**Advanced techniques for evaluating Text to SQL generation**

**Introduction**

Systems take natural language questions and generate SQL queries as output. The evaluation of these systems needs to account for the fact that SQL queries can often be written in different ways that are logically equivalent but not syntactically identical.

Traditionally, two evaluation metrics have been primarily used for assessing the accuracy of SQLs. They are **Exact Matching**(EM) and **Execution Accuracy(EX)**. EM measures whether the predicted query as a whole is equivalent to the gold query. The challenge however is it is possible to encounter false negative evaluations since a question might be solvable by multiple syntactically different but semantically identical SQL statements.  Execution accuracy is a more widely used metric, which instead of directly comparing the SQL queries, executes the queries against a database and compares the resulting outputs. This also however has its own set of challenges – such as different column order, additional columns which do not impact the final result, etc. In this paper we address these challenges and introduce new techniques and metrics which can overcome these challenges.

**Traditional Approaches**

**Component Matching**

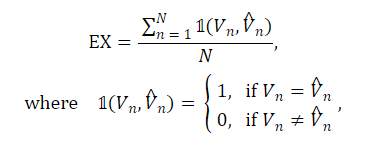
In this approach, SQL query is decomposed into several components based on clause (like SELECT, WHERE, GROUP BY), and set comparison is done for each component.

**Exact matching**

Exact matching conducts string comparison between the predicted and gold SQL queries and measure if the two queries exactly match. It is a special case of component matching, wherein the predicted query is considered correct if all the components of predicted query match the components of the ground truth query.

**Execution Accuracy**

Unlike the previous metrics, this is an execution-based evaluation i.e. instead of directly comparing the SQL statements, the SQL statements are run against a database, and the generated results are then compared. The formula for execution accuracy is as follows:<recreate formula by yourself>

****

Compared to direct SQL query matching, this approach is better, as the same result can be generated by writing SQL code in multiple ways. Component matching or exact matching would not be able to account for that, whereas execution accuracy will. Execution accuracy is widely used across different benchmarks be it Spider, Bird, etc. However, it has its own shortcomings, which are discussed in the following section

**Challenges**

**Challenge 1: Measuring efficiency of SQL query**

While the main objective of LLM-generated SQL queries is to be accurate, it also needs to be optimized for performance, especially when dealing with big data. None of the traditional metrics cover that.

**Challenge 2: Predicted query return additional relevant info**

Many a times, there are multiple results due to ambiguity in a particular question. For example, for the question: What are the top 3 restaurants in New York?, the following 2 queries are both right

SELECT name  
FROM restaurants  
GROUP BY name  
ORDER BY AVG(rating) DESC LIMIT 3

SELECT name, AVG(rating)  
FROM restaurants  
GROUP BY 1  
ORDER BY 2 DESC LIMIT 3`

This is because while the question asks for the top 3 restaurants, the second query also returns the ratings along with the restaurant names, which is still considered correct from a human evaluation perspective, as it returns some extra information along with the original ask. However existing metrics would label the second query wrong.

**Challenge 3: Inherent ambiguity in the question**

Consider the same question : What are the top 3 restaurants in Bengaluru? Consider the following 3 queries for the question

SELECT name  
FROM restaurants  
GROUP BY name  
ORDER BY AVG(rating) DESC LIMIT 3

SELECT id  
FROM restaurants  
GROUP BY name  
ORDER BY AVG(rating) DESC LIMIT 3

SELECT id, name  
FROM restaurants  
GROUP BY name  
ORDER BY AVG(rating) DESC LIMIT 3

All of the 3 queries are correct, because while the question asks for the top 3 restaurants, it does not particularly mention if it wants the restaurant id or the restaurant name. However, previously mentioned techniques would label and second and even the third query wrong, although the give the same information.

**Challenge 4: Do not account for partial match of execution results**

As discussed in both points 2 and 3, many a times the predicted query may return some additional relevant columns along with the column considered “correct”. In these cases, both exact matching and execution accuracy would give a score of zero, although the predicted query is correct.

To account for these different approaches, this paper discusses different approaches and metrics to solve this issue. The approaches can be put into 3 major categories

1. Using better techniques
2. Using better metrics
3. Using LLM as an evaluator (human aligned evaluation)

**Using better techniques**

**Multiple Gold query creation**

As shown in the restaurant example above, sometimes there are multiple columns which are acceptable to answer a given question (e.g. if the result has restaurant id along with the restaurant name, the answer is still equally correct from a human evaluation perspective). To account for these, multiple gold queries can be created, which take into consideration permutations and combinations of acceptable columns and create multiple queries.

