A Negative Sampling

As shown in Table 1. For samples for intent recognition, we retain all Unanswerable and Ambiguous types in the training sample generation, while randomly selecting an equal amount from the remaining types. No negative sampling is performed for SQL task samples.

B Error Cases

In our experiments, we manually classified errors in the outputs of three models. A single response may contain multiple errors simultaneously. Table 2 presents representative examples from the three models.

C Prompts

We defined the subsections in Tables 3 and 4 for prompts to form complete prompts.

Prompt for Chain-of-Thought in QDA-SQL

```
Based on {Conversation History} {Current Q-A Type Requirement} {Current Thematic Relation Requirement} {Database Description} {Table Data Example} Based on the above content, referring to {Goal SQL} generate a new dialogue based on the previous conversation. The newly generated SQL can differ from the target SQL. The question you ask needs to be difficult enough that it requires the use of complex SQL to answer it. Output 5 questions and corresponding SQL in the format: [{text:"",sql:""}, {text:"",sql:""}, ...] Output only in JSON format.
```

Note: For answerable types of questions, the LLM needs to generate 5 such questions along with their corresponding SQL answers simultaneously. These answers will be validated through database execution. Only those answers that are executable and yield non-empty results will be retained to ensure that the generated samples contain more meaningful Q&A.

Prompt for Verify and refine in QDA-SQL

Verifying Alignment

```
Based on {Conversation History} {Database Description} {Table Data Example} {Q-A Types Definition} {Multi-turn Dialogue Sample}[-2] and {Multi-turn Dialogue Sample}[-1] Based on the above content, is the classification correct? (For Unanswerable: can this question be achieved with more complex SQL or multi-table join query for valid SQL?) (For Ambiguous: To make sure that the question is indeed impossible to answer directly based on the information above) If correct, output {"type": "yes"} otherwise {"type": "no"} Output nothing else except JSON.
```

Optimizing Expression

```
Based on {Database Description} {Table Data Example} {Q-A Types Definition} {expression and Context Coherence Requirements:} Modify the {Multi-turn Dialogue Sample} based on the above requirements, and output in the same format after revision.
```

Scoring SQL Execution

```
Based on {Conversation History} {Database Description} {Current Question} {SQL answer to Current Question} {SQL Execution Result} Does the SQL and its execution result fully meet the user's needs? Output the rating of the system's response (1-10), output the rating result {"score":""} Output nothing else except JSON.
```

Prompt for Intent Recognition State in StateFlow

```
Based on {Database Description} {Q-A Types Definition} {Conversation History} {Current Question} Categorize and answer user questions based on database structure, output in JSON format {"type":[""], "reply":""} dont́ output other text or explanation
```

Prompt for Intent Solve State in StateFlow

```
Based on {Database Description} {Conversation History} Answer the {Current Question} using only SQL. Carefully check wrong SQL queries and write a new corrected SQL different from {Error}
```

D Licenses for Open-source Databases

We acknowledge that we have exclusively utilized the CoSQL and SParC datasets, both licensed under CC BY-SA 4.0. CC-BY-SA Creative Commons Attribution-ShareAlike 4.0 International This license is one of the open Creative Commons licenses. It allows users to share and adapt the dataset so long as they give credit to the creator and distribute any additions, transformations, or changes to the dataset under this license.

Table 1: Sample Distribution for Question Type Recognition Task and SQL Task

Question Type Recognize Task				
Answerable	**Unanswerable**	**Ambiguous**	**Improper**	**SQL Task**
22054	13797	10406	2149	14,735

Table 2: Case study on error examples generated by GPT-4 and CodeLlama-7B before and after fine-tuning.

