving Software, Data and Knowledge*, pages 59–76.

Yong Guan, Freddy Lecue, Jiaoyan Chen, Ru Li, and Jeff Z. Pan. 2024. Knowledge-aware Neuron Interpretation for Scene Classification. In *Proc of AAAI*, pages 405–413.

Jie He, Simon Chi Lok U, Víctor Gutiérrez-Basulto, and Jeff Z. Pan. 2023. BUCA: A Binary Classification Approach to Unsupervised Commonsense Question Answering. In *Proc. of the 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023)*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. *arXiv preprint arXiv:1704.08760*.

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 (Volume 1: Long Papers)*, pages 2261–2273, Online. Association for Computational Linguistics.

Fei Li and H. V. Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. *arXiv preprint arXiv:2302.05965*.

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

Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, et al. 2023c. Can llm already

serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *arXiv preprint arXiv:2305.03111*.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4870–4888.

J. Z. Pan, G. Vetere, J.M. Gomez-Perez, and H. Wu, editors. 2017a. *Exploiting Linked Data and Knowledge Graphs for Large Organisations*. Springer.

Jeff Z. Pan and Ian Horrocks. 2003. Web Ontology Reasoning with Datatype Groups. In *Proc. of the 2nd International Semantic Web Conference (ISWC2003)*.

Jeff Z. Pan and Ian Horrocks. 2005. OWL-Eu: Adding Customised Datatypes into OWL. *Journal of Web Semantics*, pages 29–39.

Jeff Z. Pan, Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jiaoyan Chen, Stefan Dietze, Hajira Jabeen, Janna Omeliyanenko, Wen Zhang, Matteo Lissandrini, Russa Biswas, Gerard de Melo, Angela Bonifati, Edlira Vakaj, Mauro Dragoni, , and Damien Graux. 2023. Large Language Models and Knowledge Graphs: Opportunities and Challenges. *Special Issue on Trends in Graph Data and Knowledge. Transactions on Graph Data and Knowledge (TGDK)*, 1:1–38.

Jeff Z. Pan, Giorgos Stamou, Vassilis Tzouvaras, and Ian Horrocks. 2005. f-SWRL: A Fuzzy Extension of SWRL. In *Proc. of the International Conference on Artificial Neural Networks (ICANN 2005), Special section on "Intelligent multimedia and semantics"*.

J.Z. Pan, D. Calvanese, T. Eiter, I. Horrocks, M. Kifer, F. Lin, and Y. Zhao. 2017b. *Reasoning Web: Logical Foundation of Knowledge Graph Construction and Querying Answering*. Springer.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, pages 149–157.

Mohammadreza Pourreza and Davood Rafiei. 2024. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. In *Proc. of NeurIPS 2024*.

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

Julien Romero, Simon Razniewski, Koninika Pal, Jeff Z. Pan, Archit Sakhadeo, and Gerhard Weikum. 2019. Commonsense Properties from Query Logs and Question Answering Forums. In *Proc. of 28th ACM International Conference on Information and Knowledge Management (CIKM 2019)*, pages 1411–1420.

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*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Giorgos Stoilos, Giorgos B. Stamou, and Jeff Z. Pan. 2006. Handling imprecise knowledge with fuzzy description logic. In *Proc. of the Internaltional Workshop on Description Logics*.

Lappoon R. Tang and Raymond J. Mooney. 2000. Automated construction of database interfaces: Intergrating statistical and relational learning for semantic parsing. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 133–141.

Pavlos Vougiouklis, Nikos Papasarantopoulos, Danna Zheng, David Tuckey, Chenxin Diao, Zhili Shen, and Jeff Z Pan. 2023. FastRAT: Fast and Efficient Cross-lingual Text-to-SQL Semantic Parsing. In *Proc. of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (JCNLP-AACL 2023)*, pages 564–576.

Hai Wan, Jinrui Liang, Jianfeng Du, Yanan Liu, Jialing Ou, Baoyi Wang, Jeff Z. Pan, and Juan Zeng. 2021. Iterative visual relationship detection via commonsense knowledge graph. *Big Data Research*, 23.

Kewen Wang, Zhe Wang, Rodney W. Topor, Jeff Z. Pan, and Grigoris Antoniou. 2014. Eliminating Concepts and Roles from Ontologies in Expressive Descriptive Logics. pages 205–232.

Lijie Wang, Ao Zhang, Kun Wu, Ke Sun, Zhenghua Li, Hua Wu, Min Zhang, and Haifeng Wang. 2020. 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. Association for Computational Linguistics.

