Context-Dependent Language Understanding

Alane Suhr
Cornell University

Types of Context

Context:

Previous instruction:

Go to the tree on the right

Previous interpretation:

`λa.move(a) ∧ λx.tree(x) ∧ right-of(x, rock) ∧ to(a, x)`

Current instruction:

Go to the other tree

`λa.move(a) ∧ λx.tree(x) ∧ ¬right-of(x, rock) ∧ to(a, x)`

Types of Context

Context:

- Previous instructions

Previous instruction:
Go to the tree on the right

Previous interpretation:
λa.move(a) ∧ λx.tree(x) ∧ right-of(x, rock) ∧ to(a, x)

Current instruction:
Go to the other tree

λa.move(a) ∧ λx.tree(x) ∧ ¬right-of(x, rock) ∧ to(a, x)

Types of Context

Context:

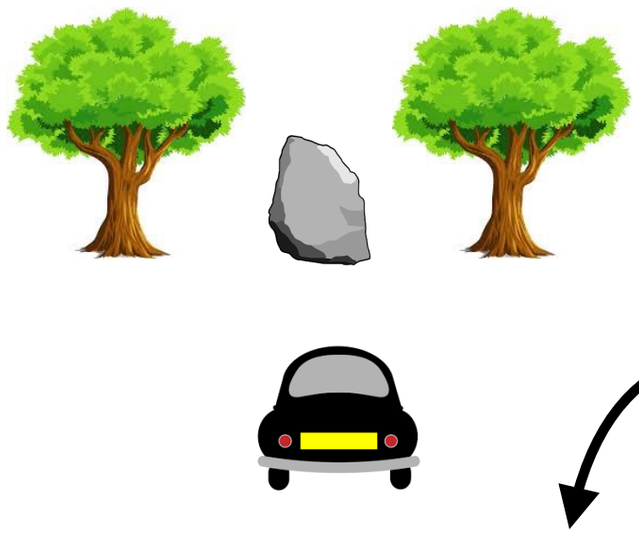
- Previous instructions
- Previous interpretations of instructions
- Current world state
- Previous world states

Previous instruction:

Go to the tree on the right

Previous interpretation:

λa.move(a) ∧ λx.tree(x) ∧ right-of(x, rock) ∧ to(a, x)



Current instruction:

Go to the other tree

λa.move(a) ∧ λx.tree(x) ∧ ¬right-of(x, rock) ∧ to(a, x)

Types of Context

Context:

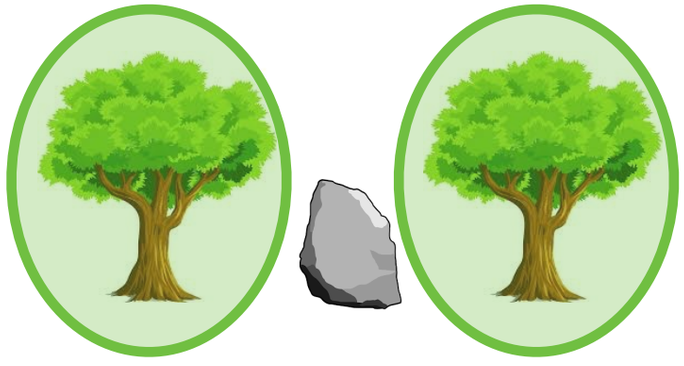
- Previous instructions
- Previous interpretations of instructions
- Current world state

Previous instruction:

Go to the tree on the right

Previous interpretation:

λa.move(a) ∧ λx.tree(x) ∧ right-of(x, rock) ∧ to(a, x)



Current instruction:

Go to the other tree

λa.move(a) ∧ λx.tree(x) ∧ ¬right-of(x, rock) ∧ to(a, x)

Types of Context

Context:

- Previous instructions
- Previous interpretations of instructions
- Current world state

Previous instruction:

Go to the tree on the right

Previous interpretation:

`λa.move(a) ∧ λx.tree(x) ∧ right-of(x, rock) ∧ to(a, x)`

Current instruction:

Go to the other tree

`λa.move(a) ∧ λx.tree(x) ∧ ¬right-of(x, rock) ∧ to(a, x)`

Types of Context

Context:

- Previous instructions
- Previous interpretations of instructions
- Current world state
- Previous actions taken by user or agent
- Previous world states

Previous instruction:

Go to the tree on the right

Previous interpretation:

`λa.move(a) ∧ λx.tree(x) ∧ right-of(x, rock) ∧ to(a, x)`

Current instruction:

Go to the other tree

`λa.move(a) ∧ λx.tree(x) ∧ ¬right-of(x, rock) ∧ to(a, x)`

Outline

1. **Case study:** natural language interface to a database
    A. Previous utterances
    B. Previous interpretations of utterances
2. **Situated interaction context**
3. **Open questions**

Case Study: Natural Language Interface to a Database

User *Show me flights from Seattle to Boston next Monday*

[Hemphill et al. 1990, Dahl et al. 1994]

Case Study: Natural Language Interface to a Database

User | *Show me* *flights* *from Seattle* *to Boston* *next Monday*

[Hemphill et al. 1990, Dahl et al. 1994]

Case Study: Natural Language Interface to a Database

User *Show me flights from Seattle to Boston next Monday*

SQL Query

```
(SELECT DISTINCT flight.flight_id FROM flight WHERE
(flight.from_airport IN (SELECT airport_service.airport_code
FROM airport_service WHERE airport_service.city_code IN
(SELECT city.city_code FROM city WHERE city.city_name =
'SEATTLE'))) AND (flight.to_airport IN (SELECT
airport_service.airport_code FROM airport_service WHERE
airport_service.city_code IN (SELECT city.city_code FROM
city WHERE city.city_name = 'BOSTON'))) AND
(flight.flight_days IN (SELECT days.days_code FROM days
WHERE days.day_name IN (SELECT date_day.day_name FROM
date_day WHERE date_day.year = 1993 AND
date_day.month_number = 2 AND date_day.day_number = 8)));
```

Case Study: Natural Language Interface to a Database

User *Show me flights from Seattle to Boston next Monday*

Result `Found 31 Flights:` 

[Hemphill et al. 1990, Dahl et al. 1994]

