ESM+: Modern Insights into Perspective on Text-to-SQL Evaluation in the Age of Large Language Models

Anonymous ACL submission

Abstract

The task of Text-to-SQL enables anyone to retrieve information from SQL databases using natural language. Despite several challenges, recent models have made remarkable advancements in this task using large language models (LLMs). Interestingly, we find that LLM-based models without fine-tuning exhibit distinct natures compared to their fine-tuned counterparts, leading to inadequacies in current evaluation metrics to accurately convey their performance. Thus, we analyze the two primary metrics, Test Suite Execution Accuracy (`EXE`) and Exact Set Matching Accuracy (`ESM`), to examine their robustness for this task and address shortcomings. We compare the performance of 9 LLM-based models using `EXE`, the original `ESM`, and our improved `ESM` (called `ESM+`). Our results show that `EXE` and `ESM` have high false positive and negative rates of 11.3% and 13.9%, while `ESM+` gives those of 0.1% and 2.6% respectively, providing a significantly more stable evaluation. We release the `ESM+` script as open-source for the community to contribute, while enjoying a more reliable assessment of Text-to-SQL.

1 Introduction

While interacting with SQL databases through natural language interfaces makes them significantly more accessible to non-experts, the task of mapping natural language requests to SQL queries for relational databases, known as Text-to-SQL, remains challenging. Lately, the advent of the transformer (Vaswani et al., 2017) and large language models (LLMs; Brown et al. (2020); Raffel et al. (2020)) has led to momentous advancements in this field. Notably, LLMs have overcome several challenges in Text-to-SQL, as the leaderboard for the Spider dataset (Yu et al., 2018), the most popular benchmark for the task, is primarily dominated by models utilizing LLMs, underscoring their effectiveness in handling complex, multi-table SQL query generation that previous approaches had struggled with.

Evaluating Text-to-SQL models is also challenging because SQL equivalence has been shown undecidable (Abiteboul et al., 1995). Text-to-SQL models are tested using two metrics: Test Suite Execution Accuracy (**`EXE`**) and Exact Set Matching Accuracy (**`ESM`**). `EXE` checks if the SQL execution result of the predicted query matches that of the gold standard query. However, `EXE` can yield false positives, as semantically different queries may produce the same execution (Figure 1a). On the other hand, `ESM` assesses the predicted query by comparing sets of keywords and their arguments to those of the gold query. While more rigorous than `EXE`, `ESM` is still prone to false negatives, because SQL queries may be semantically equivalent yet syntactically diverse (Figure 1b). These issues raise the need for a more robust evaluation metric that accurately evaluates the performance of Text-to-SQL models.

```
SELECT name FROM dogs;
```

```
SELECT name FROM dogs WHERE age < 100;
```

(a) Semantically distinct queries producing the same execution result, as there are no dogs with age ≥ 100.

```
SELECT MAX(weight) FROM dogs;
```

```
SELECT weight FROM dogs ORDER BY weight DESC
↪ LIMIT 1;
```

(b) Syntactically distinct but semantically equivalent queries for finding the weight of the heaviest dog.

Figure 1: Examples of a false positive yielded by `EXE` (1a) and a false negative yielded by `ESM` (1b).

Models using pretrained LLMs without fine-tuning, such as GPT (henceforth **PLM**), perform particularly well on `EXE`, which is the main metric used on the Spider leaderboard. Surprisingly, they do not show a similar level of performance on `ESM`. This discrepancy is even more pronounced when dealing with a more intricate task, Conversational Text-to-

SQL (Co-SQL), where the leaderboard uses `ESM` as the primary metric (Yu et al., 2019) such that no PLM-based models rank highly, a stark contrast to the Spider leaderboard. Therefore, it is critical to analyze these metrics and refine the most appropriate approach for an accurate evaluation of model performance, especially those generated by PLMs, as the dichotomy between the two metrics disproportionately affects PLM-based models more than models using fine-tuned LLMs (henceforth, **FLM**).

This paper first examines potential issues in `ESM` and proposes a new enhanced metric, called **`ESM+`**, which addresses many shortcomings present in the original metric (Section 3). Two PLM-based baseline and seven other state-of-the-art models are then evaluated on the Spider and Co-SQL datasets, comparing their performance using `EXE`, `ESM`, and `ESM+` (Section 4). Finally, a comprehensive error analysis is conducted on the evaluation results using these three metrics, revealing the superior robustness of `ESM+` (Section 5). We posit that `ESM+` will serve as a pivotal metric for assessing the real capabilities of LLM-based Text-to-SQL models, thereby enabling them to reach new heights of performance.[1]

2 Related Work

2.1 Text-to-SQL Models

The current state-of-the-art performance has been achieved by PLM-based models using GPT (OpenAI et al., 2024). Dong et al. (2023) introduced C3, which employs schema linking to rank tables and columns based on their relevance to the question, and prompts GPT to generate the final query. Pourreza and Rafiei (2023) proposed DIN-SQL that predicts schema links to determine which database components will be in the final query. It then classifies the query's difficulty and prompts GPT with one of four templates based on this difficulty to generate the final query, followed by another prompt for output debugging. Gao et al. (2023) presented DAIL-SQL, which searches for similar questions in the training set and uses them to create a few-shot prompt with GPT to generate an initial query. This is then used to find more similar queries in the training set, and the most similar ones are used in a second few-shot prompt to generate the final query. Despite achieving high ranks on the Spider leaderboard, evaluated on `EXE` (Zhong et al., 2020), none of these PLM-based models appear on the CoSQL leaderboard, evaluated on `ESM` (Yu et al., 2019).

