Details of Semantic Correction Rules

Anonymous Author(s)

The semantic correction rules are designed under the assumption that the tokens predicted by the neural network model are majorly reasonable. Therefore when the generated query violates the rules, we try to correct the generated tokens into a semantically correct from but not simply remove/rewrite the generated tokens. All rules are listed in Table 1.

Table 1: Semantic Correction Rules. Some of the rules are combined for sake of description.

NO.	Rules	Correction
1	The tables corresponding to the column names mentioned in all clauses should appear in the join path.	Add the mentioned tables into the **JOIN** clause and rewrite the join path.
2	If the **FROM** clause consists of a nested query, then all column names mentioned in the outer query should appear in the **SELECT** clause of the inner sub-query.	Add the column names mentioned in the outer query to the **SELECT** clause of the inner query.
3	The aggregation function corresponding to a string column should not be COUNT and SUM	Remove the aggregation function.
4	The **WHERE** operator corresponding to a string value should not be >, <, IN or EXISTS.	Change the operator to =
5	The **WHERE** operator corresponding to a number value should not be LIKE, IN or EXISTS.	Change the operator to =
6	The **WHERE** operator corresponding to a non-aggregate nested sub-query value should IN or NOT IN.	Change the operator to IN or NOT IN. The keyword NOT is speculated according to the original operator.
7	If the query contains the **GROUP BY** clause, all non-aggregate columns mentioned in the **SELECT** clause should appear in the **GROUP BY** clause.	Add all non-aggregate column names in the **SELECT** clause to the **GROUP BY** clause.
8	For any string value in the query, if the **WHERE** operator is =, then the column name should match the value in the database.	Change the column name to the matched one.
9	If the query contains **ORDER BY** clause and no **GROUP BY** clause, and the **SELECT** clause contains aggregate functions, the ordering function is invalid.	Remove the **ORDER BY** clause and add a **WHERE** condition: **WHERE** *column* = (SELECT *AGG*(column) FROM *table*) where *column* is extracted from the **ORDER BY** clause.
10	If the query contains a nested sub query in the **WHERE** clause as **WHERE** c1 op (**SELECT** AGG(c2) $\cdots$), the relationship between c1 and c2 should be the same column, or primary-key-foreign-key.	Copy c1 to the **SELECT** clause of the inner sub-query. If c2 is a boolean column, then add c2 to the **WHERE** clause as **WHERE** c2 = true. If c2 is not a boolean column then remove c2.
11	If the query contains a set operation (**INTERSECT**, **UNION** or **EXCEPT**), the relationship between column names in the **SELECT** clauses of different sub-queries should be either same column, or primary-key-foreign-key.	Copy the column names in the formal sub-query to the latter one.
12	If the relationship between two **WHERE** conditions is AND and the columns of the two conditions are the same, the **WHERE** conditions are invalid.	Convert the connector AND to the set operation **INTERSECT**.
13	If the **FROM** clause contains a circle join such as **FROM** t1 JOIN t2 JOIN t1, then the two **JOIN** conditions should be different.	Choose the conditions with top-2 probabilities predicted by the network.
14	If the query contains the **HAVING** clause but not the **GROUP BY** clause, the query is incorrect.	Add the **GROUP BY** clause to the query.
15	If the query contains a nested sub-query in the **HAVING** clause and the aggregate function is SUM or AVG, the **SELECT** clause in the inner sub-query should also contains a aggregate function.	Copy the aggregate function from the outer query to the inner sub-query.
16	If the query contains a nested sub-query in the **WHERE** or **HAVING** clause and the nested query contains **ORDER** clause but not **LIMIT**, then the ordering operation is invalid.	Add a **LIMIT** number to the inner sub-query based on the outer query or token matching.