

CoSQL: A Conversational Text-to-SQL Challenge for Natural Language Interfaces to Databases

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 Lasecki, Dragomir Radev

Yale University, Univ. of Michigan, Salesforce Research, Cornell University

Outline

- What is NL2SQL and why **Conversational** NL2SQL?
- Task formulation
- Data collection
- Experiments

NL2SQL Task

Interface system that allows humans to use natural language to query database (DB)



NL2SQL Task

Interface system that allows humans to use natural language to query database (DB)

Which countries in Europe have at least 3 car manufacturers?



NL2SQL Task

Interface system that allows humans to use natural language to query database (DB)

NL2SQL

Which countries in Europe have at least 3 car manufacturers?

```
SELECT T1.country_name
FROM countries AS T1 JOIN continents
AS T2 ON T1.continent = T2.cont_id
JOIN car_makers AS T3 ON
T1.country_id = T3.country
WHERE T2.continent = 'Europe'
GROUP BY T1.country_name
HAVING COUNT(*) >= 3
```



NL2SQL Task

Interface system that allows humans to use natural language to query database (DB)

NL2SQL {
Which countries in Europe have at least 3 car manufacturers?

SELECT T1.country_name
FROM countries AS T1 JOIN continents
AS T2 ON T1.continent = T2.cont_id
JOIN car_makers AS T3 ON
T1.country_id = T3.country
WHERE T2.continent = 'Europe'
GROUP BY T1.country_name
HAVING COUNT(*) >= 3



Execute



Germany

NL2SQL datasets

Great datasets

NL2SQL datasets

Great datasets

- ATIS [Price, 1990; Hemphill et al. 1990; Dahl et al., 1994]
- GeoQuery [Zelle and Mooney, 1996]
- Academic [Li and Jagadish, 2014]
- Scholar [Iyer et al., 2017]
- WikiSQL [Zhong et al., 2017]
- Spider [Yu et al., 2018]



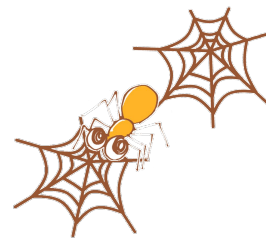
ATIS



Geo



Academic



Spider
(multi-domain)

NL2SQL datasets

Great datasets

- ATIS [Price, 1990; Hemphill et al. 1990; Dahl et al., 1994]
- GeoQuery [Zelle and Mooney, 1996]
- Academic [Li and Jagadish, 2014]
- Scholar [Iyer et al., 2017]
- WikiSQL [Zhong et al., 2017]
- Spider [Yu et al., 2018]
 - Large & multi-domain (10,000+ questions over 200 DBs)
 - Complex questions (covers most SQL keywords, multiple tables, etc.)



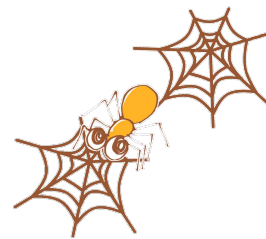
ATIS



Geo



Academic



Spider
(multi-domain)

NL2SQL datasets

Great datasets

- ATIS [Price, 1990; Hemphill et al. 1990; Dahl et al., 1994]
- GeoQuery [Zelle and Mooney, 1996]
- Academic [Li and Jagadish, 2014]
- Scholar [Iyer et al., 2017]
- WikiSQL [Zhong et al., 2017]
- Spider [Yu et al., 2018]
 - Large & multi-domain (10,000+ questions over 200 DBs)
 - Complex questions (covers most SQL keywords, multiple tables, etc.)



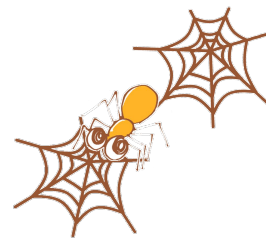
ATIS



Geo



Academic



Spider
(multi-domain)

... but assume all utterances are **single-sentence, well-formed** questions

In practice ...

Complex requests are processed through **interactions** rather than a single utterance

In practice ...

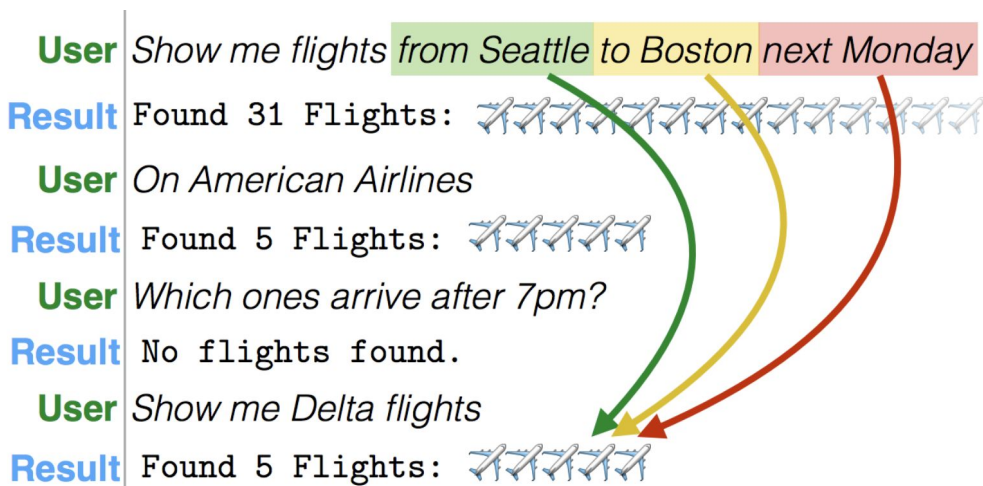
Complex requests are processed through **interactions** rather than a single utterance

- ?? “Show me Delta flights from Seattle to Boston next Monday after 7pm”

In practice ...

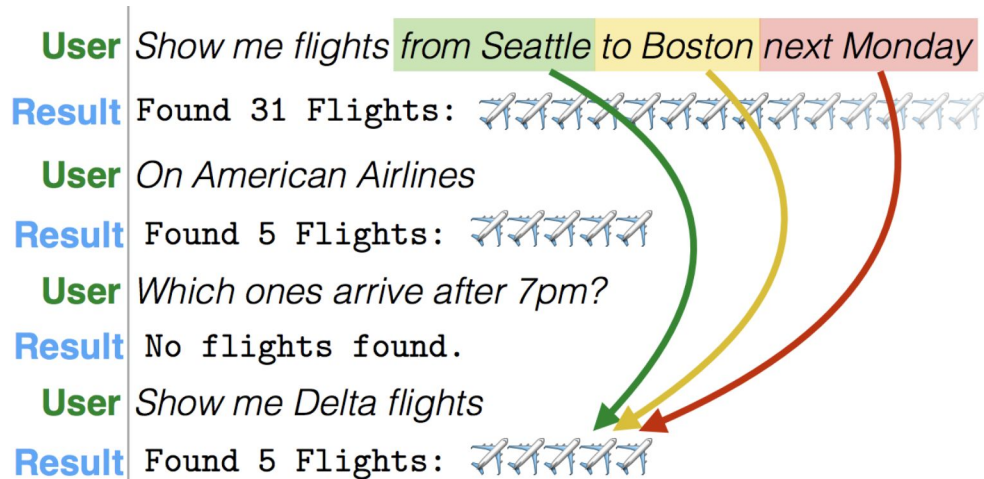
Complex requests are processed through **interactions** rather than a single utterance

- ?? “Show me Delta flights from Seattle to Boston next Monday after 7pm”
- Instead, users would explore DB interactively



How about context-dependent NL2SQL?


ATIS [Price, 1990; Hemphill et al. 1990; Dahl et al., 1994]




How about context-dependent NL2SQL?

ATIS [Price, 1990; Hemphill et al. 1990; Dahl et al., 1994]

User Show me flights from Seattle to Boston next Monday

Result Found 31 Flights: 


User On American Airlines

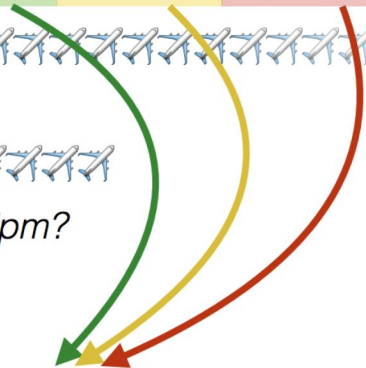
Result Found 5 Flights: 

User Which ones arrive after 7pm?

Result No flights found.

User Show me Delta flights

Result Found 5 Flights: 




- All user questions can be mapped into SQL queries


How about context-dependent NL2SQL?

ATIS [Price, 1990; Hemphill et al. 1990; Dahl et al., 1994]

User Show me flights from Seattle to Boston next Monday

Result Found 31 Flights: 


User On American Airlines

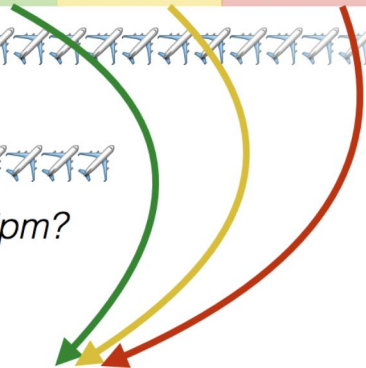
Result Found 5 Flights: 

User Which ones arrive after 7pm?

Result No flights found.

User Show me Delta flights

Result Found 5 Flights: 




- All user questions can be mapped into SQL queries
⇒ cannot handle **ambiguous/unanswerable/unrelated questions** real users may ask


How about context-dependent NL2SQL?

ATIS [Price, 1990; Hemphill et al. 1990; Dahl et al., 1994]

User Show me flights from Seattle to Boston next Monday

Result Found 31 Flights: 


User On American Airlines

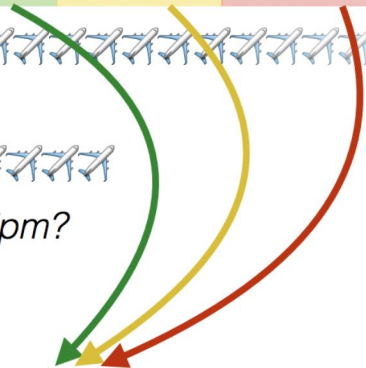
Result Found 5 Flights: 

User Which ones arrive after 7pm?