Several FLM-based models, such as fine-tuned T5 (Raffel et al., 2020), have also been introduced, showing comparable results to PLM-based models on Spider. Qi et al. (2022) proposed RASAT, which incorporates relation-aware self-attention, enabling better understanding of relations between database schemas while inheriting pre-trained weights from T5. Coupled with PICARD (Scholak et al., 2021), RASAT is also a state-of-the-art model for CoSQL. Li et al. (2023b) introduced Graphix-T5, which augments T5 with graph-aware layers to integrate semantic information from transformer blocks with structural information from graph neural networks. Li et al. (2023a) presented RESDSQL that utilizes an encoder to identify relevant schema items and a decoder to first generate the SQL skeleton with keywords, followed by the complete query.

2.2 Evaluation of SQL Equivalence

Although evaluating the equivalence of two queries plays a crucial role in advancing Text-to-SQL models, only a few works have addressed this challenge. Chu et al. (2017) introduced Cosette, an automatic SQL solver that compiles queries over relational tables and checks for their semantic equivalence, producing counterexamples when the queries are not equivalent; however, it supports a limited set of SQL operations. Zhou et al. (2019) presented EQUITAS, an automated verification tool that transforms a wide range of SQL queries into first-order logic and uses a SMT solver to verify equivalence. While computationally more efficient and capable of handling more features than Cosette, its source code is not publicly available for research purposes.

Therefore, the most accessible and widely used automatic evaluation approaches for Text-to-SQL remain `EXE` (Zhong et al., 2020) and `ESM` (Yu et al., 2018). Their combined evaluation script provides options to disable value and distinct checks, which were employed because models at the time struggled with predicting values and using the `DISTINCT` keyword while performing Text-to-SQL. However, despite the proficiency of LLM-based models in handling those aspects, their results in the literature for Spider and CoSQL are still reported with both value and distinct checking disabled. This persistent practice obscures the true performance of LLM-based models in real applications where accurate value prediction and handling of the `DISTINCT` operation are essential.

[1]All our resources, including the new evaluation script and the model outputs, are available through our open-source project: https://github.com/anonymous

3 ESM+: Enhanced Exact Set Matching

For a comprehensive analysis of the two metrics, Test Suite Execution Accuracy (EXE) and Exact Set Matching (ESM), our GPT baseline model (§4.1) is run on the development sets of the Spider (Yu et al., 2018) and Co-SQL (Yu et al., 2019) datasets. Cases of false positives (§3.1) and negatives (§3.2) in ESM are thoroughly examined through this analysis, and addressed in our new metric, ESM+ (§3.3).

3.1 False Positives in ESM

We first analyze the queries predicted by our model along with their gold standard counterparts that are considered equivalent by ESM but not by EXE. Since ESM is a more stringent metric, it is expected that no query pair considered a mismatch by EXE would be considered a match by ESM. Upon closer inspection, however, it becomes evident that ESM has several shortcomings in its evaluation approach.

One major issue is that ESM does not account for JOIN conditions, which are essential parts of many SQL queries. In Figure 2, the two queries produce different outputs such that EXE *correctly* considers them a mismatch. ESM *mistakenly* considers them a match, however, because it ignores the JOIN conditions (t2.breed_code vs. t2.breed_name).

```
SELECT t1.dog_id, t2.breed_name FROM dogs AS
↪ t1 JOIN breeds AS t2 ON t1.breed_code =
↪ t2.breed_code;
```

```
SELECT t1.dog_id, t2.breed_name FROM dogs AS
↪ t1 JOIN breeds AS t2 ON t1.breed_code =
↪ t2.breed_name;
```

Figure 2: A query pair, correctly considered a mismatch by EXE, but mistakenly considered a match by ESM.

Another issue arises when evaluating queries with the DISTINCT keyword. Even when distinct checks are enabled in the ESM script (Section 2.2), it considers DISTINCT only within aggregate keywords, e.g., COUNT or AVE, failing to recognize it in simpler and more commonly used cases (Figure 3).

```
SELECT DISTINCT name FROM dogs;
```

```
SELECT name FROM dogs;
```

Figure 3: A query pair mistakenly considered a match by ESM due to its disregard of the DISTINCT keyword.

Additionally, the ESM script ignores specified LIMIT values even when value checks are enabled (Fig. 4).

```
SELECT transcript_date FROM Transcripts
↪ ORDER BY transcript_date DESC LIMIT 2;
```

```
SELECT transcript_date FROM Transcripts
↪ ORDER BY transcript_date DESC LIMIT 1;
```

Figure 4: A query pair mistakenly considered a match by ESM due to its disregard of the LIMIT values.