Case Study: Natural Language Interface to a Database

User *On American Airlines*

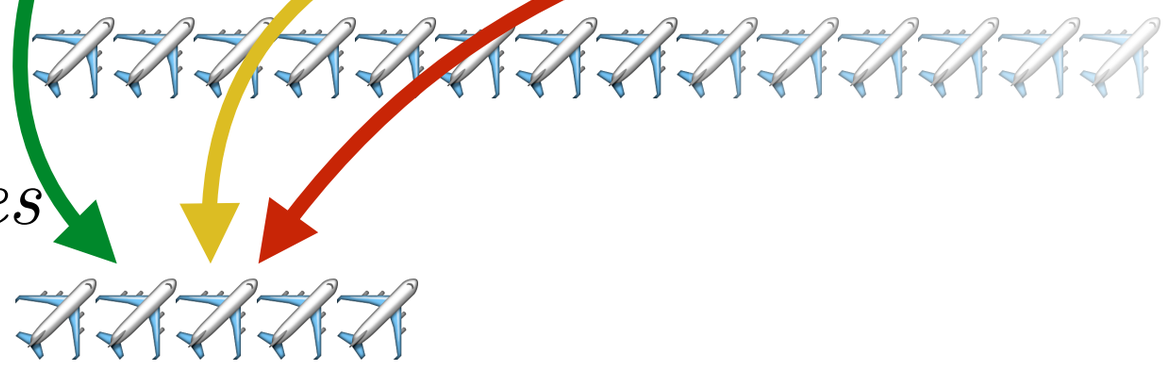
Result `Found 2764 Flights:`

[Hemphill et al. 1990, Dahl et al. 1994]

Case Study: Natural Language Interface to a Database

User *Show me flights* *from Seattle* *to Boston* *next Monday*

Result `Found 31 Flights:`

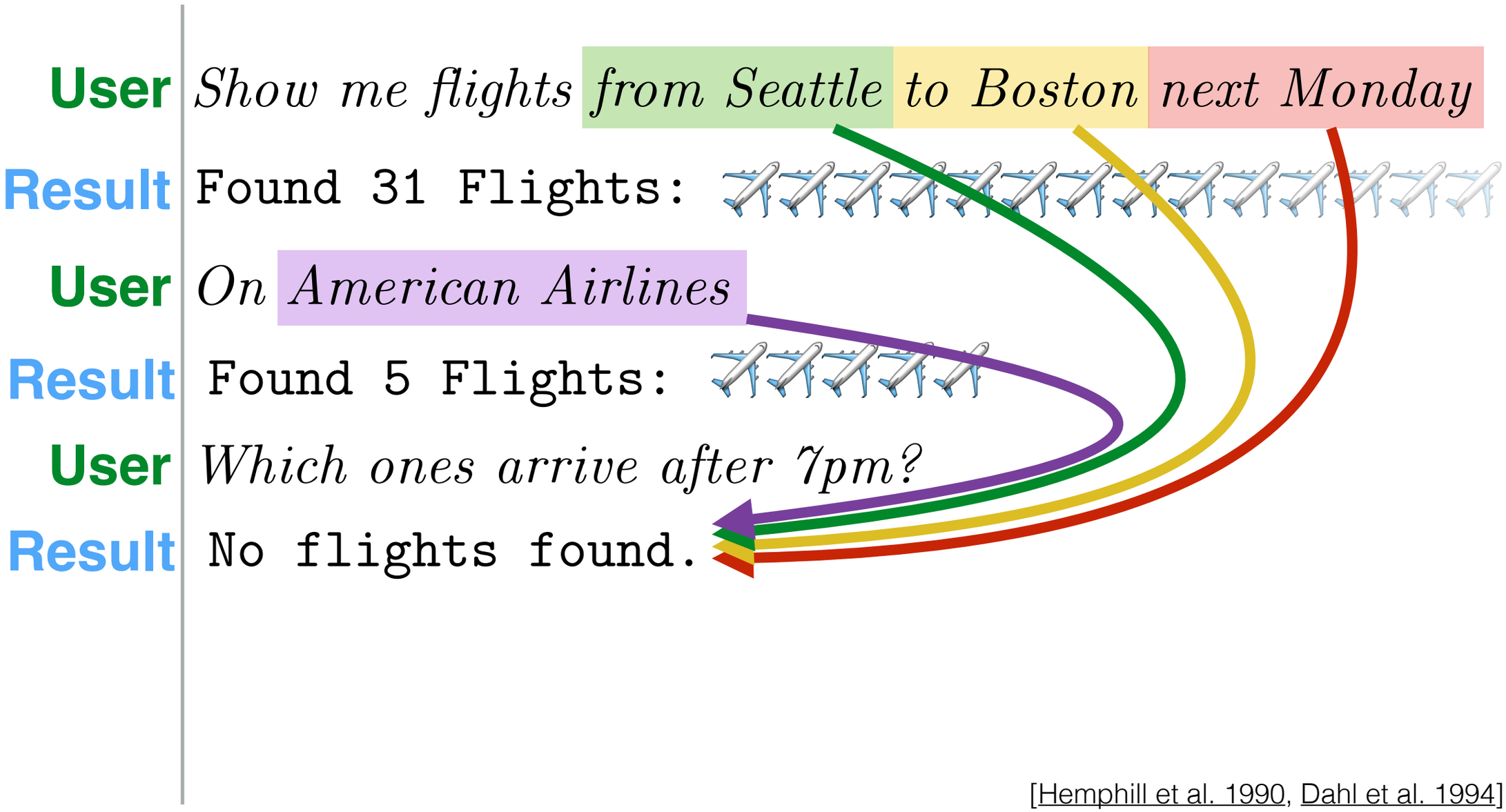


User *On American Airlines*

Result `Found 5 Flights:`

[Hemphill et al. 1990, Dahl et al. 1994]

Case Study: Natural Language Interface to a Database



[Hemphill et al. 1990, Dahl et al. 1994]

Case Study: Natural Language Interface to a Database

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:`

[Hemphill et al. 1990, Dahl et al. 1994]