Result No flights found.

User Show me Delta flights

Result Found 5 Flights: 




- All user questions can be mapped into SQL queries
⇒ cannot handle **ambiguous/unanswerable/unrelated questions** real users may ask
- System only returns exec result.


How about context-dependent NL2SQL?

ATIS [Price, 1990; Hemphill et al. 1990; Dahl et al., 1994]

User Show me flights from Seattle to Boston next Monday

Result Found 31 Flights: 


User On American Airlines

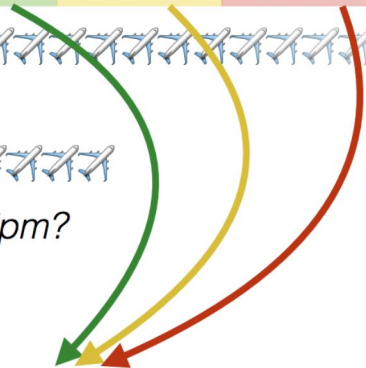
Result Found 5 Flights: 

User Which ones arrive after 7pm?

Result No flights found.

User Show me Delta flights

Result Found 5 Flights: 




- All user questions can be mapped into SQL queries
⇒ cannot handle **ambiguous/unanswerable/unrelated questions** real users may ask
- System only returns exec result.
⇒ Hard for user to check if their questions are interpreted correctly.


How about context-dependent NL2SQL?

ATIS [Price, 1990; Hemphill et al. 1990; Dahl et al., 1994]

User Show me flights from Seattle to Boston next Monday

Result Found 31 Flights: 


User On American Airlines

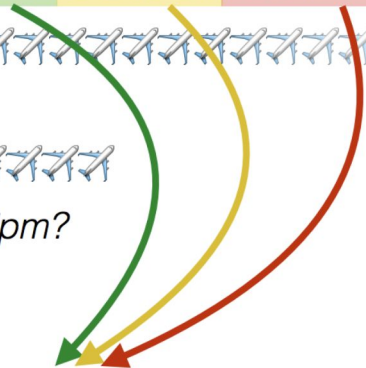
Result Found 5 Flights: 

User Which ones arrive after 7pm?

Result No flights found.

User Show me Delta flights

Result Found 5 Flights: 



- All user questions can be mapped into SQL queries
⇒ cannot handle **ambiguous/unanswerable/unrelated questions** real users may ask
- System only returns exec result.
⇒ Hard for user to check if their questions are interpreted correctly. Want **system response explaining SQL interpretation & exec result**

Our goal: Conversational NL2SQL (CoSQL)

Most previous work expects user query to be well-formed and in a single sentence

But in practice,

- User may prefer to **explore DB** through **interactive exchanges**

Our goal: Conversational NL2SQL (CoSQL)

Most previous work expects user query to be well-formed and in a single sentence

But in practice,

- User may prefer to **explore DB** through **interactive exchanges**
- User may ask **ambiguous/unanswerable/unrelated questions**, and system needs to **detect and inform** user (e.g. ask for clarification)

Our goal: Conversational NL2SQL (CoSQL)

Most previous work expects user query to be well-formed and in a single sentence

But in practice,

- User may prefer to **explore DB** through **interactive exchanges**
- User may ask **ambiguous/unanswerable/unrelated questions**, and system needs to **detect and inform** user (e.g. ask for clarification)
- More user-friendly and interpretable if system **explains the SQL interpretation and execution result** of user questions

Our goal: Conversational NL2SQL (CoSQL)

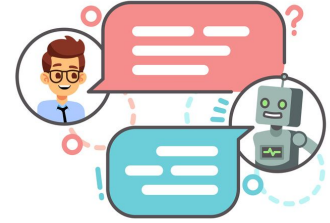
Most previous work expects user query to be well-formed and in a single sentence

But in practice,

- User may prefer to **explore DB** through **interactive exchanges**
- User may ask **ambiguous/unanswerable/unrelated questions**, and system needs to **detect and inform** user (e.g. ask for clarification)
- More user-friendly and interpretable if system **explains the SQL interpretation and execution result** of user questions

⇒ We collect real-world dialog data on Mechanical Turk

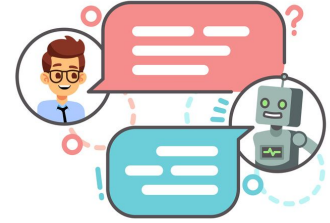
CoSQL (Desiderata & Tasks)



The system needs to:

- I. Understand users' questions, and determine whether the questions can be answered by SQL

CoSQL (Desiderata & Tasks)

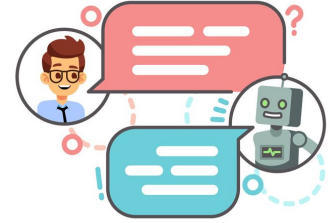


The system needs to:

- I. Understand users' questions, and determine whether the questions can be answered by SQL

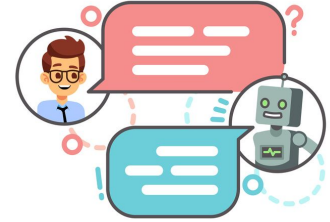
} Dialog act management

CoSQL (Desiderata & Tasks)



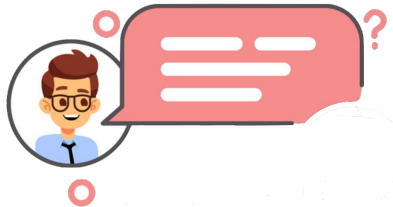
Task 1. Dialog Act Management

CoSQL (Desiderata & Tasks)



Task 1. Dialog Act Management

- Given user's question, the system classify it into an **utterance type**:

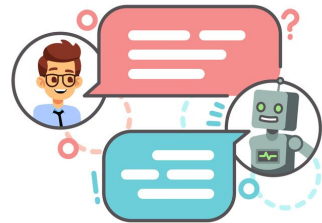


Context-dependent user question



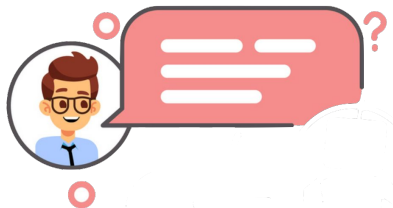
Dialog act type

CoSQL (Desiderata & Tasks)



Task 1. Dialog Act Management

- Given user's question, the system classify it into an **utterance type**:
"INFORM_SQL" (answerable with SQL), "AMBIGUOUS", "GREETING",
"CANNOT_ANSWER", "NOT_RELATED", etc.



Context-dependent user question

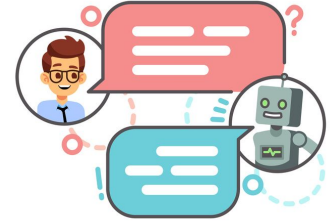
Dialog act type

Task 1: Dialog Act Management

Some interesting **dialog action types** with **examples**:

1. **INFER_SQL**: if the user's question must be answered by SQL+human inference. SQL cannot directly return the answer, but we can infer the answer based on the SQL results.
Examples: users' questions are "are they..?" (yes/no question) or "the 3rd oldest..".
Are there more female or male students overall? Can you list all the events that happened within the last 5 years?
2. **AMBIGUOUS**: the user's question is ambiguous, the system needs to double check the user's intent (e.g. what/did you mean by...?) or ask for which columns to return.
3. **AFFIRM**: affirm something said by the system (user says yes/agree)
4. **NEGATE**: negate something said by the system (user says no/deny)
5. **NOT_RELATED**: the user's question is not related to the database, the system reminds the user
6. **CANNOT_UNDERSTAND**: the user's question cannot be understood by the system, the system asks the user to rephrase or paraphrase question.
7. **CANNOT_ANSWER**: the user's question cannot be easily answered by SQL, the system tells the user its limitation.
Example: What is the average population across the counties? Which county has a population **closest** to that?

CoSQL (Desiderata & Tasks)

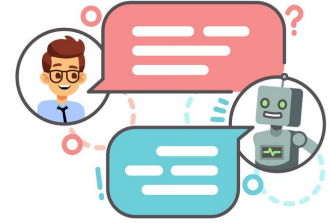


The system needs to:

- I. Understand users' questions, and determine whether the questions can be answered by SQL

} Dialog act management

CoSQL (Desiderata & Tasks)

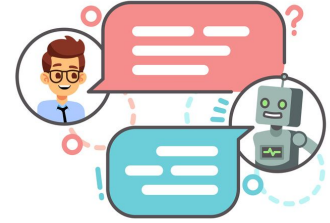


The system needs to:

- I. Understand users' questions, and determine whether the questions can be answered by SQL
- II. If the questions can be answered by SQL, translate them into SQL

} Dialog act management

CoSQL (Desiderata & Tasks)

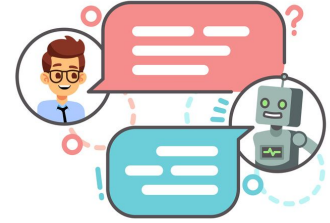


The system needs to:

- I. Understand users' questions, and determine whether the questions can be answered by SQL
- II. If the questions can be answered by SQL, translate them into SQL
- III. If the questions cannot be answered by SQL, inform user

} Dialog act management

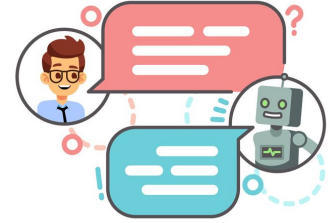
CoSQL (Desiderata & Tasks)



The system needs to:

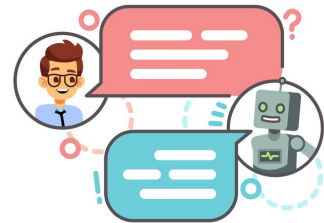
- I. Understand users' questions, and determine whether the questions can be answered by SQL
 - II. If the questions can be answered by SQL, translate them into SQL
 - III. If the questions cannot be answered by SQL, inform user
- } Dialog act management
- } Dialog SQL state tracking

CoSQL (Desiderata & Tasks)



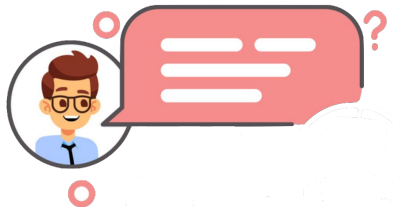
Task 2. Dialog SQL State Tracking

CoSQL (Desiderata & Tasks)



Task 2. Dialog SQL State Tracking

- Given a NL question in dialog → map it to SQL if possible.