3.2 False Negatives in ESM

We also analyze the predicted and gold query pairs that EXE finds equivalent but not ESM. Some of these cases are false positives for EXE, where the queries are semantically distinct but accidentally return the same result when executed. The other cases involve queries that are semantically equivalent but syntactically distinct, causing ESM to mistakenly considers them a mismatch. Many of these false negatives for ESM occur because assessing semantic equivalence is often contingent on certain assumptions about the database. In Figure 5, the queries are semantically equivalent only if the column dog_id is NON_NULL. This can be verified by the database schema, which gives information about tables & columns, primary key-foreign key relationships, and constraints.

```
SELECT count(dog_id) FROM dogs;
```

```
SELECT count(*) FROM dogs;
```

Figure 5: A query pair that is semantically equivalent with a verifiable assumption.

The queries in Figure 3 can be considered a match if the column name in the table dogs is UNIQUE and NON_NULL. To this end, we carefully examine every false negative case and compile verifiable assumptions that are sufficiently general for any database schema to alleviate this challenge (Table 1).

3.3 New Evaluation Metric

We present ESM+, a new evaluation metric based on ESM that addresses all the issues in Sections 3.1 and 3.2, as well as other critical issues. The following summarizes key updates in ESM+; detailed explanations and examples are provided in Appendix A.1:

1. The keywords LEFT JOIN, RIGHT JOIN, OUTER JOIN, and INNER JOIN, previously disregarded by ESM, are now properly considered.
2. ESM rebuilds queries such that all foreign keys become their primary key counterparts, causing incorrect matching. In ESM+, all foreign keys are preserved as they are.

ID	Equivalent Queries	Verifiable Assumptions
1	`SELECT _ FROM t1 WHERE c1 = (SELECT MIN/MAX(c1) FROM t1);` `SELECT _ FROM t1 ORDER BY c1 ASC/DESC LIMIT 1;`	`c1` is `UNIQUE`
2	`SELECT DISTINCT c1 FROM t1;` `SELECT c1 FROM t1;`	`c1` is `UNIQUE`
3	`SELECT c1 FROM t1 WHERE d1 INTERSECT SELECT c1 FROM t1 WHERE d2;` `SELECT c1 FROM t1 WHERE d1 AND d2;`	`c1` is `UNIQUE`
4	`SELECT c1 FROM t1 WHERE d1 UNION SELECT c1 FROM t1 WHERE d2;` `SELECT c1 FROM t1 WHERE d1 OR d2;`	`c1` is `UNIQUE`
5	`SELECT _ FROM t1 WHERE GROUP BY c1,c2,...;` `SELECT _ FROM t1 WHERE GROUP BY c1;`	`c1` is `UNIQUE`
6	`SELECT c1 FROM t1 EXCEPT (q1);` `SELECT c1 FROM t1 WHERE c1 NOT IN (q1);`	`c1` is `UNIQUE` and `NON_NULL`
7	`SELECT COUNT(*) FROM t1;` `SELECT COUNT(c1) FROM t1;`	`c1` is `NON_NULL`
8	`SELECT c1 FROM t1 WHERE c1 is NOT NULL;` `SELECT c1 FROM t1;`	`c1` is `NON_NULL`
9	`SELECT MIN/MAX(c1) FROM t1;` `SELECT c1 FROM t1 ORDER BY c1 ASC/DESC LIMIT 1;`	`t1` is not empty
10	`SELECT * FROM t1;` `SELECT  c1, c2, ... FROM t1;`	`t1` consists of only `c1`, `c2`, ...
11	`SELECT _ FROM t1 WHERE c1 = 'x';` `SELECT _ FROM t1 WHERE c1 = x;`	`x` is a number not starting with zero
12	`SELECT _ FROM t2 WHERE c2 IN (SELECT c1 FROM t1 WHERE d1);` `SELECT _ FROM t1 JOIN t2 ON t1.c1 = t2.c2 WHERE d1;`	**Case 1** (refer to the caption)
13	`SELECT X FROM t1 JOIN t2 on t1.c1 = t2.c2;` `SELECT X from t2;`	**Case 2** (refer to the caption)
14	`SELECT c1 FROM t1 as t;` `SELECT c1 FROM t1 t;`	None
15	`SELECT _ FROM t1 WHERE c1 IN/NOT IN (x, y,...);` `SELECT _ FROM t1 WHERE c1 =/!= x OR/AND c1 =/!= y OR/AND ...;`	None
16	`SELECT t1.c1 FROM table1 JOIN t2 on t1.c1 = t2.c2;` `SELECT t2.c2 FROM t1 JOIN t2 on t1.c1 = t2.c2;`	None
17	`SELECT c1 FROM t1 WHERE c1 IN (SELECT c1 FROM t1 WHERE d1);` `SELECT c1 FROM t1 WHERE d1;`	None
18	`q1;` `q1 UNION/INTERSECT q1;`	None
19	`SELECT _ FROM t1 WHERE c1 BETWEEN x AND y;` `SELECT _ FROM t1 WHERE c1 >= x/y and c1 <= x/y;`	None
20	`SELECT _ FROM t1 WHERE c1 !=/>/</>=/<=/= x;` `SELECT _ FROM t1 WHERE NOT c1 =/<=/>=/</>/!= x;`	None

Table 1: Equivalent queries with verifiable assumptions implemented in ESM+. **t***: table, **c***: column, **d***: condition, q*: full query. **Case 1**: a primary key-foreign key relation, where `t1.c1` is the primary key and `t2.c2` is the foreign key. **Case 2**: `t1.c1` must be non-composite and `X` can be any column(s) in `t2`. **/** denotes options, but consistency is required in selecting between options across corresponding elements of the queries.