Error Type	Example	Example Description
Semantic understanding	**Question:** Cool! Can you show me all of the degree summary names associated with this department? **Ground Truth:** select T2.degree_summary_name from Departments as T1 JOIN Degree_Programs as T2 on T1.department_id = T2.department_id where T1.department_name = 'statistics' **GPT-4:** To provide you with the degree summary names associated with a specific department, I would need to know the department ID or department name you are interested in. Could you please provide that information?	The user asks for degree names for "this department," meaning "statistics." LLM misses this from prior context.
Logical Insufficient database understanding	**Table columns (car_names):** *MakeId* Model Make **Table columns (car_data):** *Id* MPG Cylinders ... **Question:** Which model saves the most gasoline? That is to say, have the maximum miles per gallon? **Ground Truth:** select t1.model from car_names as t1 join cars_data as t2 on t1.makeid = t2.id order by t2.mpg desc limit 1 **GPT-4:** select model, max(mpg) as maxmpg from car_names join cars_data on car_names.make	Correct query joins car_names and car_data using makeid and id. LLM uses the wrong join fields, resulting in incorrect data selection.
Semantic understanding	**Question:** What is the population of Aruba? **Ground Truth:** SELECT population from country where name = "Aruba" **Codellama:** The population of Aruba is approximately 110,000 people, according to the Central Intelligence Agency (CIA) World Factbook.	LLM did not follow the system prompt to retrieve data from the database, leading to an inaccurate response based on external information.
SQL execution Insufficient database understanding	**Table columns (Student):** *StuID* LName FName Age Sex Major Advisor city_code **Question:** What is the city code of student id 1003? **Ground Truth:** SELECT city_code FROM student WHERE StuID = 1003 **Codellama:** SELECT CityCode FROM Student WHERE StuID = 1003	LLM used the incorrect field name CityCode instead of city_code, leading to execution errors due to insufficient understanding of the database schema.
Logical Semantic understanding	**Question:** which is the most famous stadium? **Ground Truth:** SELECT name FROM stadium ORDER BY Highest DESC LIMIT 1 **Codellama-SFT:** SELECT name FROM stadium WHERE Highest >(SELECT avg(Highest) FROM stadium)	LLM incorrectly selected stadiums above the average rather than the most famous one, as the user requested. This resulted in an incorrect condition and a misunderstanding of the question's intent.

Table 3: Subsections of prompts

{Conversation History}
User: What amenities are available in Smith Hall?
System: SELECT T3.amenity_name FROM dorm AS T1 JOIN has_amenity AS T2 ON T1.dormid = T2.dormid JOIN dorm_amenity AS T3 ON T2.amenid = T3.amenid WHERE T1.dorm_name = 'Smith Hall' ORDER BY T3.amenity_name
User: What amenities are available in dorms that house students who are at least 19 and are female?
System:SELECT T3.amenity_name FROM dorm AS T1 JOIN lives_in AS T2 ON T1.dormid = T2.dormid JOIN student AS T4 ON T2.stuid = T4.StuID JOIN has_amenity AS T5 ON T1.dormid = T5.dormid JOIN dorm_amenity AS T3 ON T5.amenid = T3.amenid WHERE T4.Age ¿= 19 AND T4.Sex = 'F' GROUP BY T3.amenity_name ORDER BY T3.amenity_name

{Database Description}
museum (Museum_ID:museum id Type:number—Name:name Type:text—Num_of_Staff:num of staff Type:number—Open_Year:open year Type:text)
visitor (ID:customer id Type:number—Name:name Type:text—Level_of_membership:level of membership Type:number—Age:age Type:number)
visit (Museum_ID:museum id Type:number—visitor_ID:customer id Type:text—Num_of_Ticket:num of ticket Type:number—Total_spent:total spent Type:number)

{Table Data Example}
museum (1,Plaza Museum,62,2000) (2,Capital Plaza Museum,25,2012) (3,Jefferson Development Museum,18,2010)
visitor (1,Gonzalo Higuaín ,8,35) (2,Guti Midfielder,5,28) (3,Arjen Robben,1,27)
visit (1,5,20,320.14) (2,5,4,89.98) (4,3,10,320.44)

{Goal SQL}
Find the first name of students who are living in the Smith Hall.

{Current Question}
What amenities are available in dorms that house students who are at least 19 and are female and live in Smith Hall?