ATIS

- Flight information, 27 tables, 162K entries
- Small corpus: <2000 interactions
- Complex queries: average 102.9 tokens each; 93% reference >3 tables
- **Long interactions: average 7 turns; maximum: 64**

[Hemphill et al. 1990, Dahl et al. 1994]

Previous Approaches

- **Model:** statistical decision tree
- **Representation:** semantic frames
- **Two-step process:**
  1. Map request to context-independent frame
  2. Use context to decide how to use past constraints
- **Context:** previous decisions on using constraints

[Miller et al. 1996]

Previous Approaches

- **Model:** weighted linear combinatory categorial grammar
- **Representation:** lambda calculus
- **Two-step process:**
  1. Map request to context-independent expression
  2. Use context to map to context-dependent expression
- **Context:** previously generated expressions

[Zettlemoyer and Collins 2009]

Suhr et al. 2018

- As an interaction progresses, the meaning of an utterance becomes highly dependent on the meaning of previous utterances
- **Interaction history:** previously generated queries and previous requests

Challenges

- How do we make use of **previous requests**?
- How do we make use of **previously generated queries**?

[Suhr et al. 2018]

Context from Previous Requests

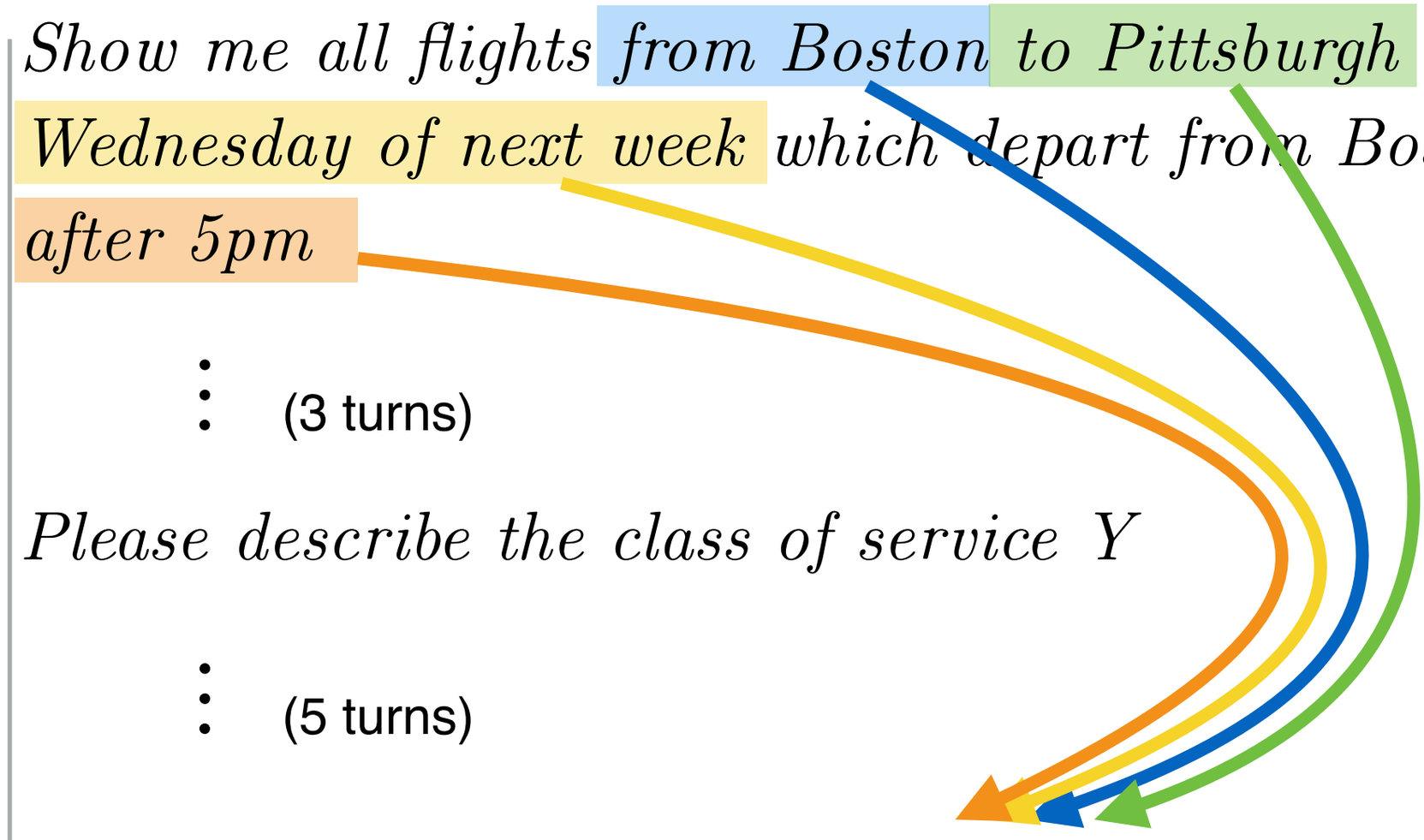
User *Show me all flights from Boston to Pittsburgh on Wednesday of next week which depart from Boston after 5pm*

⋮ (3 turns)

User *Please describe the class of service Y*

⋮ (5 turns)

User *Show the cost of tickets on flight US 345*



[Suhr et al. 2018]

Context from Previous Requests

- Relevant but elided information mentioned multiple turns before
- User may change focus during interaction
- **Solution:** implicit mechanism for carrying information from beginning to end of interaction

[Suhr et al. 2018]

Context from Previous Queries

User *Show me flights from Seattle to Boston next Monday*

SQL Query

```
(SELECT DISTINCT flight.flight_id FROM flight WHERE
(flight.from_airport IN (SELECT airport_service.airport_code
FROM airport_service WHERE airport_service.city_code IN
(SELECT city.city_code FROM city WHERE city.city_name =
'SEATTLE')) AND (flight.to_airport IN (SELECT
airport_service.airport_code FROM airport_service WHERE
airport_service.city_code IN (SELECT city.city_code FROM
city WHERE city.city_name = 'BOSTON'))) AND
(flight.flight_days IN (SELECT days.days_code FROM days
WHERE days.day_name IN (SELECT date_day.day_name FROM
date_day WHERE date_day.year = 1993 AND
date_day.month_number = 2 AND date_day.day_number = 8))));
```