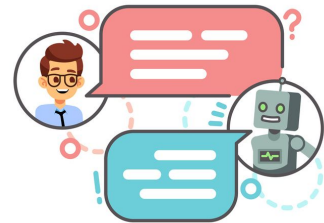


Context-dependent user question



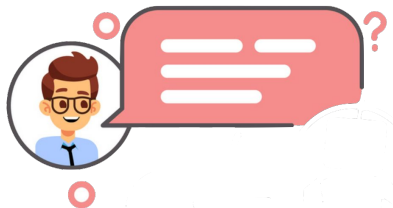
SQL query

CoSQL (Desiderata & Tasks)



Task 2. Dialog SQL State Tracking

- Given a NL question in dialog → map it to SQL if possible.
- Context-dependent semantic parsing

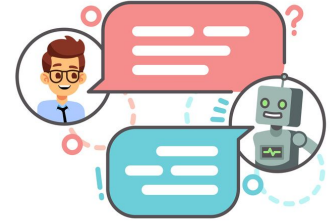


Context-dependent user question

SQL query



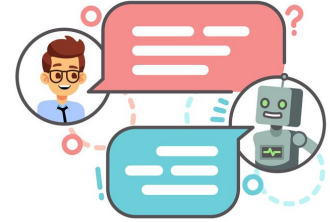
CoSQL (Desiderata & Tasks)



The system needs to:

- I. Understand users' questions, and determine whether the questions can be answered by SQL
 - II. If the questions can be answered by SQL, translate them into SQL
 - III. If the questions cannot be answered by SQL, inform user
- } Dialog act management
- } Dialog SQL state tracking

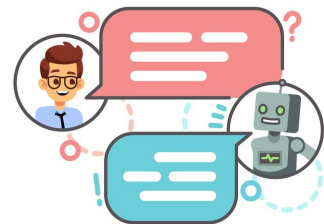
CoSQL (Desiderata & Tasks)



The system needs to:

- I. Understand users' questions, and determine whether the questions can be answered by SQL
 - II. If the questions can be answered by SQL, translate them into SQL
 - III. If the questions cannot be answered by SQL, inform user
 - IV. Show results to users
 - V. Describe how it got the result and what the result means
- } Dialog act management
- } Dialog SQL state tracking

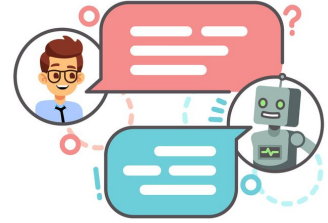
CoSQL (Desiderata & Tasks)



The system needs to:

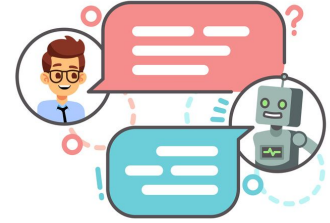
- I. Understand users' questions, and determine whether the questions can be answered by SQL
 - II. If the questions can be answered by SQL, translate them into SQL
 - III. If the questions cannot be answered by SQL, inform user
 - IV. Show results to users
 - V. Describe how it got the result and what the result means
- } Dialog act management
- } Dialog SQL state tracking
- } System response generation

CoSQL (Desiderata & Tasks)



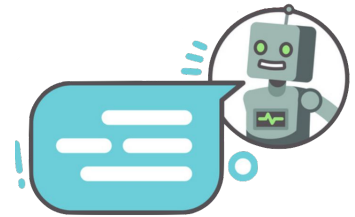
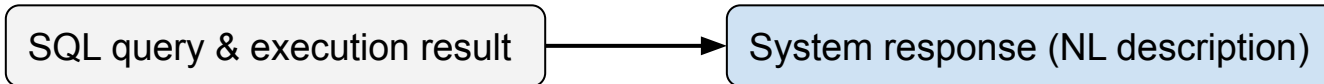
Task 3. System response generation

CoSQL (Desiderata & Tasks)

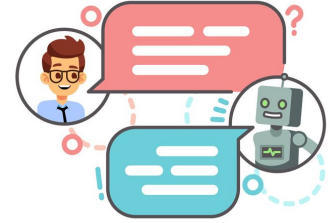


Task 3. System response generation

- SQL query and execution results → generate a NL response that describes the query and returned results

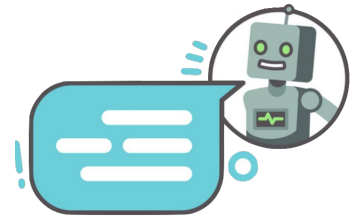
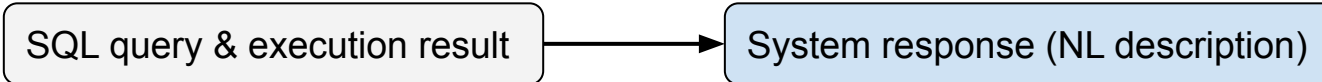


CoSQL (Desiderata & Tasks)



Task 3. System response generation

- SQL query and execution results → generate a NL response that describes the query and returned results
- Allow users to verify if the result answers their questions.



CoSQL (Example)

D_1 : Database about student dormitories containing 5 tables



CoSQL (Example)

D₁ : Database about student dormitories containing 5 tables

Q₁ : What are the names of all the dorms? `INFORM_SQL`

Database



User Question



CoSQL (Example)

D₁ : Database about student dormitories containing 5 tables

Q₁ : What are the names of all the dorms? **INFORM_SQL**

S₁ : **SELECT** dorm_name **FROM** dorm

A₁ : (Result table with many entries)

R₁ : This is the list of the names
of all the dorms. **CONFIRM_SQL**

Database

User Question

Interface System

CoSQL (Example)

D₁ : Database about student dormitories containing 5 tables

Q₁ : What are the names of all the dorms? **INFORM_SQL**

S₁ : **SELECT** dorm_name **FROM** dorm

A₁ : (Result table with many entries)

R₁ : This is the list of the names **CONFIRM_SQL**
of all the dorms.

Q₂ : Which of those dorms have a TV lounge? **INFORM_SQL**

CoSQL (Example)

D₁ : Database about student dormitories containing 5 tables

Q₁ : What are the names of all the dorms? **INFORM_SQL**

S₁ : **SELECT dorm_name FROM dorm**

A₁ : (Result table with many entries)

R₁ : This is the list of the names
of all the dorms. **CONFIRM_SQL**

Q₂ : Which of those dorms have a TV lounge? **INFORM_SQL**

S₂ : **SELECT T1.dorm_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 T3.amenity_name = 'TV
Lounge'**

A₂ : (Result table with many entries)

R₂ : This shows the names of dorms
with TV lounges. **CONFIRM_SQL**

CoSQL (Example)

D₁ : Database about student dormitories containing 5 tables

Q₁ : What are the names of all the dorms? **INFORM_SQL**

S₁ : `SELECT dorm_name FROM dorm`

A₁ : (Result table with many entries)

R₁ : This is the list of the names
of all the dorms. **CONFIRM_SQL**

Q₂ : Which of those dorms have a TV lounge? **INFORM_SQL**

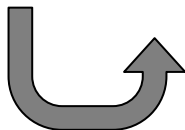
S₂ : `SELECT T1.dorm_name FROM dorm AS T1 JOIN has_amenity AS
T2 ON T1.dormid = T2.dormid JOIN dorm_amenity AS T3 ON
T2.amenid = T3.amenid WHERE T3.amenity_name = 'TV
Lounge'`

A₂ : (Result table with many entries)

R₂ : This shows the names of dorms
with TV lounges. **CONFIRM_SQL**

Q₃ : What dorms have no study
rooms as amenities?

AMBIGUOUS



CoSQL (Example)

D₁ : Database about student dormitories containing 5 tables

Q₁ : What are the names of all the dorms? **INFORM_SQL**

S₁ : `SELECT dorm_name FROM dorm`

A₁ : (Result table with many entries)

R₁ : This is the list of the names of all the dorms. **CONFIRM_SQL**

Q₂ : Which of those dorms have a TV lounge? **INFORM_SQL**

S₂ : `SELECT T1.dorm_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 T3.amenity_name = 'TV Lounge'`

A₂ : (Result table with many entries)

R₂ : This shows the names of dorms with TV lounges. **CONFIRM_SQL**

Q₃ : What dorms have no study rooms as amenities?

AMBIGUOUS

R₃ : Do you mean among those with TV Lounges?

CLARIFY

Q₄ : Yes.