3. Conditions for any `JOIN` are now assessed that were previously disregarded by ESM.
4. ESM extends schema and alias checks always to the entire query, causing issues with sub-queries where aliases are local. ESM+ properly scopes the schema and alias checks exclusively to their corresponding sub-queries.
5. While ESM checks for `DISTINCT` only within aggregate functions, ESM+ consistently considers it across the entire query (§3.1).
6. The value of `LIMIT` is always checked, which were previously disregarded by ESM (§3.1).
7. ESM allows the keyword `IN` followed by a sub-query, but doesn't allow a list of values. ESM+ properly parses and evaluates value lists within the `IN` keyword.

Additionally, a set of verifiable equivalence rules is devised to enhance false negatives in ESM (§3.2). Table 1 provides a full list of equivalent queries and verifiable assumptions incorporated into ESM+.

```
### You are a sql generator, only output plain SQL code, starting with "SELECT" and nothing else.
### Answer the questions based on the following schema for the database (table (col1 [example value],
col2 [example value],...))
### Only output what is necessary to answer the question, do not output any additional information.
### If you are unable to answer the question, output your best guess.
# continents (ContId* [1], Continent [america])
# countries (CountryId* [1], CountryName [japan], Continent [1]) Foreign key continent references
continents.contid
# car_makers (Id* [1], Maker [amc], FullName [Ford Motor Company], Country [1]) Foreign key country
references countries.countryid
# mode_list (ModelId* [1], Maker [1], Model [amc] UNIQUE) Foreign key maker references car_makers.id
# car_names (MakeId* [1], Model [chevrolet], Make [volkswagen model 111]) Foreign key model references
model_list.model
# cars_data (Id* [1], MPG [18], Cylinders [8], Edispl [307.0], Horsepower [130], Weight [3504],
Accelerate [12.0], Year [1970]) Foreign key id references car_names.makeid

Which companies have three or more models?
SELECT
```

Figure 6: An example of the schema-based prompt used for our PLM baseline models.

4 Experiments

4.1 PLM Baseline Models

We build strong baseline models by using schema-based prompting with two PLMs, GPT 4-Turbo (**GPT4**) and Claude 3-Opus (**CLA3**), and run them on the Spider and CoSQL datasets. These models leverage PLMs' intrinsic capabilities to interpret natural language inputs and generate corresponding queries, without fine-tuning on the target datasets. Figure 6 describes the prompt used by our models; detailed explanation are provided in Appendix A.2.

4.2 Spider Models

Three PLM-based and three FLM-based models are evaluated on the Spider dataset (Yu et al., 2018). Section 2 provides the descriptions of these models. Below are their names as listed on the leaderboard:[2]

- **DAIL** (PLM): DAIL-SQL + GPT4 (Gao et al., 2023) `https://github.com/BeachWang/DAIL-SQL`
- **DIN** (PLM): DIN-SQL + GPT4 (Pourreza and Rafiei, 2023) `https://github.com/MohammadrezaPourreza/Few-shot-NL2SQL-with-prompting`
- **C3** (PLM): C3 + ChatGPT + Zero-Shot (Dong et al., 2023) `https://github.com/bigbigwatermalon/C3SQL`
- **R+N** (FLM): RESDSQL-3B + NatSQL (Li et al., 2023a) `https://github.com/RUCKBReasoning/RESDSQL`
- **G+P** (FLM): Graphix-3B + PICARD (Li et al., 2023b) `https://github.com/AlibabaResearch/DAMO-ConvAI/tree/main/graphix`
- **R+P** (FLM): RASAT + PICARD (Qi et al., 2022) `https://github.com/LUMIA-Group/rasat`

For the development set, we obtain the outputs for DAIL, DIN, C3, and G+P from their repositories, while we reproduce the outputs for R+N and R+P using their sources. For the evaluation set, since it was not released when these models debuted, no model outputs are publicly available for it; thus, we reproduce the outputs for all models on this set.

4.3 CoSQL Models

Three FLM-based state-of-the-art models are evaluated on the CoSQL dataset (Yu et al., 2019):

- **R+P**: the same model as described in Section 4.2
- **RAS**: R+P without PICARD (Scholak et al., 2021)
- **STAR**: the highest scoring model in the leaderboard[3] (Cai et al., 2022) `https://github.com/AlibabaResearch/DAMO-ConvAI/tree/main/star`

Since no PLM-based models have been introduced for this task, our baselines are the only PLM-based models compared against these FLM-based ones.

4.4 Results

Table 2 shows the results of our baseline models (§4.1) and the six Spider models (§4.2) with respect to `EXE`, `ESM`, and `ESM+`. For the development set, R+N performs the highest across all three metrics. DAIL shows a competitive score compared to R+N on `EXE`, although its `ESM` score is 10.5% lower. This discrepancy is diminished to 3.7% with `ESM+`; more importantly, DAIL regains the 2nd-place ranking on `ESM+`, as it does on `EXE`. The trend is quite clear; FLM-based models exhibit 6-7% decreases in performance from `ESM` to `ESM+`, whereas PLM-based models show 1-28% increases. This impact is even more dramatic for simpler models; CLA3 performs relatively well on `EXE` (2.8% lower than R+N) but extremely poor on `ESM` (47.1% lower than R+N), which is substantially recovered on `ESM+` (13.2%