Context from Previous Queries

User *On American Airlines*

SQL Query

```
(SELECT DISTINCT flight.flight_id FROM flight WHERE
(flight.airline_code = 'AA') AND (flight.from_airport IN
(SELECT airport_service.airport_code FROM airport_service
WHERE airport_service.city_code IN (SELECT city.city_code
FROM city WHERE city.city_name = ‘SEATTLE'))) AND
(flight.to_airport IN (SELECT airport_service.airport_code
FROM airport_service WHERE airport_service.city_code IN
(SELECT city.city_code FROM city WHERE city.city_name =
'BOSTON'))) AND (flight.flight_days IN (SELECT days.days_code
FROM days WHERE days.day_name IN (SELECT date_day.day_name
FROM date_day WHERE date_day.year = 1993 AND
date_day.month_number = 2 AND date_day.day_number = 8)));
```

Context from Previous Queries

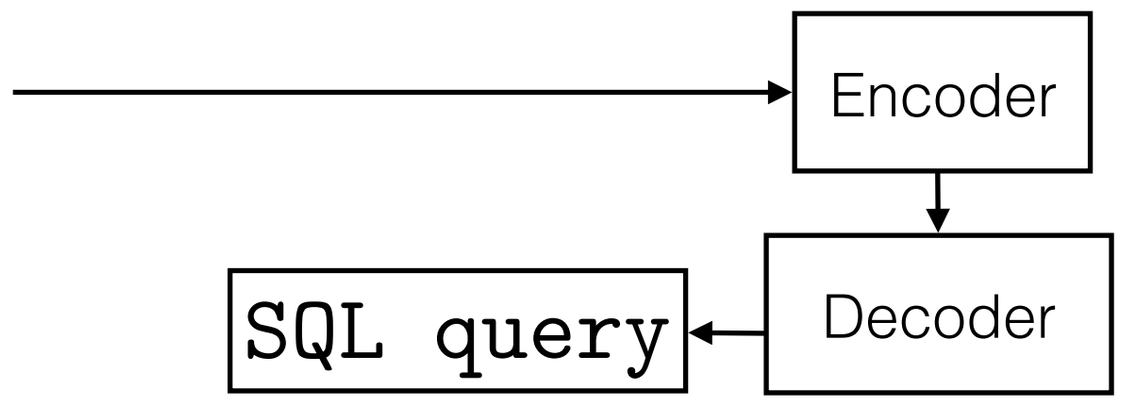
- Segments corresponding to earlier mentioned constraints and entities appear in later queries

- **Solution:** explicit mechanism for composing later SQL queries from segments of previous ones

[Suhr et al. 2018]

Semantic Parsing as Sequence-to-Sequence

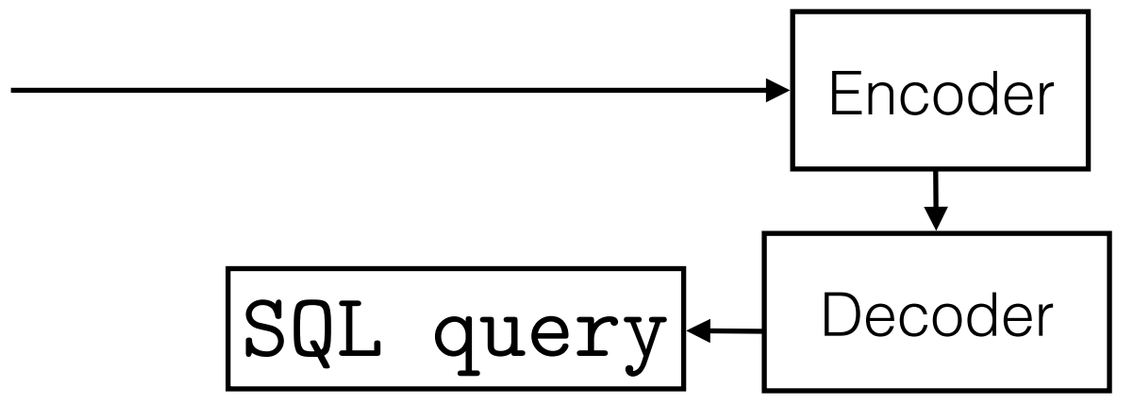
Show me flights from Seattle to Boston next Monday



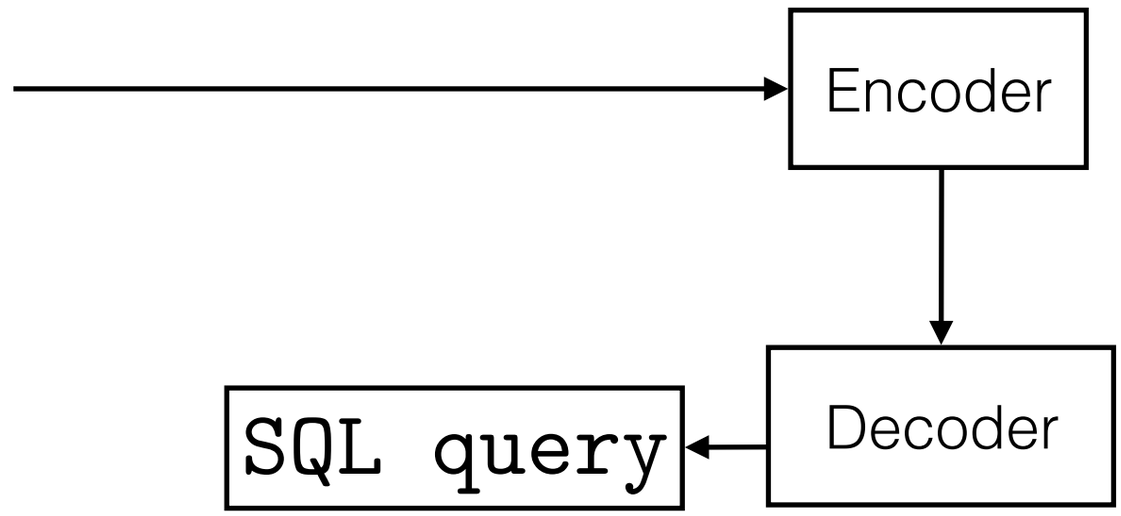
[Suhr et al. 2018]