AFFIRM



CoSQL (Example)

D₁ : Database about student dormitories containing 5 tables

Q₁ : What are the names of all the dorms? **INFORM_SQL**

S₁ : **SELECT** dorm_name **FROM** dorm

A₁ : (Result table with many entries)

R₁ : This is the list of the names of all the dorms. **CONFIRM_SQL**

Q₂ : Which of those dorms have a TV lounge? **INFORM_SQL**

S₂ : **SELECT** T1.dorm_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** T3.amenity_name = 'TV Lounge'

A₂ : (Result table with many entries)

R₂ : This shows the names of dorms with TV lounges. **CONFIRM_SQL**

Q₃ : What dorms have no study rooms as amenities? **AMBIGUOUS**

R₃ : Do you mean among those with TV Lounges? **CLARIFY**

Q₄ : Yes. **AFFIRM**

S₄ : **SELECT** T1.dorm_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** T3.amenity_name = 'TV Lounge' **EXCEPT** **SELECT** T1.dorm_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** T3.amenity_name = 'Study Room'

A₄ : Fawlty Towers

R₄ : Fawlty Towers is the name of the dorm that has a TV lounge but not a study room as an amenity. **CONFIRM_SQL**



CoSQL (Example)

D₁ : Database about student dormitories containing 5 tables

Q₁ : What are the names of all the dorms? **INFORM_SQL**

S₁ : `SELECT dorm_name FROM dorm`

A₁ : (Result table with many entries)

R₁ : This is the list of the names of all the dorms. **CONFIRM_SQL**

Q₂ : Which of those dorms have a TV lounge? **INFORM_SQL**

S₂ : `SELECT T1.dorm_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 T3.amenity_name = 'TV Lounge'`

A₂ : (Result table with many entries)

R₂ : This shows the names of dorms with TV lounges. **CONFIRM_SQL**

Q₃ : What dorms have no study rooms as amenities? **AMBIGUOUS**

R₃ : Do you mean among those with TV Lounges? **CLARIFY**

Q₄ : Yes. **AFFIRM**

S₄ : `SELECT T1.dorm_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 T3.amenity_name = 'TV Lounge' EXCEPT SELECT T1.dorm_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 T3.amenity_name = 'Study Room'`

A₄ : Fawlty Towers

R₄ : Fawlty Towers is the name of the dorm that has a TV lounge but not a study room as an amenity. **CONFIRM_SQL**

Q₈ : Thanks! ... **THANK_YOU**

R₈ : You are welcome. **WELCOME**



System Response Generation (example)

Examples

1. If the returned result can be combined with the SQL description, combine them together to generate the response. For example:

Given SQL: `SELECT avg(salary) FROM instructor`

Result Returned: `200k`

System Response Generation (example)

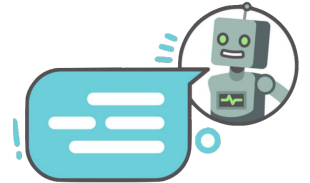
Examples

1. If the returned result can be combined with the SQL description, combine them together to generate the response. For example:

Given SQL: `SELECT avg(salary) FROM instructor`

Result Returned: `200k`

Response: `The average salary of all instructors is 200k.`



System Response Generation (example)

Examples

2. If the returned result is too large and cannot be combined with the SQL description, describe them separately. For example:

Given SQL: `SELECT avg(T1.salary), T1.department_id
FROM instructor as T1 JOIN department
as T2 ON T1.department_id = T2.id
GROUP BY T1.department_id`

Result Returned: **a long table**

System Response Generation (example)

Examples

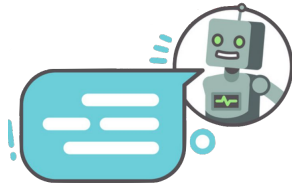
2. If the returned result is too large and cannot be combined with the SQL description, describe them separately. For example:

Given SQL: `SELECT avg(T1.salary), T1.department_id
FROM instructor as T1 JOIN department
as T2 ON T1.department_id = T2.id
GROUP BY T1.department_id`

Result Returned: **a long table**

Response: **Here is the result table that shows the average salary in each department.**

For example, the average of CS professors is 250k.



Task-oriented dialog task v.s. CoSQL

Task-oriented dialog: dialog state-tracking with pre-defined slots and values

```
Topic:'food'  Cuisine:'Chinese'  
Neighborhood:'Palo Alto'
```


Task-oriented dialog task v.s. CoSQL

Task-oriented dialog: dialog state-tracking with pre-defined slots and values

```
Topic:'food'  Cuisine:'Chinese'  
Neighborhood:'Palo Alto'
```

CoSQL: cover **more diverse & complex** semantics of practical user questions with SQL

```
SELECT T2.name, T2.budget  
FROM instructor as T1 JOIN department as T2 ON  
T1.department_id = T2.id  
GROUP BY T1.department_id  
HAVING avg(T1.salary) > (SELECT avg(salary) FROM instructor)
```

Task-oriented dialog task v.s. CoSQL

Task-oriented dialog: dialog state-tracking with pre-defined slots and values

```
Topic:'food'  Cuisine:'Chinese'  
Neighborhood:'Palo Alto'
```

CoSQL: cover **more diverse & complex** semantics of practical user questions with SQL

```
SELECT T2.name, T2.budget  
FROM instructor as T1 JOIN department as T2 ON  
T1.department_id = T2.id  
GROUP BY T1.department_id  
HAVING avg(T1.salary) > (SELECT avg(salary) FROM instructor)
```

Task-oriented dialog task v.s. CoSQL

Task-oriented dialog: dialog state-tracking with pre-defined slots and values

```
Topic:'food'  Cuisine:'Chinese'  
Neighborhood:'Palo Alto'
```

CoSQL: cover **more diverse & complex** semantics of practical user questions with SQL

```
SELECT T2.name, T2.budget  
FROM instructor as T1 JOIN department as T2 ON  
T1.department_id = T2.id  
GROUP BY T1.department_id  
HAVING avg(T1.salary) > (SELECT avg(salary) FROM instructor)
```

Task-oriented dialog task v.s. CoSQL

Task-oriented dialog: dialog state-tracking with pre-defined slots and values

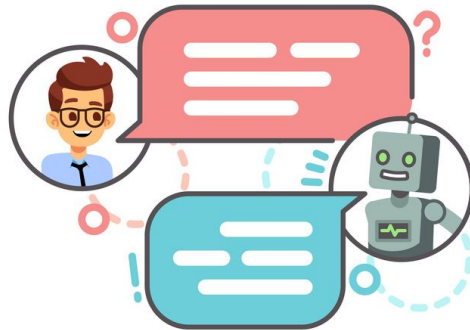
```
Topic:'food'  Cuisine:'Chinese'  
Neighborhood:'Palo Alto'
```

CoSQL: cover **more diverse & complex** semantics of practical user questions with SQL

```
SELECT T2.name, T2.budget  
FROM instructor as T1 JOIN department as T2 ON  
T1.department_id = T2.id  
GROUP BY T1.department_id  
HAVING avg(T1.salary) > (SELECT avg(salary) FROM instructor)
```

Data Collection

- **Wizard of Oz method:** Real-time dialog between two players (**user** & **system**)

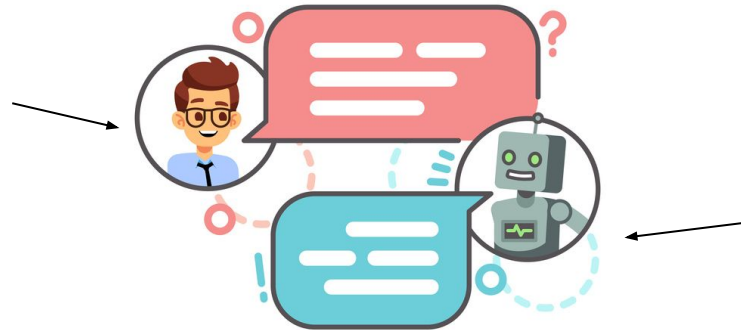


Data Collection

- **Wizard of Oz method:** Real-time dialog between two players (**user** & **system**)

DB User: Crowd Workers

Interface System:
CS students with SQL skill

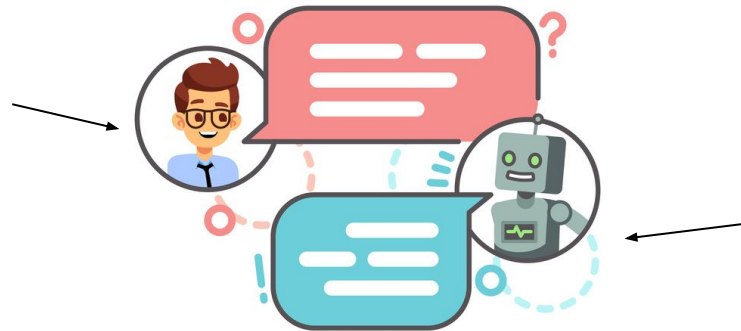


Data Collection

- **Wizard of Oz method:** Real-time dialog between two players (**user** & **system**)
- Take each question from *Spider* as user's dialog goal.

DB User: Crowd Workers

Interface System:
CS students with SQL skill



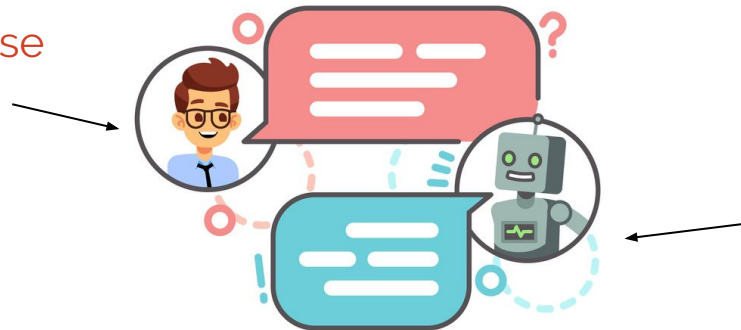
Data Collection

- **Wizard of Oz method:** Real-time dialog between two players (**user** & **system**)
- Take each question from *Spider* as user's dialog goal.