[2] `https://yale-lily.github.io/spider`

[3] `https://yale-lily.github.io/cosql`

Model		Development Set EXE	ESM	ESM+	Evaluation Set EXE	ESM	ESM+	Reported EXE	ESM
GPT4	PLM	73.0 (8)	40.5 (7)	54.8 (8)	71.6 (7)	38.1 (6)	53.6 (6)	-	-
CLA3	PLM	81.3 (5)	33.4 (8)	61.5 (6)	79.1 (5)	32.1 (7)	60.4 (7)	-	-
DAIL	PLM	83.1 (2)	70.0 (4)	71.0 (2)	**83.1 (1)**	66.1 (3)	67.8 (3)	**86.2 (1)**	66.5 (4)
DIN	PLM	82.8 (3)	60.1 (5)	63.6 (5)	82.3 (2)	60.7 (4)	63.9 (4)	85.3 (2)	60.0 (5)
C3	PLM	81.9 (4)	46.9 (6)	60.1 (7)	80.6 (3)	44.6 (5)	58.2 (5)	82.3 (3)	-
R+N	FLM	**84.1 (1)**	**80.5 (1)**	**74.7 (1)**	80.0 (4)	**72.0 (1)**	**69.5 (1)**	79.9 (4)	72.0 (2)
G+P	FLM	80.9 (6)	77.1 (2)	70.8 (3)	-	-	-	77.6 (5)	**74.0 (1)**
R+P	FLM	78.2 (7)	75.2 (3)	67.9 (4)	79.0 (6)	70.6 (2)	68.8 (2)	75.5 (6)	70.9 (3)

Table 2: Model performance on the Spider dataset in %. Column-wise rankings are indicated in parentheses. The **Evaluation Set** columns display the results from the model outputs reproduced by us, while the **Reported** columns show the results on the evaluation set as reported in the respective literature and the leaderboard for those models.

lower than R+N). This is because ESM does not handle query styles that deviate from the Spider dataset as effectively. This has less impact on FLM-based models since they are trained to learn those styles from the training set; however, it has a huge impact on PLM-based models that often produce queries in styles that are not captured in the training set, and yet are still semantically equivalent.

For the evaluation set, the trend between ESM and ESM+ stays consistent. It is evident that PLM-based models dominate FLM-based models on EXE as the best PLM-based model, DAIL, gives a 3.1% higher score than the best FLM-based model, R+N.[4] This dominance is reversed for ESM, where R+N's score is 5.9% higher than DAIL's, although the gap is reduced to 1.7% on ESM+. Note that the EXE scores of PLM-based models decrease from the reported scores to our replicated results, whereas they stay similar or even increase for the FLM-based models. This is due to the high variance in PLM-based approaches, which we discuss further in Section 5.2.

Model		EXE Q	I	ESM Q	I	ESM+ Q	I
GPT4	P	70.0	39.2	45.9	16.7	**54.7**	**22.9**
CLA3	P	**72.9**	**41.0**	38.3	13.7	54.0	21.2
R+P	F	66.9	39.6	58.6	27.0	54.5	**22.9**
RAS	F	63.2	34.1	56.1	25.9	52.2	21.5
STAR	F	28.3	11.3	**59.8**	**30.7**	21.2	6.8

Table 3: Model results on the CoSQL development set in %. **Q/I**: Question/Interaction-level evaluation.

Table 3 illustrates the results of our baseline models and the three CoSQL models (§4.3). For both the question-level and interaction-level evaluations, our simple PLM-based models significantly outperform the FLM-based models on EXE, whereas the trend is again reversed for ESM. Notably, the ESM+ results align with the EXE results, as both GPT4 and CLA3 show state-of-the-art performance alongside R+P on ESM+. When comparing the performance of CLA3 and R+P, their EXE results for the Spider evaluation set are the same, whereas CLA3 scores 6% higher on EXE for the CoSQL dataset. This implies that while their SQL generation abilities are comparable, CLA3 exhibits a superior dialogue context understanding ability, leading to its higher EXE performance for CoSQL. Note that STAR, ranked highest on the CoSQL leaderboard based on ESM, does not produce any value, rendering its results on EXE and ESM+ incomparable to the others.

The performance decrease from ESM to ESM+ for the FLM-based models across all datasets is likely because they are not optimized for the SQL features that ESM does not assess (§3), causing incorrect handling of those (e.g., generating random conditions for JOIN would not have any impact on ESM but it affects ESM+). This highlights the need for a robust evaluation metric, such as ESM+, to facilitate enhancement in the field of Text-to-SQL.

5 Discussions

5.1 Model Evaluation

Upon analysis of why PLM-based models achieve high EXE scores but not on ESM+, we find that they often generate queries that would be equivalent to the gold query under certain verifiable assumptions specific to the particular tables. These assumptions are not enforced in ESM+ because they are not generalizable across different schemas. Nevertheless, they happen to hold true for those tables, leading to false positives in EXE, as the predicted queries are not guaranteed to produce correct results across all schemas. In such cases, ESM+ is a more robust metric than EXE, yielding fewer false positives.

[4] Unfortunately, we were unable to run the G+P model, so its results on the evaluation set are omitted from Table 2.