Zhe Wang, Kewen Wang, Rodney Topor, and Jeff Z. Pan. 2010. Forgetting for Knowledge Bases in DL-Lite. In *Special Issue "Commonsense Reasoning for the Semantic Web" of the Journal of Annals of Mathematics and Artificial Intelligence*, 1-2.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, , and Thomas Dillig. 2017. Sqlizer: Query synthesis from natural language. In *International Conference on Object-Oriented Programming, Systems, Languages, and Applications, ACM*, pages 63:1–63:26.

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

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

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

# A Prompts

***API Doc Prompt:***

```
### Complete SQLite SQL query only and with no explanation
### English SQLite SQL tables, with their properties:
#
# Activity(actid, activity_name)
# Participates_in(stuid, actid)
# Faculty_Participates_in(FacID, actid)
# Student(StuID, LName, Fname, Age, Sex, Major, Advisor, city_code)
# Faculty(FacID, Lname, Fname, Rank, Sex, Phone, Room, Building)
#
### How many more people have participated in mountain climbing activities than chess activities?
SELECT
```

***CT-3 Prompt:***

```
CREATE TABLE Activity (
    actid INTEGER PRIMARY KEY,
    activity_name varchar(25)
)
/* 3 example rows:
SELECT * FROM Activity LIMIT 3;
actid activity_name
770 Mountain Climbing
771 Canoeing
772 Kayaking
*/

CREATE TABLE Participates_in (
    stuid INTEGER,
    actid INTEGER,
    FOREIGN KEY(stuid) REFERENCES Student(StuID),
    FOREIGN KEY(actid) REFERENCES Activity(actid)
)
/* 3 example rows:
SELECT * FROM Participates_in LIMIT 3;
stuid actid
1001 770
1001 771
1001 777

...

-- Using valid SQLite, answer the following questions for the tables provided above.
-- How many more people have participated in mountain climbing activities than chess activities?
SELECT
```

***CT-3 + COT Prompt:***

```
CREATE TABLE Singer (
    singer_id INTEGER PRIMARY KEY,
    nation TEXT not null,
    name TEXT not null,
    age INTEGER not null,
    salary REAL null )
/* 3 example rows:
SELECT * FROM Singer LIMIT 3;
singer_id nation name age salary
0 China Aotian  18 3000
1 Japan Hiroshi 30 2000
2 USA   Harry   28 2500
*/

-- Using valid SQLite, answer the following questions for the tables provided above.
-- How many singers in USA who is older than 27?
The final SQL is: Let's think step by step.
1. The 'older than 27' refers to age > 27 in SQL.
2. 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 age > 27


CREATE TABLE Activity (
    actid INTEGER PRIMARY KEY,
    activity_name varchar(25)
)
/* 3 example rows:
SELECT * FROM Activity LIMIT 3;
actid activity_name
770 Mountain Climbing
771 Canoeing
772 Kayaking
*/

CREATE TABLE Participates_in (
    stuid INTEGER,
    actid INTEGER,
    FOREIGN KEY(stuid) REFERENCES Student(StuID),
    FOREIGN KEY(actid) REFERENCES Activity(actid)
)
/* 3 example rows:
SELECT * FROM Participates_in LIMIT 3;
stuid actid
1001 770
1001 771
1001 777

...

-- Using valid SQLite, answer the following questions for the tables provided above.
-- How many more people have participated in mountain climbing activities than chess activities?
The final SQL is: Let's think step by step.
```

Figure 5: The example of API Doc prompt, CT-3 prompt, and CT-3+COT prompt.

## B Execution Accuracy Algorithm

**Algorithm 1:** Execution Match Check

**Result:** Check if the execution results of p_sql and g_sql against db are equivalent
**Input:** p_sql, g_sql, db

```
if p_sql is not valid for execution against db then
    return False;
else
    Connect to the database db;
    Execute p_sql and store the results in pred_res;
    Execute g_sql and store the results in gold_res;
    Close the database connection;
    if pred_res is exactly equal to gold_res then
        return True;
    else if the number of rows or columns in pred_res and gold_res are different then
        return False;
    else if g_sql contains an outermost ORDER BY clause then
        Compare sets of columns in pred_res and gold_res;
        return True if equivalent, False otherwise;
    else
        Calculate element frequency of each row and column in both pred_res and gold_res;
        Check if every frequency in pred_res is present in gold_res for both rows and columns;
        return True if all frequencies match, False otherwise;
    end
end
```