DB User: Crowd Workers

1. Explore DB
2. Query DB
3. Verify the response from system

Interface System:
CS students with SQL skill

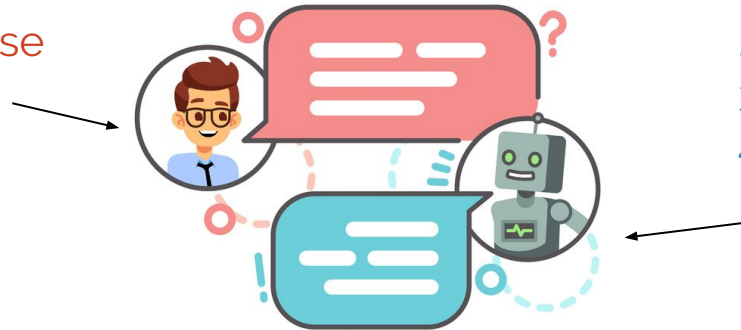


Data Collection

- **Wizard of Oz method:** Real-time dialog between two players (**user** & **system**)
- Take each question from *Spider* as user's dialog goal.

DB User: Crowd Workers

1. Explore DB
2. Query DB
3. Verify the response from system



Interface System:

CS students with SQL skill

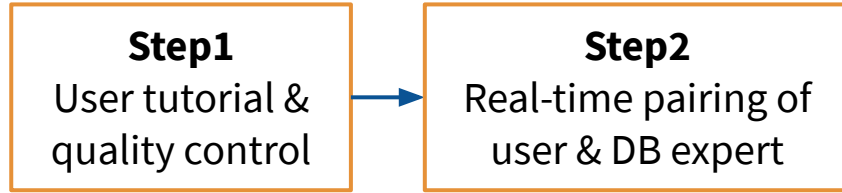
1. Dialog act management
2. SQL translation
3. Response generation
4. Inform of ambiguous/unanswerable questions

Data Collection (MTurk Process)

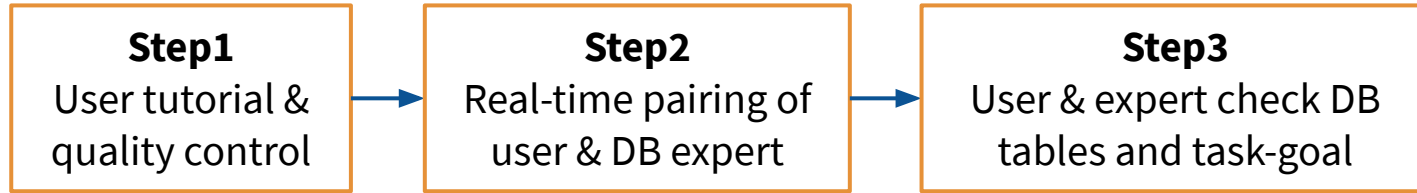
Step1

User tutorial &
quality control

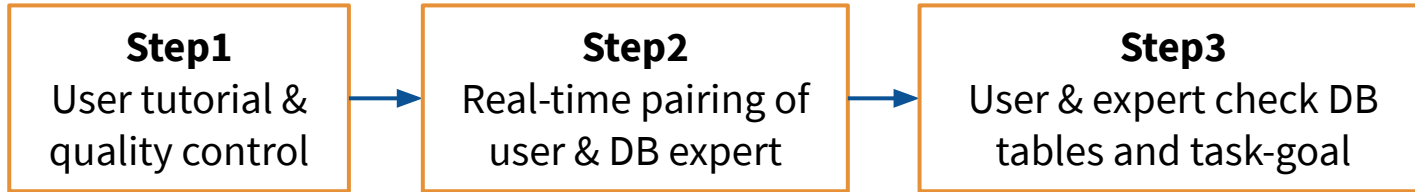
Data Collection (MTurk Process)



Data Collection (MTurk Process)



Data Collection (MTurk Process)



1. Read the tables below, which will be used by the assistant to answer your questions. Once complete, click "Next" to proceed

Table Name: Product_Suppliers

product id	supplier id	date supplied from	date supplied to	total amount purchased	total value purchased
4	3	2017-06-19 00:49:05	2018-03-24 19:29:18	89366.05	36014.6
8	4	2017-07-02 00:35:12	2018-03-26 07:30:49	26086.57	36274.56
3	3	2017-10-14 19:15:37	2018-03-24 02:29:44	16762.45	7273.74

Table Name: Order_Items

order item id	order id	product id
1	9	7
2	1	3
3	5	2

2. You are playing the role of a user who wants to know the answer to the question below. Remember:

- Ask at least 3 [related questions](#) about the data on the tables, you can refer to the question below.
- Good questions build up on previous questions. You can either break the given question down into small questions or ask other [related questions](#)
- You'll get \$ 0.5 bonus later if you follow the rules and ask more than 4 good [related questions!](#)

QUESTION: Return the ids of all products that were ordered more than three times or supplied more than 8000.

TASK ID: 3669, You are User

Hi, how can I help you?
Assistant

Results:

product_id
4
5
8

Enter Message

Information Request:
Return the ids of all products that were ordered more than three times or supplied more than 8000.

Final SQL:
SELECT product_id FROM Order_Items GROUP BY product_id HAVING count(*) > 3 UNION SELECT product_id FROM Product_Suppliers GROUP BY product_id HAVING sum(total_amount_purchased) > 8000

Database Name:
Department_store

Reference Table: Order_Items

Structure	Content
order_item_id	order_id
1	9
2	1
3	5
4	14
5	16
6	14
7	6
8	12
9	13
10	14

Reference Table: Product_Suppliers

Structure	Content				
product_id	supplier_id	date_supplied_from	date_supplied_to	total_amount_purchased	total_value_purchased
4	3	2017-06-19 00:49:05	2018-03-24 19:29:18	89366.05	36014.6
8	4	2017-07-02 00:35:12	2018-03-26 07:30:49	26086.57	36274.56
3	3	2017-10-14 19:15:37	2018-03-24 02:29:44	16762.45	7273.74
7	1	2017-08-03 00:58:42	2018-03-24 02:38:31	22330.08	8042.26
16	4	2017-12-08 09:14:09	2018-03-24 23:03:30	28318.21	29836.26
11	1	2017-10-01 19:46:53	2018-03-24 00:22:36	30489.74	67216.31
11	3	2017-07-13 16:03:24	2018-03-24 23:01:03	31862.59	76960.42

TASK ID: 3669, You are Assistant

Enter Message

Step 1: select USER labels:
inform_arg | infer_arg | | ambiguous | | affirm | | negate | | not_related | | cannot_understand | | cannot_answer | | greeting | | good_bye | | | thank_you | | | sleep | |

Step 2: select EXPERT labels:
confirm_arg | | clarify | | | reject | | | request_more | | | greeting | | | sorry | | | welcome | | | good_bye | | | sleep | |

Step 3: If the user's question can be answered by SQL, write/execute SQL query, and click "SQL confirm" button to show the result table to the user.

Step 4: write message and click send

Step 5: After the whole dialog ends, on the left panel: 1) grade the user's performance, 2) write some comments if there are some mistakes needed to be corrected during the future dialog review. 3) click button "DIALOG COMPLETED"

RESULTS

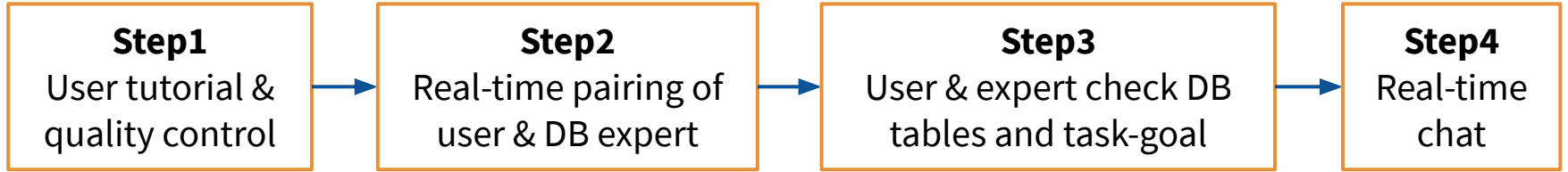
```

SELECT product_id FROM Order_Items GROUP BY product_id HAVING count(*) > 3 UNION
SELECT product_id FROM Product_Suppliers GROUP BY product_id HAVING
sum(total_amount_purchased) > 8000
  
```

Results (3 rows)

product_id
4
5
8

Data Collection (MTurk Process)



User side

1. Read the tables below, which will be used by the assistant to answer your questions. Once complete, click "Next" to proceed

Table Name: Product_Suppliers

product id	supplier id	date supplied from	date supplied to	total amount purchased	total value purchased
4	3	2017-06-19 00:49:05	2018-03-24 19:29:18	89366.05	36014.6
8	4	2017-07-02 00:35:12	2018-03-26 07:30:49	26086.57	36274.56
3	3	2017-10-14 19:15:37	2018-03-24 02:29:44	16762.45	7273.74

Table Name: Order_Items

order item id	order id	product id
1	9	7
2	1	3
3	5	2

2. You are playing the role of a user who wants to know the answer to the question below. Remember:

- Ask at least 3 [related questions](#) about the data on the tables, you can refer to the question below.
- Good questions build up on previous questions. You can either break the given question down into small questions or ask other [related questions](#)
- You'll get \$ 0.5 bonus later if you follow the rules and ask more than 4 good [related questions!](#)

QUESTION: Return the ids of all products that were ordered more than three times or supplied more than 8000.

TASK ID: 3669, You are User

Hi, how can I help you?
Assistant

Results:

product_id
4
5
8

Enter Message

System side

Information Request:
Return the ids of all products that were ordered more than three times or supplied more than 8000.