For Spider, our baseline models give much lower ESM scores compared to most other models. This is because the other models leverage the training set in some way. The FLM-based models are finetuned on the set, thereby imitating its query style. DAIL searches for questions similar to the input from the training set and uses them for few-shot prompting. DIN and C3 employ highly specialized prompts designed with the dataset's style in mind, such as calibration hints and elaborate classification prompting. Since our baselines are built without any specific style in mind, they are much more creative in query generation, which is exactly the type of prediction heavily penalized by ESM. However, it is alleviated with ESM+, as the gaps to the other models become much less stark, providing a more accurate depiction of model efficacy on this task.

This is also the reason why the FLM-based models do not exhibit a performance increase from ESM to ESM+. Since their generation styles closely match the dataset's style and ESM already incorporates necessary verifiable assumptions for this style, only a few of the new verifiable rules introduced in ESM+ are applied to evaluate these outputs. Moreover, the issues addressed in our metric cause certain outputs to be evaluated more strictly, resulting in their ESM+ scores dropping compared to their ESM scores.

5.2 PLM Variance

The discrepancy between the published results and our reproduced results on EXE for the PLM-based models in Table 2 is due to the high natural variability inherent in PLMs, such as GPT and Claude. This variability not only hinders the replicability of the work but also creates a situation where, given enough attempts, even a worse model can outperform a more consistent model. This is exacerbated when EXE is used as the primary evaluation metric, since many of the tables do not have sufficient edge cases to catch all the false assumptions made by these models. Given this high variability, we recommend evaluating PLM-based models multiple times and reporting their average scores with variances, rather than solely reporting the top scores, which does not represent their practical performance.

With ESM+, however, we see that the discrepancy decreases drastically. ESM+ aims to reduce the variance in PLMs by being more stringent, akin to ESM, so that it forces the model to generate a query that *always* predicts the correct values, which is much more challenging, but leads to less variance in the model outputs when evaluated under ESM+.

5.3 Error Analysis

To understand whether our new metric gives a more accurate evaluation, we perform an analysis of the false positives and false negatives that each metric produces for each model on the Spider evaluation set. Since disabling distinct and value checks leads to an abundance of false positives in both EXE and ESM due to not considering those conditions, and most current state-of-the-art models predict values, we analyze them with those checks enabled.

Model		EXE		ESM		ESM+	
		FP	FN	FP	FN	FP	FN
GPT4	P	12.8	0.0	1.8	21.0	0.1	3.7
CLA3	P	12.1	0.0	0.6	32.0	0.6	4.6
DAIL	P	10.7	0.0	2.1	8.2	0.1	2.0
DIN	P	13.2	0.0	2.4	10.8	0.1	2.3
C3	P	15.3	0.0	2.0	19.3	0.3	3.3
R+N	F	7.2	0.0	3.4	3.0	0.1	1.0
R+P	F	7.7	0.0	1.1	3.1	0.1	1.0

Table 4: False positives and negative rates (%) for all models with respect to the three metrics on the Spider evaluation set. ESM and EXE are evaluated with distinct and value checking enabled.

Table 4 presents the error analysis results. Despite enabling distinct and value checks, EXE and ESM still yield a high volume of false positives and false negatives, respectively. For all models, the amount of false positives from EXE and false negatives from ESM decreases significantly in ESM+. The decrease in false positives from EXE stems from the new constraints in ESM+ that correctly identify mismatches. The false negative decrease from ESM is attributed to our equivalence rules in Table 1. Lastly, the decrease in false positives from ESM is due to the fixes of the issues described in Section 3.2.

Most models tend to make assumptions that are not verified within the schema, such as the uniqueness of columns, which causes false positives upon execution. On both EXE and ESM, the false positives and false negatives disproportionately affect certain models more than others. However, ESM+ exhibits a notably smaller discrepancy among the best and worst models, implying that ESM+ is a less biased metric than either EXE or ESM. When comparing individual models for ESM+, our baselines yield the highest false negative rates among the others. This trend follows for C3 and DIN, two other models with high diversity in output, indicating that more equivalence rules can be added to decrease the false negative rate, which we will explore in future work.

5.4 Equivalence Rule Analysis

To look into this further, we perform an analysis of improvement of the ESM+ metric as equivalence rules are added (Figure 7).

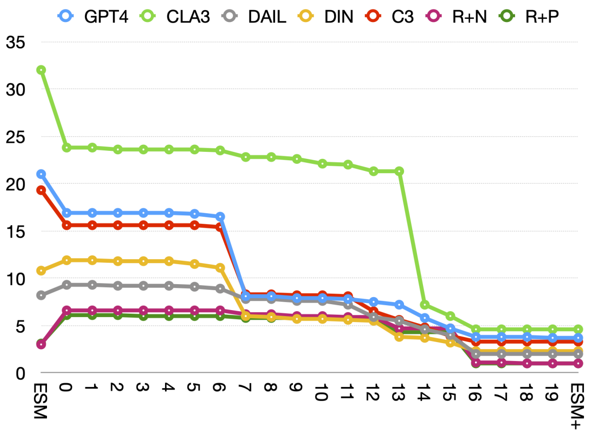

Figure 7: False negative rates on ESM+ (%) as our equivalence rules are accumulated. **ESM**: no update is applied, **0**: ESM with the issues in §3.2 fixed, n: 0 + equivalence rules 1 to n in Table 1 are applied, **ESM+**: 0 + all of 20 equivalence rules are applied, which is our final ESM+.