## C Performance w.r.t Different Reasoning

| Reasoning Types | EN | | | | | | ZH | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GPT-3.5 + API Doc | | GPT-3.5 + CT-3 | | GPT-3.5 + CT-3 + COT | | GPT-3.5 + API Doc | | GPT-3.5 + CT-3 | | GPT-3.5 + CT-3 + COT | |
| | VA | EX | VA | EX | VA | EX | VA | EX | VA | EX | VA | EX |
| A | 78.78 | 23.02 | 83.81 | 24.46 | 79.50 | 25.54 | 84.17 | 19.06 | 88.85 | 21.58 | 77.34 | 19.42 |
| A+C | 86.60 | 14.05 | 85.95 | 13.73 | 78.43 | 15.69 | 89.87 | 12.09 | 91.18 | 16.67 | 76.80 | 16.99 |
| A+H | 78.95 | 9.21 | 83.77 | 7.46 | 67.54 | 4.82 | 83.77 | 6.58 | 92.54 | 6.14 | 64.47 | 4.82 |
| A+C+H | 85.65 | 4.35 | 82.61 | 5.22 | 73.04 | 3.48 | 86.09 | 2.61 | 92.61 | 3.91 | 70.43 | 4.35 |

Table 3: Performance with respect to different reasoning types.

| Reasoning Types | EN | | | | | | ZH | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GPT-3.5 + API Doc | | GPT-3.5 + CT-3 | | GPT-3.5 + CT-3 + COT | | GPT-3.5 + API Doc | | GPT-3.5 + CT-3 | | GPT-3.5 + CT-3 + COT | |
| | VA | EX | VA | EX | VA | EX | VA | EX | VA | EX | VA | EX |
| Addition | 80.95 | 33.33 | 88.10 | 23.81 | 73.81 | 16.67 | 92.86 | 30.95 | 95.24 | 26.19 | 78.57 | 19.05 |
| Subtraction | 72.41 | 15.52 | 80.17 | 22.41 | 75.00 | 23.28 | 81.03 | 16.38 | 87.93 | 13.79 | 80.17 | 13.79 |
| Multiplication | 84.52 | 26.19 | 85.12 | 26.19 | 81.55 | 28.57 | 85.71 | 20.24 | 89.88 | 25.60 | 74.40 | 20.24 |
| Division | 81.25 | 16.07 | 79.46 | 18.75 | 76.79 | 28.57 | 80.36 | 12.50 | 85.71 | 18.75 | 68.75 | 16.96 |

Table 4: Performance with respect to different arithmetic operations on data with arithmetic reasoning only.

| Reasoning Types | EN | | | | | | ZH | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GPT-3.5 + API Doc | | GPT-3.5 + CT-3 | | GPT-3.5 + CT-3 + COT | | GPT-3.5 + API Doc | | GPT-3.5 + CT-3 | | GPT-3.5 + CT-3 + COT | |
| | VA | EX | VA | EX | VA | EX | VA | EX | VA | EX | VA | EX |
| w/o knowledge | 86.19 | 9.89 | 84.51 | 10.07 | 76.12 | 10.45 | 88.25 | 8.02 | 91.79 | 11.19 | 74.07 | 11.57 |
| w/ knowledge | 83.58 | 9.89 | 87.13 | 12.31 | 74.81 | 13.43 | 85.07 | 9.70 | 87.69 | 12.13 | 73.32 | 13.62 |

Table 5: Performance for questions needed commonsense reasoning with and without explicit knowledge input.

| Reasoning Types | EN | | | | | | ZH | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GPT-3.5 + API Doc | | GPT-3.5 + CT-3 | | GPT-3.5 + CT-3 + COT | | GPT-3.5 + API Doc | | GPT-3.5 + CT-3 | | GPT-3.5 + CT-3 + COT | |
| | VA | EX | VA | EX | VA | EX | VA | EX | VA | EX | VA | EX |
| Hypothetical | 82.31 | 6.77 | 83.19 | 6.33 | 70.31 | 4.15 | 84.93 | 4.59 | 92.58 | 5.02 | 67.47 | 4.59 |
| Factual | 82.53 | 17.25 | 83.84 | 18.12 | 79.48 | 20.09 | 87.77 | 15.5 | 89.52 | 17.47 | 79.48 | 16.81 |

Table 6: Performance comparison for hypothetical questions and corresponding factual questions.