Semantic Parsing as Sequence-to-Sequence

Show me flights from Seattle to Boston next Monday

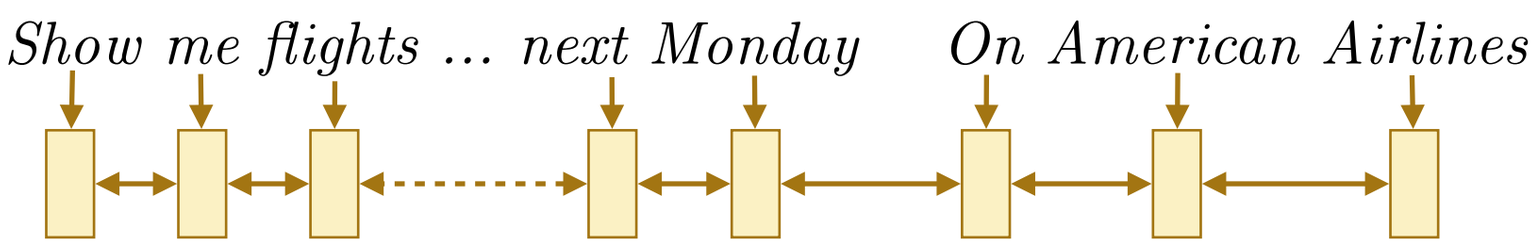


On American Airlines



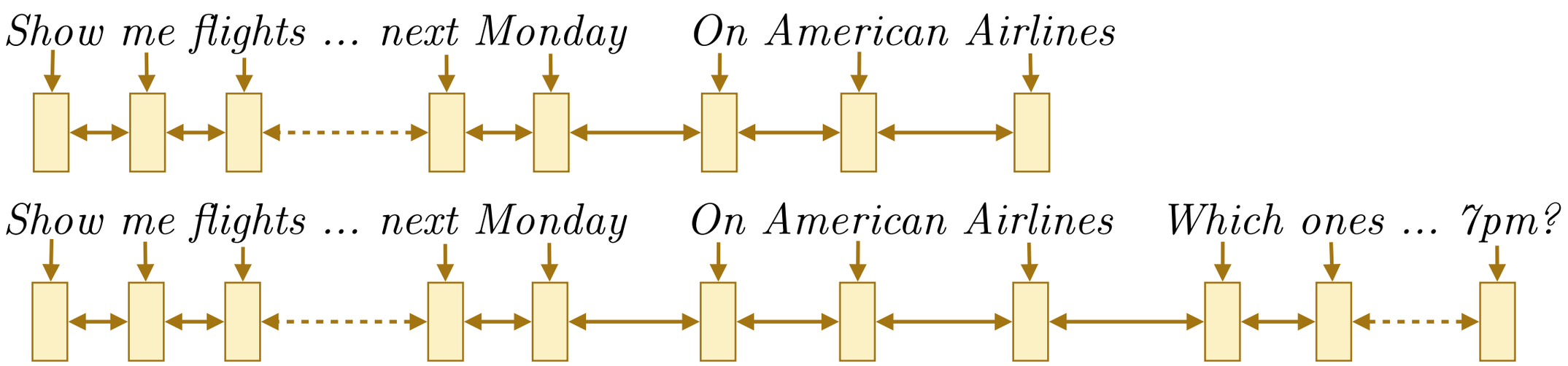
[Suhr et al. 2018]