{SQL answer to Current Question}
SELECT T3.amenity_name FROM dorm AS T1 JOIN has_amenity AS T2 ON T1.dormid = T2.dormid JOIN dorm_amenity AS T3 ON T2.amenid = T3.amenid JOIN lives_in AS T4 ON T1.dormid = T4.dormid JOIN student AS T5 ON T4.stuid = T5.stuid WHERE T1.dorm_name = 'Smith Hall' AND T5.age ¿= 19 AND T5.sex = 'F' ORDER BY T3.amenity_name

{SQL Execution Result}
('4 Walls',), ('4 Walls',), ('4 Walls',), ('4 Walls',), ('Carpeted Rooms',), ('Carpeted Rooms',), ('Carpeted Rooms',), ('Carpeted Rooms',), ('Heat',), ('Heat',)

{Current Q-A Type Requirement}
The user's question is clear, and the queried column exists. The system needs to provide an SQL query based on the context.
Note: For instance, an item has been randomly selected, and in this case, it is classified as Answerable.

{Current Thematic Relation Requirement}
The Current Question asks for the same property about another entity.
Note: Randomly select an item, in this case, Participant Shift.

{Expression and Context Coherence Requirements}
1 The user's question and the system's answer match the type of behavior, and the system gives a completely correct and detailed answer that fully satisfies the user's needs.
2 The system's answer is completely relevant to the user's question, with no extraneous or jumping content, and excellent coherence.
3 The system's answers are completely relevant to the user's question, with no irrelevant or off-topic content.
4 The system's answers show a high degree of variability, with a high degree of diversity in language expression and content, and a high degree of innovation and novelty.
5 If there is CANNOT_ANSWER please focus on its question (text) according to the Database Description is not likely to be answered, please determine that this type of question is not possible to generate SQL, you have to modify the Q&A into right type.
6 For AMBIGUOUS questions, making theQ&A more ambiguous makes their CLARIFY follow-up questions more meaningful. makes it completely impossible to generate the corresponding SQL.
7 If the type is incorrectly labeled, please change it. Note: there are only five combinations of type.
isuser:"True" means this is a user question, in the tone of the user's question.
isuser:"false" means this is a system answer, in the tone of the system answer.
Make the text field sentences of user questions and system responses express a tone that is appropriate to their role type and indicates the type of behavior, if it does not match the description, then modify the Q&A, if the deviation is large, then delete the group of Q&A.
Case-by-case checking for fulfillment of requirements to filter and modify the Q&A.

Table 4: Subsections of prompts

{Q-A Types Definition}

The user question type can be the following dialogue behaviors:
INFORM SQL: Users provide requests through natural language questions. If the user's question can be answered through SQL, the system needs to write SQL and explain(don't need to explain based on the SQL results)
INFER SQL: If the user's question must be answered through SQL and human inference. For example, if the user's question is a "yes/no" question, or about "the third oldest...", SQL cannot return the answer directly (or is too complex), but we can infer the answer based on the SQL results.
AMBIGUOUS: The user's question is ambiguous and the system needs to reconfirm the user's intent (e.g., "Did you mean...?") or ask the user to specify which columns or values to return.
AFFIRM: Confirm what the system said (the user agrees/affirms).
NEGATE: Negate what the system said (the user disagrees/denies it).
CANNOT_ANSWER: The question contains additional information not found in the database. (When the user is unfamiliar with the database schema or its implications) The system cannot easily answer the user's question via SQL, and the system informs the user of its limitations. The system needs to detect what information the user needs but is not included in the database.
IMPROPER: Inappropriate questions, such as small talk or questions asking for advice. The system should provide smooth and reasonable daily answers
For the system, define the following dialog behavior:
CONFIRM SQL: The system creates a natural language response describing the SQL and results table and asks the user to confirm that the system understood his/her intent.
CLARIFY: Asks the user to reconfirm and clarify his/her intentions when their question is ambiguous.
REJECT: Tells the user that the system did not understand/cannot answer his/her question, or that the user's question is irrelevant to the topic.
REQUEST MORE: Ask the user if they need more information.
GREETING: Greeting the user.
SORRY: Apologize to the user.
WELCOME: Tell users you are welcome!
Only 5 Q&A combinations are allowed
Type 1:INFER_SQL-CONFIRM_SQL
Type 2:INFORM_SQL-CONFIRM_SQL
Type 3: AMBIGUOUS-CLARIFY-INFER_SQL/INFORM SQL
Type 4:CANNOT_ANSWER-SORRY
Type 5: IMPROPER-REQUEST _MORE /GREETING /SORRY /WELCOME /GOOD_BYE
Note: To diversify the content and make the evaluation process more straightforward, we use the Q-A Types defined by CoSQL during generation. However, in the subsequent evaluation, we simplify these into the four clearer categories presented in this paper. For example, questions of the Answerable type during generation can be INFORM_SQL or INFER_SQL. For specific definitions, refer to (Yu et al. 2019).