# D Archer Examples

***Arithmetic Reasoning***

**Database**

```
# Activity (actid, activity_name)
# Participates_in (stuid, actid)
# Faculty_Participates_in (FacID, actid)
# Student (StuID, LName, Fname, Age, Sex, Major, Advisor, city_code)
# Faculty (FacID, Lname, Fname, Rank, Sex, Phone, Room, Building)
```

**Question**

Among the students who took part in volleyball activities, what is the percentage of those who have the same advisor as Michael Leighton but are from different cities?
参加过排球活动的学生中，和迈克尔·莱顿同一个辅导员但来自不同城市的学生占百分之多少？

**SQL**

```
SELECT 100.0 * COUNT ( DISTINCT ( A.stuid ) ) / ( SELECT COUNT ( DISTINCT ( A.stuid ) ) FROM
Participates_in A JOIN Student B ON A.stuid = B.stuid JOIN Activity C ON A.actid = C.actid WHERE
C.activity_name = "Volleyball" ) AS percent FROM Participates_in A JOIN Student B ON A.stuid = B.stuid
JOIN Activity C ON A.actid = C.actid WHERE B.Advisor = ( SELECT Advisor FROM Student WHERE Fname =
"Michael" AND Lname = "Leighton" ) AND B.city_code != ( SELECT city_code FROM Student WHERE Fname =
"Michael" AND Lname = "Leighton" ) AND C.activity_name = "Volleyball"
```

**Database**

```
# circuits (circuitId, circuitRef, name, location, country, lat, lng, alt, url)
# races (raceId, year, round, circuitId, name, date, time, url)
# drivers (driverId, driverRef, number, code, forename, surname, dob, nationality, url)
# status (statusId, status)
# seasons (year, url) # constructors(constructorId, constructorRef, name, nationality, url)
# constructorStandings (constructorStandingsId, raceId, constructorId, points, position, positionText, wins)
# results (resultId, raceId, driverId, constructorId, number, grid, position, positionText, positionOrder, points, laps, time, milliseconds, fastestLap,
rank, fastestLapTime, fastestLapSpeed, statusId)
# driverStandings (driverStandingsId, raceId, driverId, points, position, positionText, wins)
# constructorResults (constructorResultsId, raceId, constructorId, points, status)
# qualifying (qualifyId, raceId, driverId, constructorId, number, position, q1, q2, q3)
# pitStops (raceId, driverId, stop, lap, time, duration, milliseconds)
# lapTimes (raceId, driverId, lap, position, time, milliseconds)
```

**Question**

Which countries have more than twice as many racing circuits as Japan?
列出赛道数量比日本赛道数量的两倍还要多的国家。

**SQL**

```
SELECT B.country FROM circuits B , ( SELECT COUNT ( * ) AS n_japan FROM circuits B WHERE B.country =
"Japan" ) GROUP BY B.country HAVING COUNT ( * ) > 2 * n_japan
```

Figure 6: The example of Archer data requiring Arithmetic Reasoning.

***Commonsense Reasoning***

**Database**

# circuits (circuitId, circuitRef, name, location, country, lat, lng, alt, url)
# races (raceId, year, round, circuitId, name, date, time, url)
# drivers (driverId, driverRef, number, code, forename, surname, dob, nationality, url)
# status (statusId, status)
# seasons (year, url) # constructors(constructorId, constructorRef, name, nationality, url)
# constructorStandings (constructorStandingsId, raceId, constructorId, points, position, positionText, wins)
# results (resultId, raceId, driverId, constructorId, number, grid, position, positionText, positionOrder, points, laps, time, milliseconds, fastestLap, rank, fastestLapTime, fastestLapSpeed, statusId)
# driverStandings (driverStandingsId, raceId, driverId, points, position, positionText, wins)
# constructorResults (constructorResultsId, raceId, constructorId, points, status)
# qualifying (qualifyId, raceId, driverId, constructorId, number, position, q1, q2, q3)
# pitStops (raceId, driverId, stop, lap, time, duration, milliseconds)
# lapTimes (raceId, driverId, lap, position, time, milliseconds)

**Question**

Find me the name of the circuit which is farthest in distance from the Tropic of Capricorn.
给出距离南回归线距离最远的赛道名称。

**Commonsense Knowledge**

The Tropic of Capricorn lies at 23.4394 degrees south of the Equator. The north latitude is positive, and the south latitude is negative.