Incorporating Previous Requests



Simple approach: encode interaction history with a bidirectional RNN

Incorporating Previous Requests



Simple approach: encode interaction history with a bidirectional RNN

Downside: need to re-run bidirectional encoder with each new utterance

Incorporating Previous Requests

Downside: need to re-run bidirectional encoder with each new utterance

Imposes tradeoff between efficiency and access to interaction history

n, number of previous concatenated utterances

Incorporating Previous Requests

Downside: need to re-run bidirectional encoder with each new utterance

Imposes tradeoff between efficiency and access to interaction history

$n = 0$

Previous constraints not available to decoder

n, number of previous concatenated utterances

Incorporating Previous Requests

Downside: need to re-run bidirectional encoder with each new utterance

Imposes tradeoff between efficiency and access to interaction history

$n = 0$

Previous constraints not available to decoder

$n = \infty$

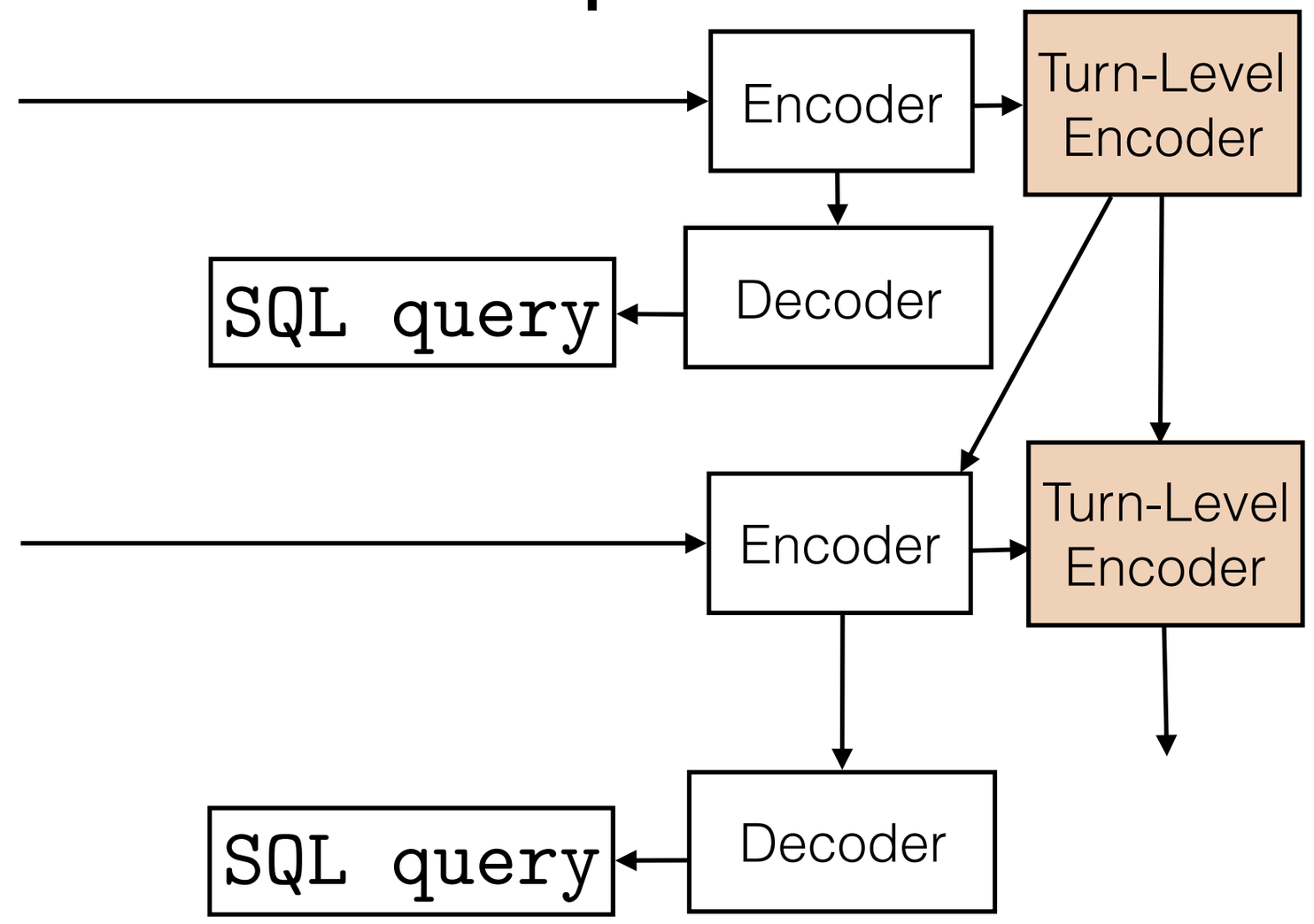
Inefficient to encode — must re-encode all utterances

n, number of previous concatenated utterances

Incorporating Previous Requests

Show me flights from Seattle to Boston next Monday

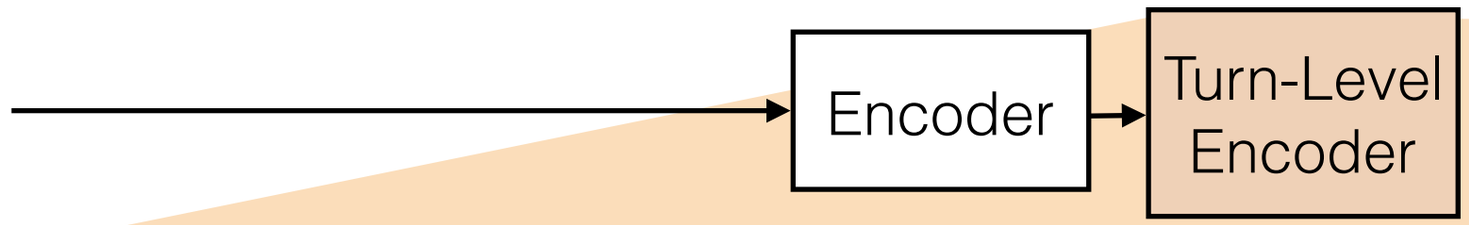
On American Airlines



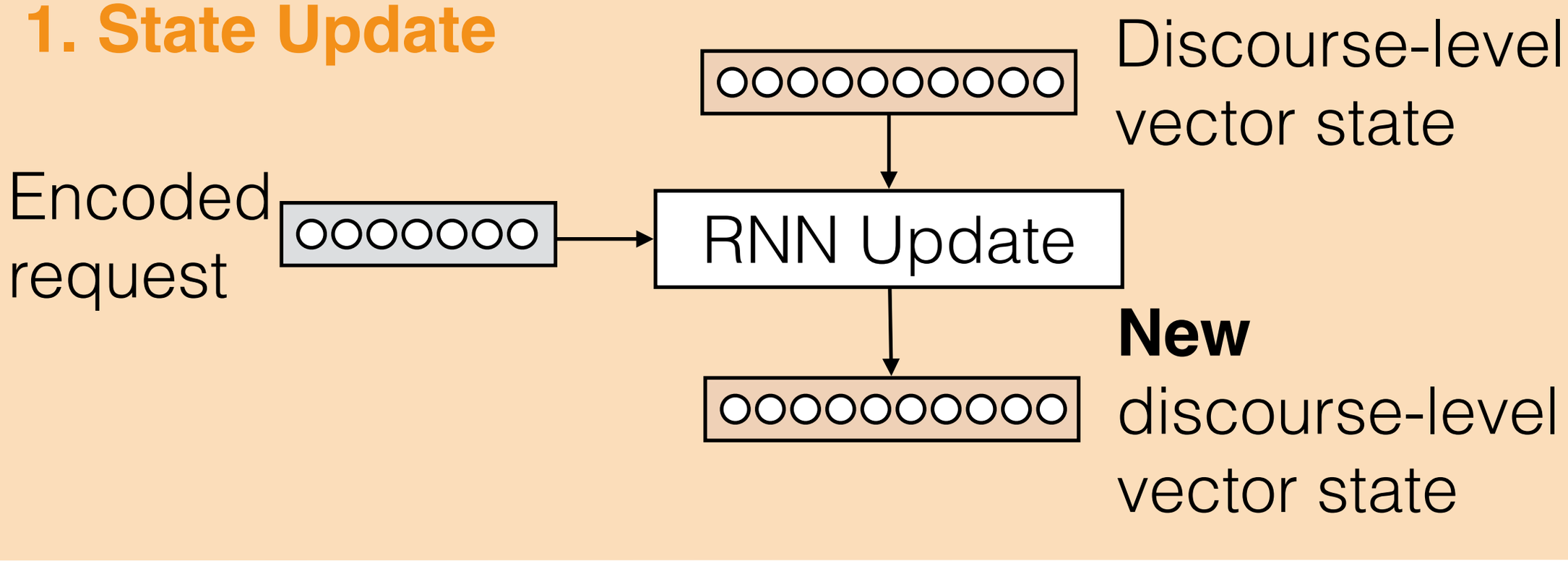
[Suhr et al. 2018]