When no equivalence rules are used, there is already a large decrease in false negative rate from ESM to ESM+ in our baseline models and C3 model due to the fixes in functionality (§3.3). The other models actually have an increased false negative rate with the fixes. Some issues we fixed, like JOIN condition checking, along with the implementation of value and distinct checking, caused certain SQL queries that were previously evaluated as equivalent by ESM to now be evaluated as semantically distinct. However, when the equivalence rules are added, we find that that the queries actually were equivalent. In such cases, ESM had a lower false negative rate than ESM+ with no equivalence rules, but for the wrong reasons. By the same token, in most of those cases ESM had a higher false positive rate. ESM+ evaluates these cases correctly.

The overall trend shows that as expected, each rule we cumulatively add decreases the false negative rate of ESM+. However, the equivalence rules did not have equal impact on the improvement from ESM to ESM+. Certain models benefitted more from certain rules, and others didn't have a reliance on any one rule in particular. The models that directly used Spider's training set (DAIL, R+N, R+P) all had the smallest reduction in false negative rate with any one individual rule. Rule 13 (which entailed unecessary use of JOIN) was the most important addition for these models because even the training set was not consistent in whether the JOIN keywords it used were necessary. Thus, the models that relied on it had similar levels of variation.

The models that were more reliant on generating without access to examples from the training set had much more reliance on specific rules, indicating a certain preference for styles of SQL queries. All the models based on GPT (GPT4, DAIL, DIN, C3) had Rule 7 as the most useful, indicating that GPT has a bias towards generating COUNT (c1) instead of COUNT (*). CLA3, on the other hand, had a dramatic 14.1% decrease in false negative rate when adding Rule 14, indicating that it prefers to write aliases without the AS keyword.

If the trend continues as new equivalence rules are added, we may see some models surpass others in terms of ESM+ as their false negative rates decline. As more rules are added, the discrepancy in false negatives between the best and worst models decreases, showing that each equivalence rule added reduces bias in ESM+.

6 Conclusion

This study introduces Enhanced Exact Set Matching (ESM+), a novel evaluation metric for Text-to-SQL that overcomes several limitations of the previous metrics, Test Suite Execution (EXE) and Exact Set Matching (ESM). Our findings indicate that ESM+ offers a substantial improvement by reducing the occurrences of both false positives and false negatives that commonly plague the earlier metrics. By adopting the more rigorous approach of ESM and incorporating verifiable equivalence rules to allow query diversity, ESM+ can discern more granular distinction in query correctness, allowing for a more accurate measurement of the semantic accuracy of the generated queries and a better understanding of LLMs' true capabilities in generating SQL queries.

Moving forward, we plan to extend the list of verifiable rules to strengthen ESM+ with the help of community feedback, thereby increasing its robustness in evaluating complex SQL query structures. As we continue to refine and enhance ESM+, our goal is to establish a new standard for evaluating Text-to-SQL models that can accurately represent their practical utility and technical proficiency in real-world applications. With the introduction of ESM+, we hope that more PLM-based approaches will be proposed to tackle CoSQL as well as Spider, as they will no longer be as restricted by the lack of variation enforced by ESM.

7 Limitations

ESM+ inherits certain limitations from ESM that could affect its efficacy, listed here:

1. **Retrieving columns from a subquery:** Queries retrieving columns from the subquery are not properly parsed. An example of this is `SELECT c1 FROM (SELECT * FROM t1)`.

2. **Precedence of conditional statements:** Queries using parentheses to order conditional statements are not always handled correctly. For example, the SQL query `SELECT c1 FROM t1 WHERE c1 = x AND (c2 = y OR c1 = z)` is incorrectly treated the same with and without parentheses.

3. **Aliases:** Only table names can have aliases. In SQL, column names, as well as expressions including aggregates can be given aliases. Although the ESM+ evaluation script will not break upon encountering them (like ESM), it will not consider them when they are actually used.

4. **Additional equivalence rules:** There could be any number of additional equivalence rules to be added to further decrease the false negative rate of ESM+. Missing equivalence rules could punish certain types of generation, leading to inaccurate model evaluation.

Addressing these limitations of ESM+ critical when evaluating the text-to-SQL task.

In addition, while analyzing Spider and CoSQL, we noticed that sometimes the the gold queries make non-verifiable assumptions about the question or the real world (Figure 8).

How many graduates of the school are there?

```
SELECT COUNT(*) FROM students;
```

Figure 8: A question and gold query pair from CoSQL that assumes that every student has graduated. This assumption is not verifiable.

A potential option to mitigate this issue would be to have multiple possible correct queries for each question, in order to allow for a larger array of interpretations of each question. We recognize that Spider 2.0 is under development, and we hope that it corrects this aspect of Spider, but it is crucial to address this issue in both Spider and in CoSQL.

8 Ethical Considerations

We utilized the Spider and CoSQL datasets, which are maintained under the Apache 2.0 license and distributed under the CC BY-SA 4.0 license. Our use of these datasets adhered strictly to the terms specified by these licenses, ensuring compliance with their intended and allowed use.