Final SQL:
SELECT product_id FROM Order_Items GROUP BY product_id HAVING count(*) > 3 UNION SELECT product_id FROM Product_Suppliers GROUP BY product_id HAVING sum(total_amount_purchased) > 8000

Database Name:
Department_store

Reference Table: Order_Items

order_item_id	order_id	product_id
1	9	7
2	1	3
3	5	2
4	14	10
5	15	4
6	14	13
7	6	13
8	12	8
9	13	13
10	14	13

Reference Table: Product_Suppliers

product_id	supplier_id	date_supplied_from	date_supplied_to	total_amount_purchased	total_value_purchased
4	3	2017-06-19 00:49:05	2018-03-24 19:29:18	89366.05	36014.6
8	4	2017-07-02 00:35:12	2018-03-26 07:30:49	26086.57	36274.56
3	3	2017-10-14 19:15:37	2018-03-24 02:29:44	16762.45	7273.74
7	1	2017-08-22 00:58:42	2018-03-24 02:29:44	22332.08	8042.76
10	4	2017-12-08 09:14:09	2018-03-24 23:03:30	35338.21	29836.26
11	1	2017-10-01 19:46:53	2018-03-24 02:29:44	30149.74	67216.31
13	3	2017-07-13 10:02:24	2018-03-24 23:01:03	31863.59	76992.42

TASK ID: 3669, You are Assistant

Enter Message

Step 1: select USER labels:
inform_sgd || infer_and || ambiguous || affirm || negate || not_related || not_understand || cannot_answer || greeting || good_bye || thank_you || drop ||

Step 2: select EXPERT labels:
confirm_and || clarify || reject || request_more || greeting || sorry || welcome || good_bye || drop ||

Step 3: If the user's question can be answered by SQL, write/associate SQL query and click "SQL confirm" button to show the result table to the user.

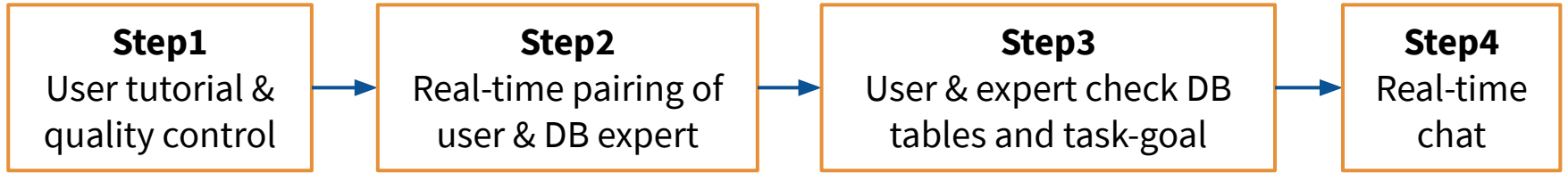
Step 4: write message and click send

Step 5: After the whole dialog ends, on the left panel: 1) grade the user's performance, 2) write some comments if there are some mistakes needed to be corrected during the future dialog reviews, 3) click button "DIALOG COMPLETED"

Results (3 rows)

product_id
4
5
8

Data Collection (MTurk Process)



User side

System side

1. Read the tables below, which will be used by the assistant to answer your questions. Once complete, click "Next" to proceed

Table Name: Product_Suppliers

product id	supplier id	date supplied from	date supplied to	total amount purchased	total value purchased
4	3	2017-06-19 00:49:05	2018-03-24 19:29:18	89366.05	36014.6
8	4	2017-07-02 00:35:12	2018-03-26 07:30:49	26086.57	36274.56
3	3	2017-10-14 19:15:37	2018-03-24 02:29:44	16762.45	7273.74

Table Name: Order_Items

order item id	order id	product id
1	9	7
2	1	3
3	5	2

2. You are playing the role of a user who wants to know the answer to the question below. Remember:

- Ask at least 3 [related questions](#) about the data on the tables, you can refer to the question below.
- Good questions build up on previous questions. You can either break the given question down into small questions or ask other [related questions](#)
- You'll get \$ 0.5 bonus later if you follow the rules and ask more than 4 good [related questions!](#)

QUESTION: Return the ids of all products that were ordered more than three times or supplied more than 8000.

TASK ID: 3669, You are User

Hi, how can I help you?
Assistant

Results:

product_id
4
5
8

Enter Message SEND

Information Request:
Return the ids of all products that were ordered more than three times or supplied more than 8000.

Final SQL:
SELECT product_id FROM Order_Items GROUP BY product_id HAVING count(*) > 3 UNION SELECT product_id FROM Product_Suppliers GROUP BY product_id HAVING sum(total_amount_purchased) > 8000

Database Name:
Department_store

Reference Table Order_Items

order_item_id	order_id	product_id
1	9	7
2	1	3
3	5	2
4	14	10
5	15	4
6	14	13
7	6	13
8	12	8
9	13	12
10	14	13

Reference Table Product_Suppliers

product_id	supplier_id	date_supplied_from	date_supplied_to	total_amount_purchased	total_value_purchased
4	3	2017-06-19 00:49:05	2018-03-24 19:29:18	89366.05	36014.6
8	4	2017-07-02 00:35:12	2018-03-26 07:30:49	26086.57	36274.56
3	3	2017-10-14 19:15:37	2018-03-24 02:29:44	16762.45	7273.74
7	1	2017-08-22 09:14:05	2018-03-24 02:29:44	22320.08	8042.76
10	4	2017-12-08 09:14:05	2018-03-24 23:03:30	35338.21	29836.26
11	1	2017-01-10 19:40:53	2018-03-24 02:29:44	30149.74	67216.31
12	3	2017-07-13 10:02:24	2018-03-24 23:01:03	31863.59	76992.42

TASK ID: 3669, You are Assistant

Enter Message SEND

Step 1: select USER labels:
inform_sgd || infer_and || ambiguous || affirm || negate || not_related || not_understand || cannot_answer || greeting || good_bye || thank_you || drop ||

Step 2: select EXPERT labels:
confirm_and || clarify || reject || request_more || greeting || sorry || welcome || good_bye || drop ||

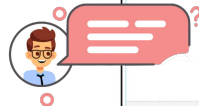
Step 3: If the user's question can be answered by SQL, write/associate SQL query, and click "SQL confirm" button to show the result table to the user.

Step 4: write message and click send

Step 5: After the whole dialog ends, on the left panel: 1) grade the user's performance, 2) write some comments if there are some mistakes needed to be corrected during the future dialog reviews. 3) click button "DIALOG COMPLETED"

Results (3 rows)

product_id
4
5
8



Dataset Statistics

CoSQL v.s. context-dependent NL2SQL

	ATIS	CoSQL
# Q sequence	1658	3,007
# user questions	11,653	15,598*
# databases	1	200
# tables	27	1020
Avg. Q len	10.2	11.2
Vocab	1582	9,585
Avg. # Q turns	7.0	5.2
Unanswerable Q	✗	✓
User intent	✗	✓
System response	✗	✓

Dataset Statistics

CoSQL v.s. context-dependent NL2SQL ⇒ **More practically useful & interpretable**

	ATIS	CoSQL
# Q sequence	1658	3,007
# user questions	11,653	15,598*
# databases	1	200
# tables	27	1020
Avg. Q len	10.2	11.2
Vocab	1582	9,585
Avg. # Q turns	7.0	5.2
Unanswerable Q	✗	✓
User intent	✗	✓
System response	✗	✓

Dataset Statistics

CoSQL v.s. task-oriented dialogue datasets

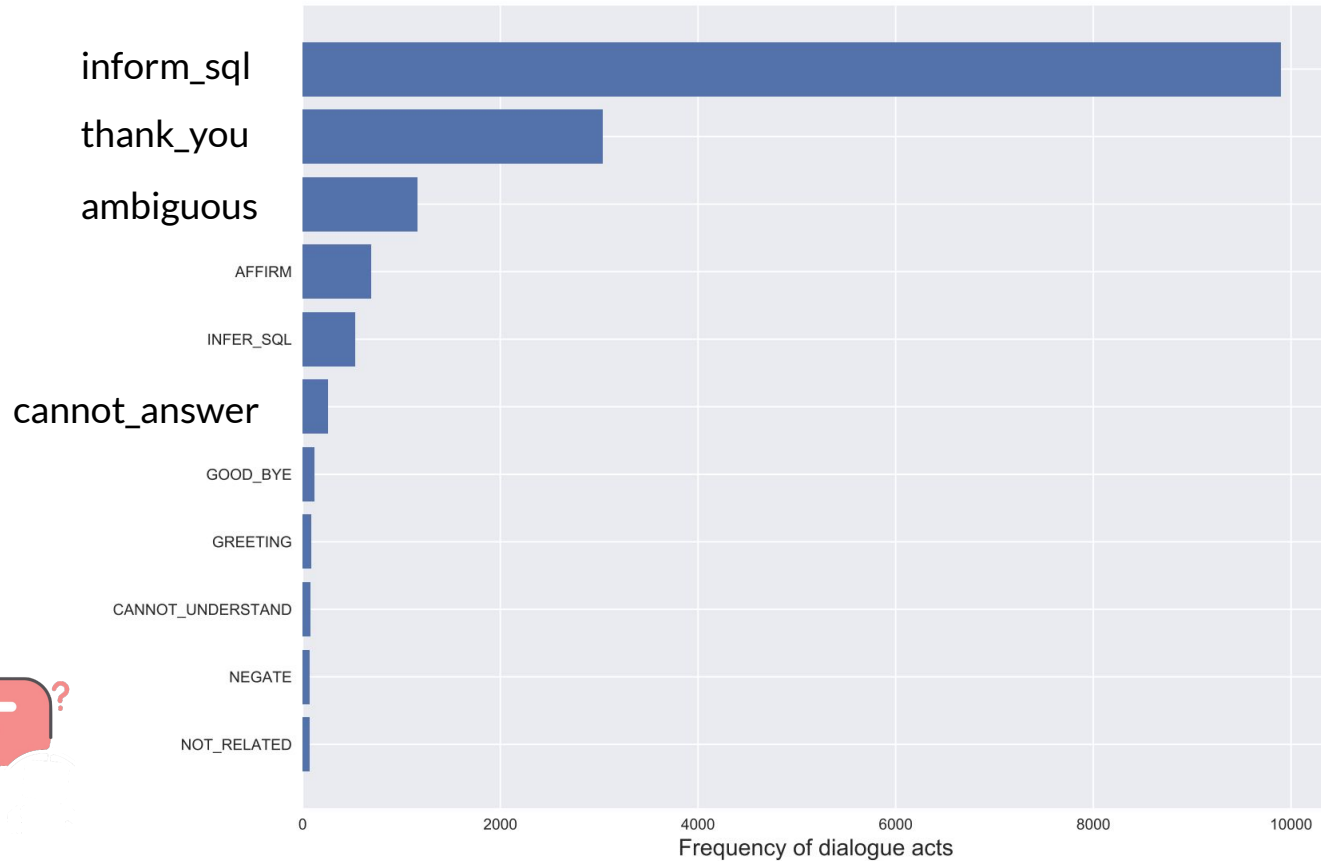
	DSTC2	WOZ 2.0	KVRET	MultiWOZ	CoSQL
# dialogs	1,612	600	2,425	8,438	2,164
Total # turns	23,354	4,472	12,732	115,424	22,422
Total # tokens	199,431	50,264	102,077	1,520,970	22,8197
Avg. # turns/dialog	14.49	7.45	5.25	13.68	10.36
Avg. # tokens/turn	8.54	11.24	8.02	13.18	11.34
Total # unique tokens	986	2,142	2,842	24,071	7,502
# databases	1	1	1	7	140
# Slots #	8	4	13	25	3,696
# Values #	212	99	1,363	4,510	>1,000,000