Turn-level Encoder

Show me flights from Seattle to Boston next Monday



1. State Update

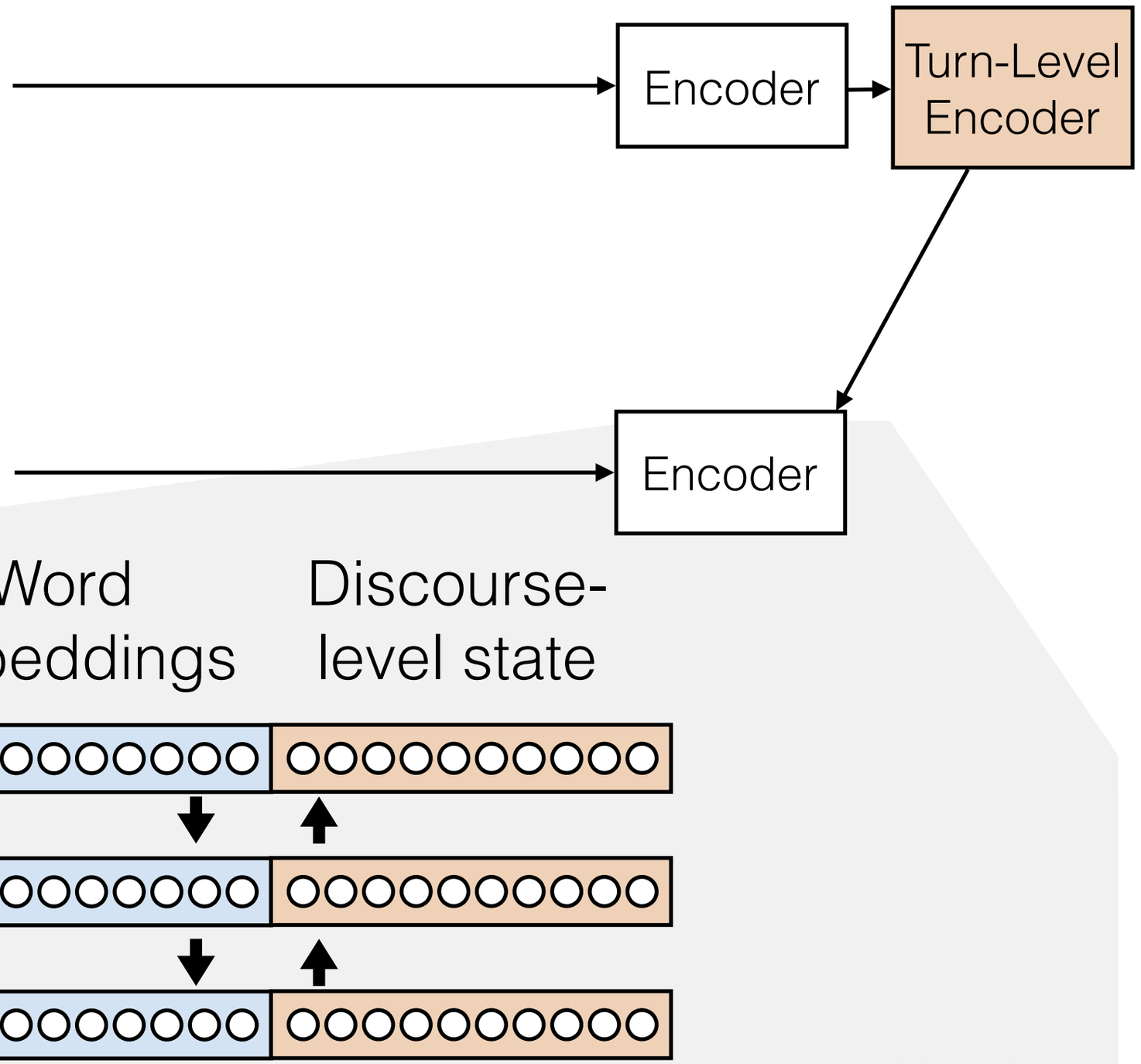


[Suhr et al. 2018]

Turn-level Encoder

Show me flights from Seattle to Boston next Monday

On American Airlines



2. Using State

[Suhr et al. 2018]

Turn-level Encoder

How can we make use of previous requests?

- Efficiently? —> Only run bidirectional RNN over newest utterance
- Using all information from interaction history?

—> Vector state persistent through interaction

Challenges

Challenges

- How do we make use of **previous requests**?
- How do we make use of **previously generated queries**?