In conducting this study, we have upheld the highest standards of ethical research, focusing particularly on transparency and fairness. Our methodologies, data sources, and results are fully documented and openly accessible to ensure that our work is transparent, reproducible, and verifiable by the research community. We recognize the potential for biases in automated systems and have rigorously tested our models and methods to identify and mitigate any such biases. We do not collect any personal information or data.

Moreover, we consider the broader impacts and risks of our research, particularly in how such technologies might influence data accessibility and decision-making processes in industries that traditionally rely on SQL expertise. By enhancing the usability of database systems through natural language interfaces, our work aims to help democratize data access, enabling a wider range of users to leverage database resources effectively.

References

Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Zefeng Cai, Xiangyu Li, Binyuan Hui, Min Yang, Bowen Li, Binhua Li, Zheng Cao, Weijie Li, Fei Huang, Luo Si, and Yongbin Li. 2022. Star: Sql guided pre-training for context-dependent text-to-sql parsing. *Preprint*, arXiv:2210.11888.

Shumo Chu, Chenglong Wang, Konstantin Weitz, and Alvin Cheung. 2017. Cosette: An automated prover for SQL. In *8th Biennial Conference on Innovative Data Systems Research, CIDR 2017, Chaminade, CA, USA, January 8-11, 2017, Online Proceedings*. www.cidrdb.org.

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

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *Preprint*, arXiv:2308.15363.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. *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. *Preprint*, arXiv:2301.07507.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turlxey, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. GPT-4 Technical Report.

Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Preprint*, arXiv:2304.11015.

Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Yu Cheng, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. Rasat: Integrating relational structures into pretrained seq2seq model for text-to-sql. *Preprint*, arXiv:2205.06983.

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

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. *Preprint*, arXiv:2109.05093.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Tao Yu, Rui Zhang, He Yang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter S Lasecki, and Dragomir Radev. 2019. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. *Preprint*, arXiv:1909.05378.

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*, Brussels, Belgium. Association for Computational Linguistics.

Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic evaluation for text-to-sql with distilled test suites. *Preprint*, arXiv:2010.02840.

Qi Zhou, Joy Arulraj, Shamkant B. Navathe, William Harris, and Dong Xu. 2019. Automated verification of query equivalence using satisfiability modulo theories. *PVLDB*, 12(11):1276–1288.

A Appendix

A.1 Key updates in ESM+

This section describes the key updates of ESM+ summarized in Section 3.3.

1. We added functionality for ESM+ to parse and interpret the keywords LEFT JOIN, RIGHT JOIN, OUTER JOIN, and INNER JOIN, and include them in equivalence checks. Any query with those keywords is automatically disregarded by ESM.
2. ESM rebuilds queries such that all foreign keys become their primary key counterparts, causing incorrect matching. Foreign keys are not the same as primary keys. Although only data in their primary key counterpart may be present in the foreign key, the foreign key can include different amounts of them, as well as NULL values.
3. Conditions for any JOIN are now assessed. ESM collected information about the join conditions, but never compared the conditions of the two queries. This bug was fixed in ESM+.
4. ESM extends schema and alias checks always to the entire query, causing issues with sub-queries where aliases are local. ESM+ properly scopes the schema and alias checks exclusively to their corresponding sub-queries (Listing 1).

```
SELECT c1 FROM t1 AS t JOIN t2 ON t.c1=t2.c2
↪ WHERE c1 IN (SELECT c3 FROM t3 AS t);
```

Listing 1: ESM evaluates this incorrectly as it does not recognize that t is not only an alias for t3 in the subquery but also for t1 in the outer query.

5. While ESM checks for DISTINCT only within aggregate functions, ESM+ consistently considers it across the entire query (§3.1).
6. The value of LIMIT is always checked, which were previously disregarded by ESM (§3.1).
7. ESM allows the keyword IN followed by a subquery, but doesn't allow a list of values. ESM+ properly parses and evaluates value lists within the IN keyword (Listing 2).

```
SELECT c1 FROM t1 WHERE c1 IN (1, 2, 3);
```

Listing 2: ESM disregards this query as it is not capable of parsing a list of values within the IN keyword.

8. CoSQL's evaluation script replaces all occurrences of the string 'value' with '1' when value checks are disabled, causing execution errors if a column is named value. In ESM+, all values are properly considered.

A.2 PLM Baseline Models

Our prompt is shown in Figure 6. In this notation, columns with an asterisk (*) denote a primary key column. The examples enclosed in brackets represent the database content from that column which best aligns with the user question. This alignment is determined by a two-stage n-gram similarity matching process: initially at the question level, and subsequently at the character level if no matches are found or in the event of ties. This approach is inspired by Qi et al. (2022), who only included database content of the most relevant column to the question. We decided to use examples from every column to give the PLM a more holistic understanding of the database content. After the example within brackets, we include any restrictions from the schema of the database, such as NON_NULL or UNIQUE. The foreign key relations at the end of each table are included to give the PLM an understanding of the underlying structure of the database, following Gao et al. (2023), who asserted that these relations help PLMs with the prediction of JOIN clauses.

A.3 Computational Resources

All experiments replicating existing models that used GPUs were run on two A6000 GPUs, and followed the hyperparameters listed on github for those models (§4.2,4.3). For the baseline models, GPT4 adopted the OpenAI model gpt-4-0125-preview, and CLA3 was based on the Anthropic model claude-3-opus-20240229. Both baseline models used temperature 0 for all experiments.