SELECT id, name  
FROM restaurants  
GROUP BY name  
ORDER BY AVG(rating) DESC LIMIT 3

SELECT name  
FROM restaurants  
GROUP BY name  
ORDER BY AVG(rating) DESC LIMIT 3

SELECT id   
FROM restaurants  
GROUP BY name  
ORDER BY AVG(rating) DESC LIMIT 3

Once we create multiple gold truths, if the output obtained by running the predicted query matches any one of the outputs of the queries above, the predicted query is considered valid

**Subset Evaluation based Execution Accuracy**

This is a 2 step process which ensures

1. For each column in df1, we check to see if the same column of values exists in df2. So unlike execution accuracy where we compare row wise, here we compare the data column wise. We ignore data types, column names (since these could be [aliased](https://www.w3schools.com/sql/sql_alias.asp)) and row order while doing so.

2. After picking out the relevant columns from df2, renaming them with the names from df1, we check that the overall dataframe matches df1. This is to ensure that we don’t accidentally match on shuffled data columns (which is quite common for column data types with low cardinality like booleans or enums) with the same unordered list of values but from the wrong column.

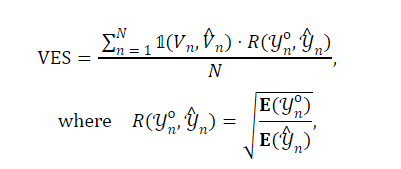
**Alternative metrics**

**Soft F1 Score**

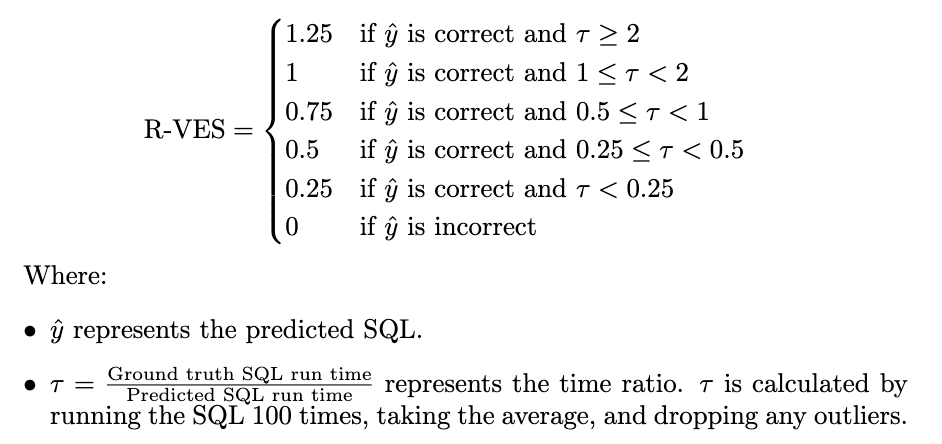
 This metric is specifically designed to assess the performance of text-to-SQL models by measuring the similarity between the tables produced by predicted SQL queries and those from the ground truth. In a nutshell, the soft F1-score is a more lenient metric that reduces the impact of column order and missing values in the tables produced by predicted SQL queries.

true positives (tp) = SUM(Matched)  
false positives (fp) = SUM(Pred\_only)  
false negatives (fn) = SUM(Gold\_only)  
Precision = tp / (tp + fp)  
Recall = tp / (tp + fn)   
F1 = 2 \* Precision \* Recall / (Precision + Recall)

**Valid Efficiency Score**

****

**Reward based Valid Efficiency Score**



**LLM as an evaluator**

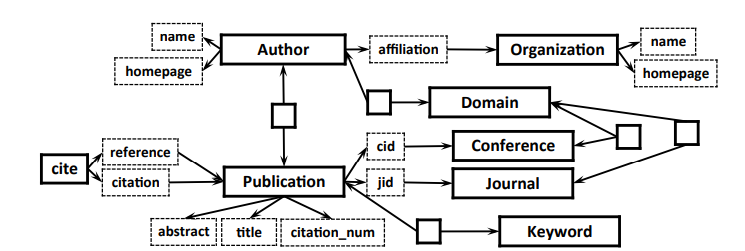
**Accuracy score**

**Experiment**

**Experimental settings**

**Dataset**

For performing the experiment, the academic dataset, published as a part of [1] by Fei Li and H. V. Jagadish is used. A simplified schema of the dataset is shown below



**Experiment Design**

To ensure comprehensive analysis of the different metrics introduced above, 7 different scenarios are considered, listed below

Scenario 1: Predicted sql query is same as golden query

**Question**

What is the average number of citations received by publications in each year?

**Gold query**

SELECT year, AVG(citation\_num) AS avg\_citations

FROM publication

GROUP BY year;

**Predicted query**

SELECT year, AVG(citation\_num) AS avg\_citations

FROM publication

GROUP BY year;

Scenario 2: Predicted sql query is different from golden query, but both give same execution result

**Question**

How does the ratio of publications to journals change over the years? Return the annual numbers of publications and journals as well.