**SQL**

```
SELECT name FROM circuits ORDER BY ABS ( lat - ( - 23.4394 ) ) DESC LIMIT 1
```

**Question**

Provide the ID, first name, and number of races for drivers who have competed in at least twice as many races as Allen Berg and have the same nationality as the famous singer Michael Jackson.
请提供参加过的比赛次数至少是艾伦·伯格的两倍且与著名的歌手迈克尔·杰克逊具有相同国籍的车手的ID、名字、比赛次数。

**Commonsense Knowledge**

Michael Joseph Jackson was an American singer, songwriter, dancer, and philanthropist.

**SQL**

```
SELECT A.driverId , forename AS first_name , COUNT ( * ) AS n_races FROM drivers A JOIN results B ON A.driverId = B.driverId GROUP BY A.driverId HAVING COUNT ( * ) >= 2 * ( SELECT COUNT ( * ) FROM drivers A JOIN results B ON A.driverId = B.driverId WHERE A.forename = "Allen" AND A.surname = "Berg" ) AND A.nationality = "American"
```

Figure 7: The example of Archer data requiring Commonsense Reasoning.

***Hypothetical Reasoning***

**Database**

```
# Activity(actid, activity_name)
# Participates_in(stuid, actid)
# Faculty_Participates_in(FacID, actid)
# Student(StuID, LName, Fname, Age, Sex, Major, Advisor, city_code)
# Faculty(FacID, Lname, Fname, Rank, Sex, Phone, Room, Building)
```

**Question**

If no student who is at least 5 years older than Linda Smith ever participated in volleyball activities, among the students who took part in volleyball activities, what is the percentage of those who have the same advisor as Michael Leighton but are from different cities?
假如比琳达史密斯年长至少五岁的学生都没有参加过排球活动，那参加过排球活动的学生中，和迈克尔·莱顿同一个辅导员但来自不同城市的学生占百分之多少？

**SQL**

```
SELECT 100.0 * COUNT ( DISTINCT ( A.stuid ) ) / ( SELECT COUNT ( DISTINCT ( A.stuid ) ) FROM
Participates_in A JOIN Student B ON A.stuid = B.stuid JOIN Activity C ON A.actid = C.actid WHERE
C.activity_name = "Volleyball" AND B.Age < 5 + ( SELECT Age FROM Student WHERE Fname = "Linda" AND Lname
= "Smith" ) ) AS percent FROM Participates_in A JOIN Student B ON A.stuid = B.stuid JOIN Activity C ON
A.actid = C.actid WHERE B.Advisor = ( SELECT Advisor FROM Student WHERE Fname = "Michael" AND Lname =
"Leighton" ) AND B.city_code != ( SELECT city_code FROM Student WHERE Fname = "Michael" AND Lname =
"Leighton" ) AND C.activity_name = "Volleyball" AND B.Age < 5 + ( SELECT Age FROM Student WHERE Fname =
"Linda" AND Lname = "Smith" )
```

**Question**

If the students whose major subject ID is 550 and are older than 20 years old have all participated in soccer activities, what percentage of people who participated in soccer activities are female?
假如主修专业id为550的学生中大于20岁的学生都参加过足球活动，所有参加过足球活动的人中女性占百分之多少？

**SQL**

```
SELECT 100.0 * ( COUNT ( DISTINCT ( id ) ) + ( SELECT COUNT ( DISTINCT ( id ) ) FROM Student WHERE major =
"550" AND age > 20 ) ) / ( ( SELECT COUNT ( DISTINCT ( id ) ) FROM ( SELECT A.FacID AS id , A.actid , B.Sex
FROM Faculty_Participates_in A JOIN Faculty B ON A.FacID = B.FacID UNION ALL SELECT A.stuid AS id , A.actid ,
B.Sex FROM Participates_in A JOIN Student B ON A.stuid = B.stuid WHERE NOT ( B.major = "550" AND B.age > 20 )
) A JOIN Activity B ON A.actid = B.actid WHERE B.activity_name = "Soccer" ) + ( SELECT COUNT ( DISTINCT ( id )
) FROM Student WHERE major = "550" AND age > 20 ) ) AS percent FROM ( SELECT A.FacID AS id , A.actid , B.Sex
FROM Faculty_Participates_in A JOIN Faculty B ON A.FacID = B.FacID UNION ALL SELECT A.stuid AS id , A.actid ,
B.Sex FROM Participates_in A JOIN Student B ON A.stuid = B.stuid WHERE NOT ( B.major = "550" AND B.age > 20 )
) A JOIN Activity B ON A.actid = B.actid WHERE B.activity_name = "Soccer" AND A.Sex = "F"
```

Figure 8: The example of Archer data requiring Hypothetic Reasoning.