{Multi-turn Dialogue Sample}

type: "INFER_SQL" isUser: true text: "What amenities are available in Smith Hall?",
type: "CONFIRM_SQL" isUser: false query:"SELECT T3.amenity_name FROM dorm AS T1 JOIN has_amenity AS T2 ON T1.dormid = T2.dormid JOIN dorm_amenity AS T3 ON T2.amenid = T3.amenid WHERE T1.dorm_name = 'Smith Hall' ORDER BY T3.amenity_name",
type: "INFORM_SQL" isUser: true text: "What amenities are available in dorms that house students who are at least 19 and are female?",
type: "CONFIRM_SQL" isUser: false query:"SELECT T3.amenity_name FROM dorm AS T1 JOIN lives_in AS T2 ON T1.dormid = T2.dormid JOIN student AS T4 ON T2.stuid = T4.StuID JOIN has_amenity AS T5 ON T1.dormid = T5.dormid JOIN dorm_amenity AS T3 ON T5.amenid = T3.amenid WHERE T4.Age ¿= 19 AND T4.Sex = 'F' GROUP BY T3.amenity_name ORDER BY T3.amenity_name",
type: "INFORM_SQL" isUser: true text: "What amenities are available in dorms that house students who are at least 19 and are female and live in Smith Hall?",
type: "CONFIRM_SQL" isUser: false query:"SELECT T3.amenity_name FROM dorm AS T1 JOIN has_amenity AS T2 ON T1.dormid = T2.dormid JOIN dorm_amenity AS T3 ON T2.amenid = T3.amenid JOIN lives_in AS T4 ON T1.dormid = T4.dormid JOIN student AS T5 ON T4.stuid = T5.stuid WHERE T1.dorm_name = 'Smith Hall' AND T5.age ¿= 19 AND T5.sex = 'F' ORDER BY T3.amenity_name"

{Error}

These are the wrong SQL query: Observation Error executing query:SELECT c.‘Maker‘ FROM ‘Car_Names‘ AS c LEFT OUTER JOIN ‘Model_List‘ AS m ON (c.‘Id‘ = m.‘Maker‘) LEFT OUTER JOIN ‘Car_Makers‘ AS k ON (m.‘Maker‘ = k.‘Id‘) WHERE c.‘Country‘ LIKE '%USA%' GROUP BY c.‘Maker‘ ORDER BY COUNT(DISTINCT m.‘Model‘) DESC;-no such column: c.Maker Observation Error executing query:SELECT DISTINCT Maker, Country FROM Car_Makers CROSS JOIN Model_List ML EXCEPT SELECT Maker, COUNTRY FROM cars_Data CD RIGHT JOIN Car_Makers CA MONGO THRU model_LIST WHERE NOT EXISTS (SELECT * FROM cars_DATA EXCEPT SELECT makeID FROM Car_Makers) AND country !=" INTERSECT SELECT maker , COUNTRY FROM Model_List MD INNER JOIN Car_Makers MC OND MD.maker=MC.id;-near "MONGO": syntax error"

References

Yu, T.; Zhang, R.; Er, H. Y.; Li, S.; Xue, E.; Pang, B.; Lin, X. V.; Tan, Y. C.; Shi, T.; Li, Z.; et al. 2019. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. *arXiv preprint arXiv:1909.05378*.