**Gold query**

SELECT publication.year, COUNT(DISTINCT publication.pid) AS num\_publications,

COUNT(DISTINCT publication.jid) AS num\_journals,

CAST(COUNT(DISTINCT publication.pid) AS REAL) / NULLIF(COUNT(DISTINCT publication.jid), 0) AS ratio

FROM publication GROUP BY publication.year ORDER BY publication.year;

**Predicted query**

WITH publication\_count AS (

    SELECT year, COUNT(DISTINCT pid) AS num\_publications

    FROM publication

    GROUP BY year

),

journal\_count AS (

    SELECT year, COUNT(DISTINCT jid) AS num\_journals

    FROM publication

    GROUP BY year

)

SELECT pc.year, pc.num\_publications, jc.num\_journals, pc.num\_publications \* 1.0 / jc.num\_journals AS ratio\_publications\_to\_journals

FROM publication\_count pc

JOIN journal\_count jc ON pc.year = jc.year

ORDER BY pc.year;

Scenario 3: Predicted sql query gives one additional column compared to golden query

**Question**

What is the average number of references cited by publications in each domain name?

**Gold query**

SELECT domain.name, domain.did, AVG(publication.reference\_num) AS average\_references

FROM domain\_publication JOIN publication ON domain\_publication.pid = publication.pid JOIN domain ON domain.did = domain\_publication.did

GROUP BY domain.name, domain.did

**Predicted query**

SELECT d.name AS domain\_name, AVG(p.reference\_num) AS avg\_references\_cited

FROM domain d

JOIN domain\_publication dp ON d.did = dp.did

JOIN publication p ON dp.pid = p.pid

GROUP BY d.name

Scenario 4: Execution result is same but order of columns is different

**Question**

What is the average number of references cited by publications in each domain name?

**Gold query**

SELECT COUNT(DISTINCT publication.pid), publication.year AS total\_publications FROM publication GROUP BY publication.year ORDER BY publication.year

**Predicted query**

SELECT publication.year, COUNT(DISTINCT publication.pid) AS total\_publications FROM publication GROUP BY publication.year ORDER BY publication.year

Scenario 5: Execution result is same but order of rows is different

**Question**

What is the total number of publications published in each year?

**Gold query**

SELECT publication.year, COUNT(DISTINCT publication.pid) AS total\_publications FROM publication GROUP BY publication.year ORDER BY publication.year desc

**Predicted Query**SELECT publication.year, COUNT(DISTINCT publication.pid) AS total\_publications FROM publication GROUP BY publication.year ORDER BY publication.year asc

Scenario 6: Predicted sql query results are partially correct (i.e. only a subset of rows are present)

<add example>

Scenario 7: Predicted sql query results are completely wrong

**Question**

What is the ratio of publications presented in conferences to publications published in journals?

**Gold query**

SELECT CAST(COUNT(DISTINCT CASE WHEN NOT cid IS NULL THEN pid END) AS REAL) / NULLIF(COUNT(DISTINCT CASE WHEN NOT jid IS NULL THEN pid END), 0) AS ratio FROM publication;

**Predicted query**

SELECT  
(SELECT COUNT(\*) FROM publication WHERE cid IS NOT NULL) AS conference\_publications,  
(SELECT COUNT(\*) FROM publication WHERE jid IS NOT NULL) AS journal\_publications;

**Results and Analysis**

**Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Exact Matching** | **Execution Accuracy** | **Multiple gold truth** | **Soft F1 score** |  |
| **Scenario 1** | 1 | 1 | 1 | 1 |  |
| **Scenario 2** | 0 | 1 | 1 | 1 |  |
| **Scenario 3** | 0 |  |  |  |  |
| **Scenario 4** | 0 |  |  |  |  |
| **Scenario 5** | 0 |  |  |  |  |
| **Scenario 6** | 0 | 0 | 0 | 0 |  |

**References**

1. [**https://arxiv.org/html/2403.02951v1#A3.SS1**](https://arxiv.org/html/2403.02951v1#A3.SS1)
2. [**https://defog.ai/blog/open-sourcing-sqleval**](https://defog.ai/blog/open-sourcing-sqleval)
3. [**https://ekzhu.medium.com/human-aligned-text-to-sql-evaluation-399123fa0a64**](https://ekzhu.medium.com/human-aligned-text-to-sql-evaluation-399123fa0a64)
4. [**https://github.com/jkkummerfeld/text2sql-data**](https://github.com/jkkummerfeld/text2sql-data)
5. [**https://blog.premai.io/state-of-text2sql-2024**](https://blog.premai.io/state-of-text2sql-2024)
6. [**https://github.com/taoyds/spider**](https://github.com/taoyds/spider)
7. Constructing an Interactive Natural Language Interface for Relational Databases
8. <https://github.com/bird-bench/mini_dev>

**p = predicted**

**g= gold**

**In accuracy column order matters**

**Questions**

1. **Does F1 score account for different row order**