Dataset Statistics

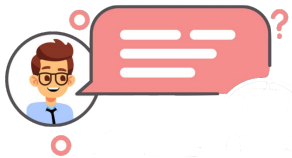
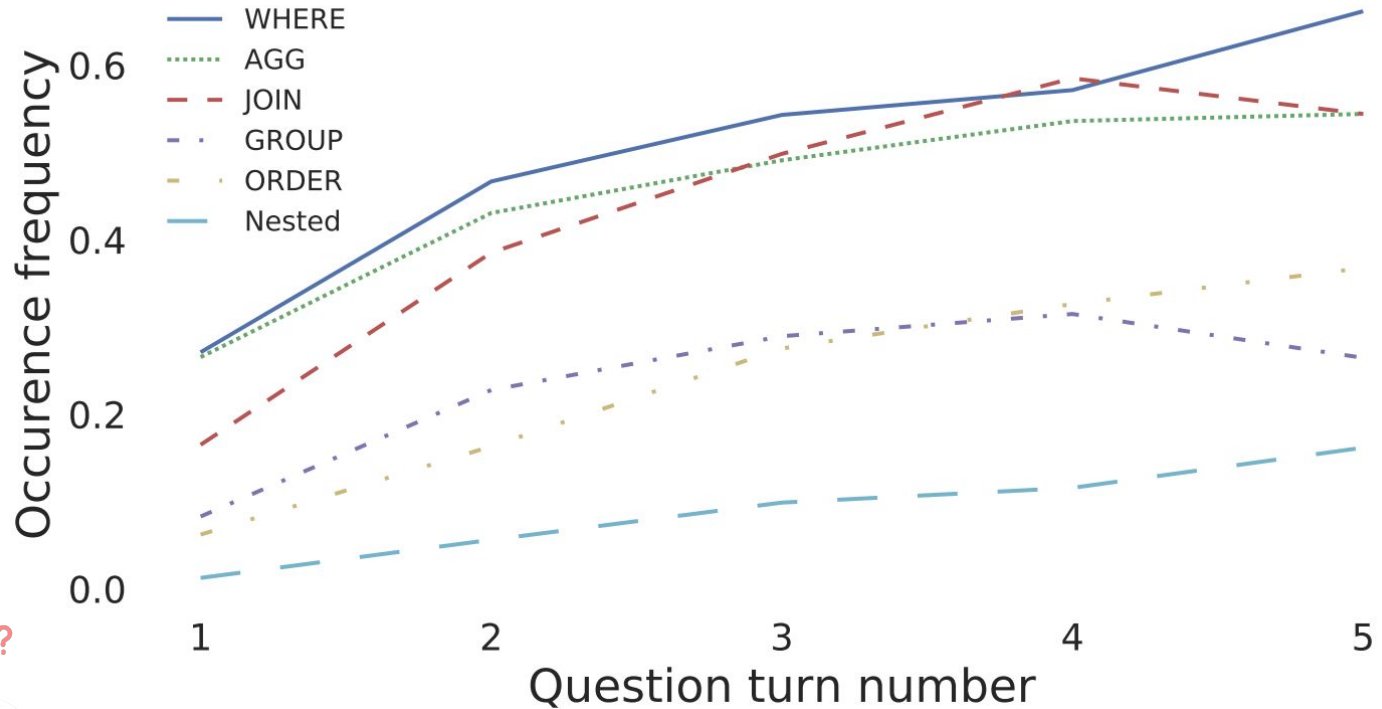
CoSQL v.s. task-oriented dialogue datasets ⇒ Semantically diverse

	DSTC2	WOZ 2.0	KVRET	MultiWOZ	CoSQL
# dialogs	1,612	600	2,425	8,438	2,164
Total # turns	23,354	4,472	12,732	115,424	22,422
Total # tokens	199,431	50,264	102,077	1,520,970	22,8197
Avg. # turns/dialog	14.49	7.45	5.25	13.68	10.36
Avg. # tokens/turn	8.54	11.24	8.02	13.18	11.34
Total # unique tokens	986	2,142	2,842	24,071	7,502
# databases	1	1	1	7	140
# Slots #	8	4	13	25	3,696
# Values #	212	99	1,363	4,510	>1,000,000

Dialog act distribution

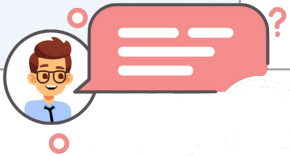


Semantic change by turns



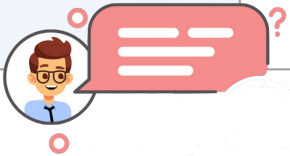
User follow-up question characteristics

	Coreference	Change constraint	Different attribute of same topic	Same attribute for different topic	Question about the answer
Previous Q	How many <u>female students</u> are in the class?				
Follow up Q					



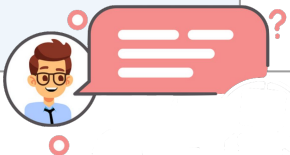
User follow-up question characteristics

	Coreference	Change constraint	Different attribute of same topic	Same attribute for different topic	Question about the answer
Previous Q	How many <u>female students</u> are in the class?				
Follow up Q	What are their names?				



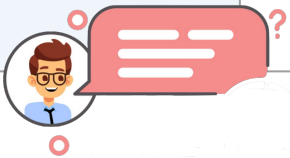
User follow-up question characteristics

	Coreference	Change constraint	Different attribute of same topic	Same attribute for different topic	Question about the answer
Previous Q	How many <u>female students</u> are in the class?	Show me courses taught in the winter			
Follow up Q	What are their names?	How about in the summer ?			



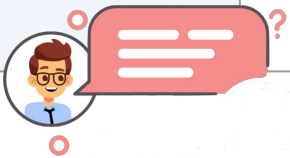
User follow-up question characteristics

	Coreference	Change constraint	Different attribute of same topic	Same attribute for different topic	Question about the answer
Previous Q	How many <u>female students</u> are in the class?	Show me courses taught in the winter	Where is the location of the conference ?		
Follow up Q	What are their names?	How about in the summer ?	What is the time ?		



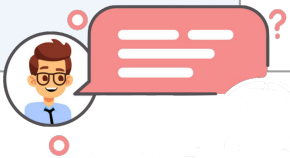
User follow-up question characteristics

	Coreference	Change constraint	Different attribute of same topic	Same attribute for different topic	Question about the answer
Previous Q	How many <u>female students</u> are in the class?	Show me courses taught in the winter	Where is the location of the conference ?	What is the price of a bagel ?	
Follow up Q	What are their names?	How about in the summer ?	What is the time ?	And a swiss roll ?	



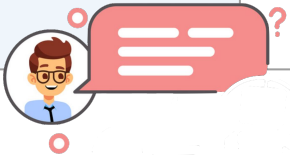
User follow-up question characteristics

	Coreference	Change constraint	Different attribute of same topic	Same attribute for different topic	Question about the answer
Previous Q	How many <u>female students</u> are in the class?	Show me courses taught in the winter	Where is the location of the conference ?	What is the price of a bagel ?	What is the highest rated college in CT? Answer: Yale
Follow up Q	What are their names?	How about in the summer ?	What is the time ?	And a swiss roll ?	

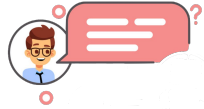


User follow-up question characteristics

	Coreference	Change constraint	Different attribute of same topic	Same attribute for different topic	Question about the answer
Previous Q	How many <u>female students</u> are in the class?	Show me courses taught in the winter	Where is the location of the conference ?	What is the price of a bagel ?	What is the highest rated college in CT? Answer: Yale
Follow up Q	What are their names?	How about in the summer ?	What is the time ?	And a swiss roll ?	How many departments does Yale have?



Tasks & Experiments

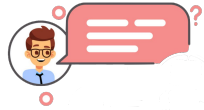


Context-dependent user question



Dialog act type

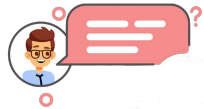
Tasks & Experiments



Context-dependent user question



Dialog act type

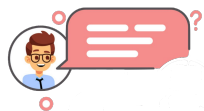


Context-dependent user question



SQL query

Tasks & Experiments

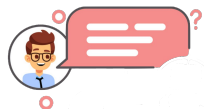


Context-dependent user question



Dialog act type

Dialog act management

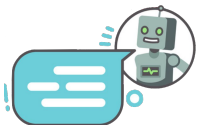


Context-dependent user question



SQL query

Dialog SQL state tracking



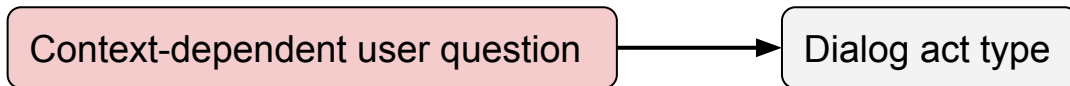
SQL query & execution result



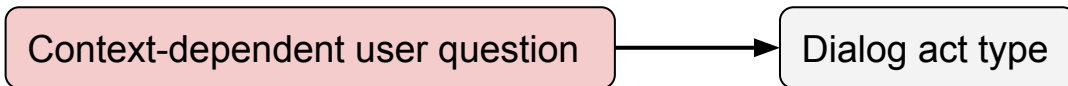
System response (NL description)

System response generation

Task 1: User Dialogue Act Prediction



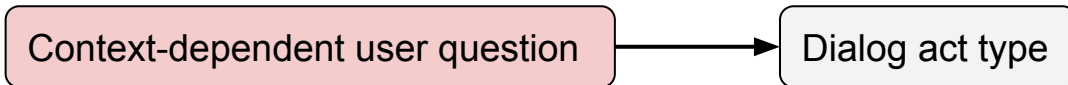
Task 1: User Dialogue Act Prediction



Models

- Majority -- always predict "INFORM_SQL"
- TBCNN [Mouet al., 2016]

Task 1: User Dialogue Act Prediction



Models

- Majority -- always predict “INFORM_SQL”
- TBCNN [Mouet al., 2016]

Model	Dev	Test
Majority	63.3	62.8
TBCNN-pair	84.2	83.9

Accuracy (%)

Task 1: User Dialogue Act Prediction

Context-dependent user question

Dialog act type

Models

- Majority -- always predict “INFORM_SQL”
- TBCNN [Mouet al., 2016]

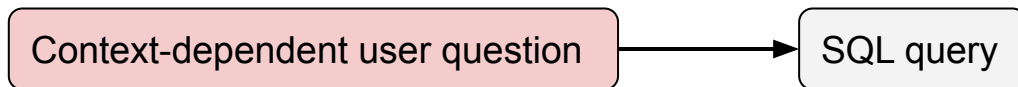
Model	Dev	Test
Majority	63.3	62.8
TBCNN-pair	84.2	83.9

Accuracy (%)

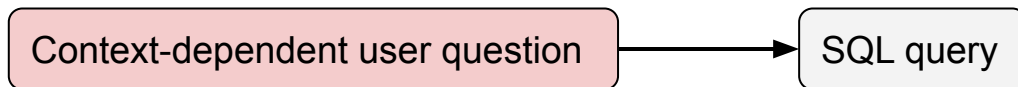
✓ common types
 (“inform_sql”, “thank_you”,
 “greetings”)

✗ other types
 (“ambiguous”,
 “cannot_answer”, etc.)

Task 2: SQL state tracking (context-dependent NL2SQL)



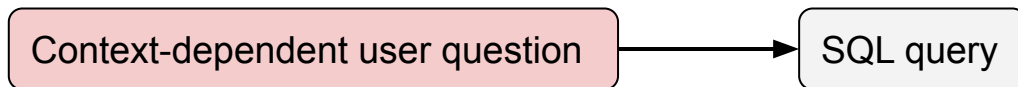
Task 2: SQL state tracking (context-dependent NL2SQL)



Models

- Context-dependent Seq2Seq [Suhr et al., 2018]
- Context-dependent SyntaxSQLNet [Yu et al., 2019]

Task 2: SQL state tracking (context-dependent NL2SQL)

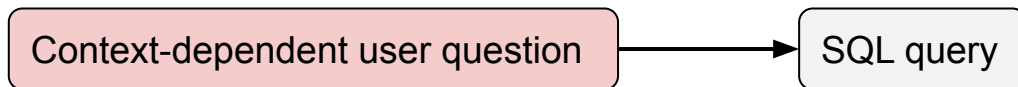


Models

- Context-dependent Seq2Seq [Suhr et al., 2018]
- Context-dependent SyntaxSQLNet [Yu et al., 2019]

Model	Question Match	
	Dev	Test
CD-Seq2Seq	13.8	13.9
SyntaxSQL-con	15.1	14.1

Task 2: SQL state tracking (context-dependent NL2SQL)

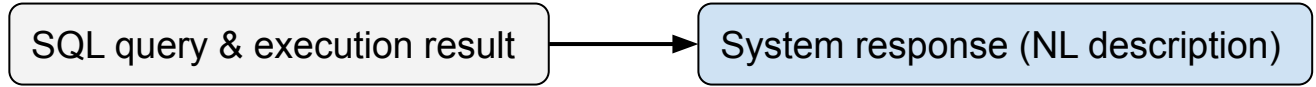


Models

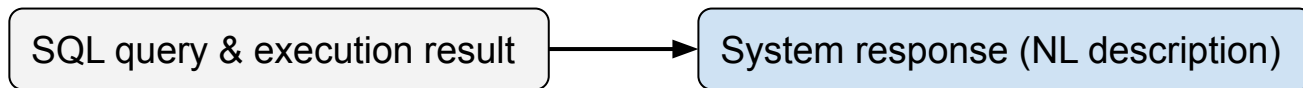
- Context-dependent Seq2Seq [Suhr et al., 2018]
- Context-dependent SyntaxSQLNet [Yu et al., 2019]

Model	Question Match		On Spider (single turn):
	Dev	Test	
CD-Seq2Seq	13.8	13.9	
SyntaxSQL-con	15.1	14.1	← 21%

Task 3: System Response Generation



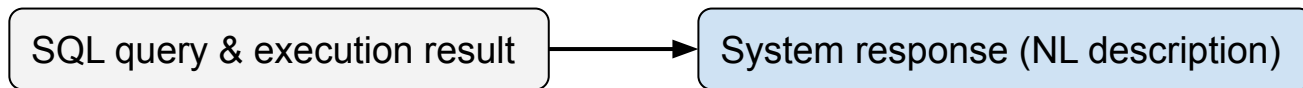
Task 3: System Response Generation



Models

- Template -- prepared from (SQL, response) pairs in training data
- Seq2seq
- Pointer-generator [See et al., 2017]

Task 3: System Response Generation

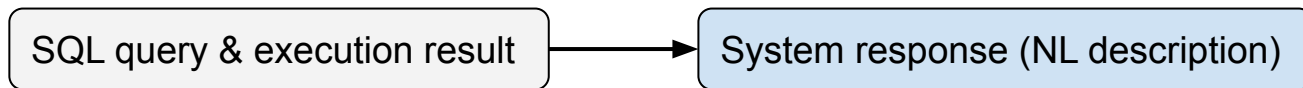


Models

- Template -- prepared from (SQL, response) pairs in training data
- Seq2seq
- Pointer-generator [See et al., 2017]

Model	BLEU		LCR (%)	Grammar
	Dev	Test	Test	Test
Template	9.5	9.3	41.0	4.0
Seq2Seq	15.3	14.1	27.0	3.5
Pointer-generator	16.4	15.1	35.0	3.6

Task 3: System Response Generation



Models

- Template -- prepared from (SQL, response) pairs in training data
- Seq2seq
- Pointer-generator [See et al., 2017]

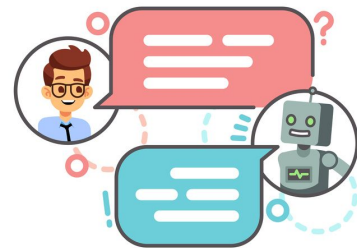
Model	BLEU		LCR (%)	Grammar
	Dev	Test	Test	Test
Template	9.5	9.3	41.0	4.0
Seq2Seq	15.3	14.1	27.0	3.5
Pointer-generator	16.4	15.1	35.0	3.6

Semantic correctness

Human evaluation

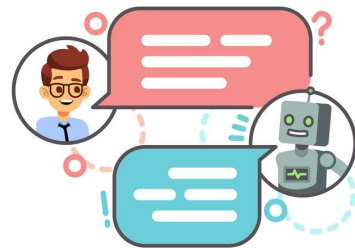
Summary

- CoSQL, first truly **conversational** NL2SQL corpus
 - Include ambiguous/unanswerable questions real users ask
 - Include system response to improve interpretability & user experience



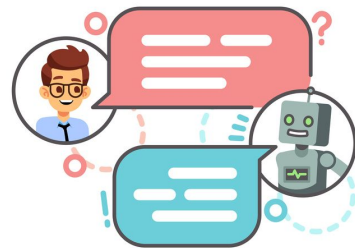
Summary

- CoSQL, first truly **conversational** NL2SQL corpus
 - Include ambiguous/unanswerable questions real users ask
 - Include system response to improve interpretability & user experience
- More diversity and complexity in semantics and discourse (e.g. ambiguous questions, multi-domain), compared to related datasets



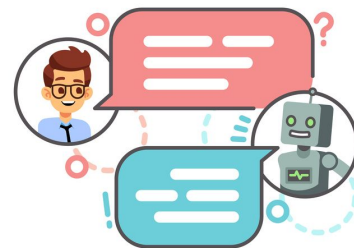
Summary

- CoSQL, first truly **conversational** NL2SQL corpus
 - Include ambiguous/unanswerable questions real users ask
 - Include system response to improve interpretability & user experience
- More diversity and complexity in semantics and discourse (e.g. ambiguous questions, multi-domain), compared to related datasets
- Experiments on the three tasks show a large room for future research



Summary

- CoSQL, first truly **conversational** NL2SQL corpus
 - Include ambiguous/unanswerable questions real users ask
 - Include system response to improve interpretability & user experience
- More diversity and complexity in semantics and discourse (e.g. ambiguous questions, multi-domain), compared to related datasets
- Experiments on the three tasks show a large room for future research
- CoSQL project page: <https://yale-lily.github.io/cosql>



Thank you!

Michihiro Yasunaga