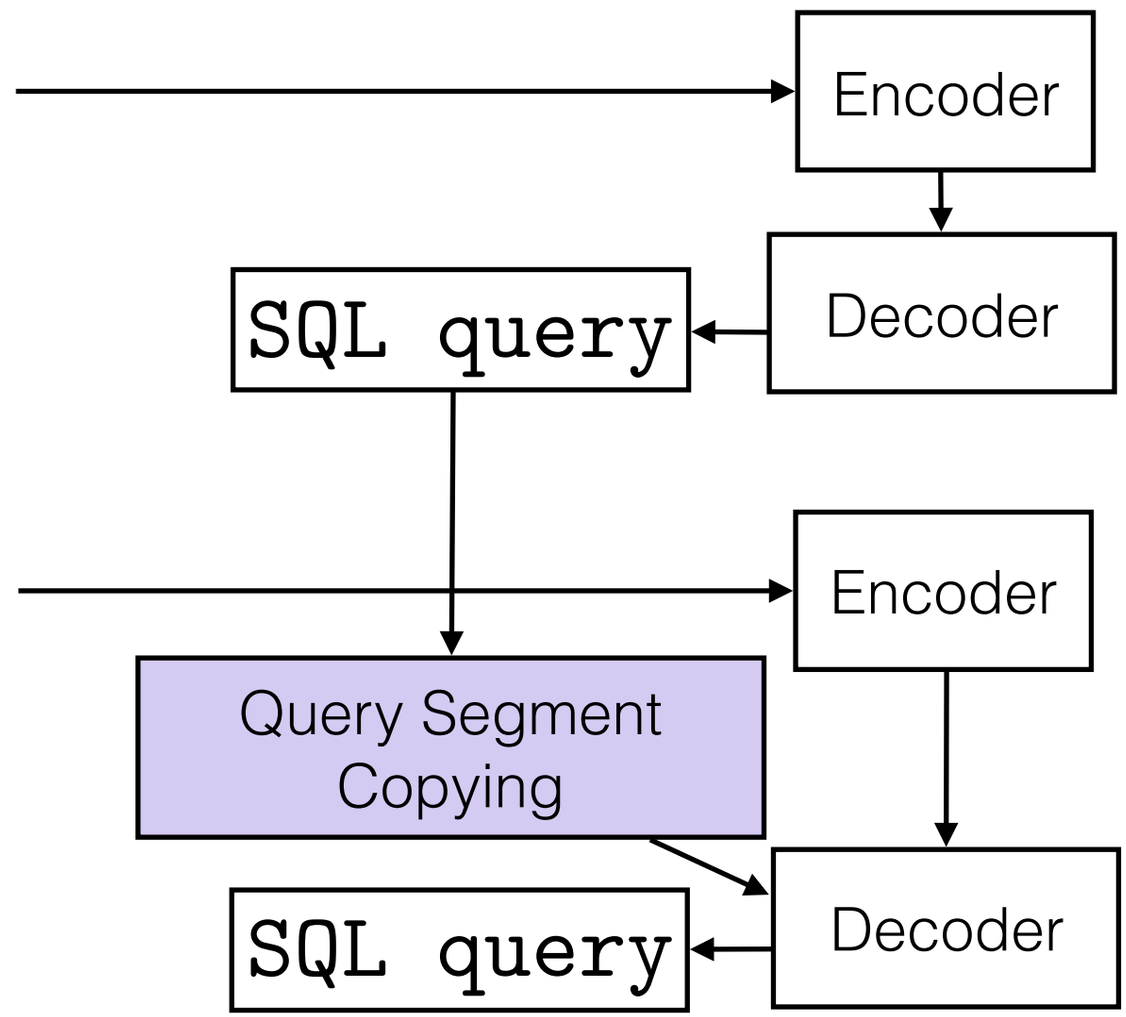
→ **Turn-level encoder**

[Suhr et al. 2018]

Incorporating Previous Queries

Show me flights from Seattle to Boston next Monday

On American Airlines



[Suhr et al. 2018]

Query Segment Copying

Previous Query:

```
(SELECT DISTINCT flight.flight_id FROM flight
WHERE (flight.from_airport IN (SELECT
airport_service.airport_code FROM airport_service
WHERE airport_service.city_code IN (SELECT
city.city_code FROM city WHERE city.city_name =
                    ⋮
```

Decoder

1. Segment Extraction

```
city.city_name = 'SEATTLE'
city.city_name = 'BOSTON'
date_day.year = 1993
date_day.month_number = 2
date_day.day_number = 8
          ⋮
```

Deterministic, operates on the SQL tree

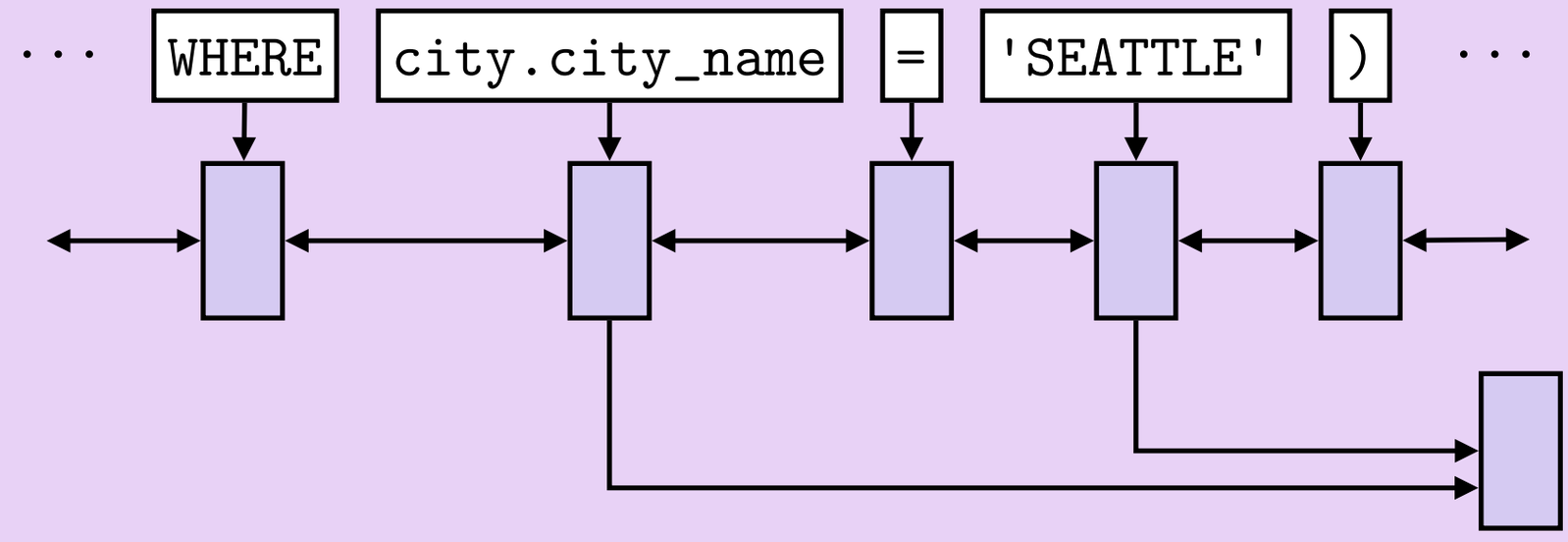
Query Segment Copying

Previous Query:

```
(SELECT DISTINCT flight.flight_id FROM flight
WHERE (flight.from_airport IN (SELECT
airport_service.airport_code FROM airport_service
WHERE airport_service.city_code IN (SELECT
city.city_code FROM city WHERE city.city_name =
```

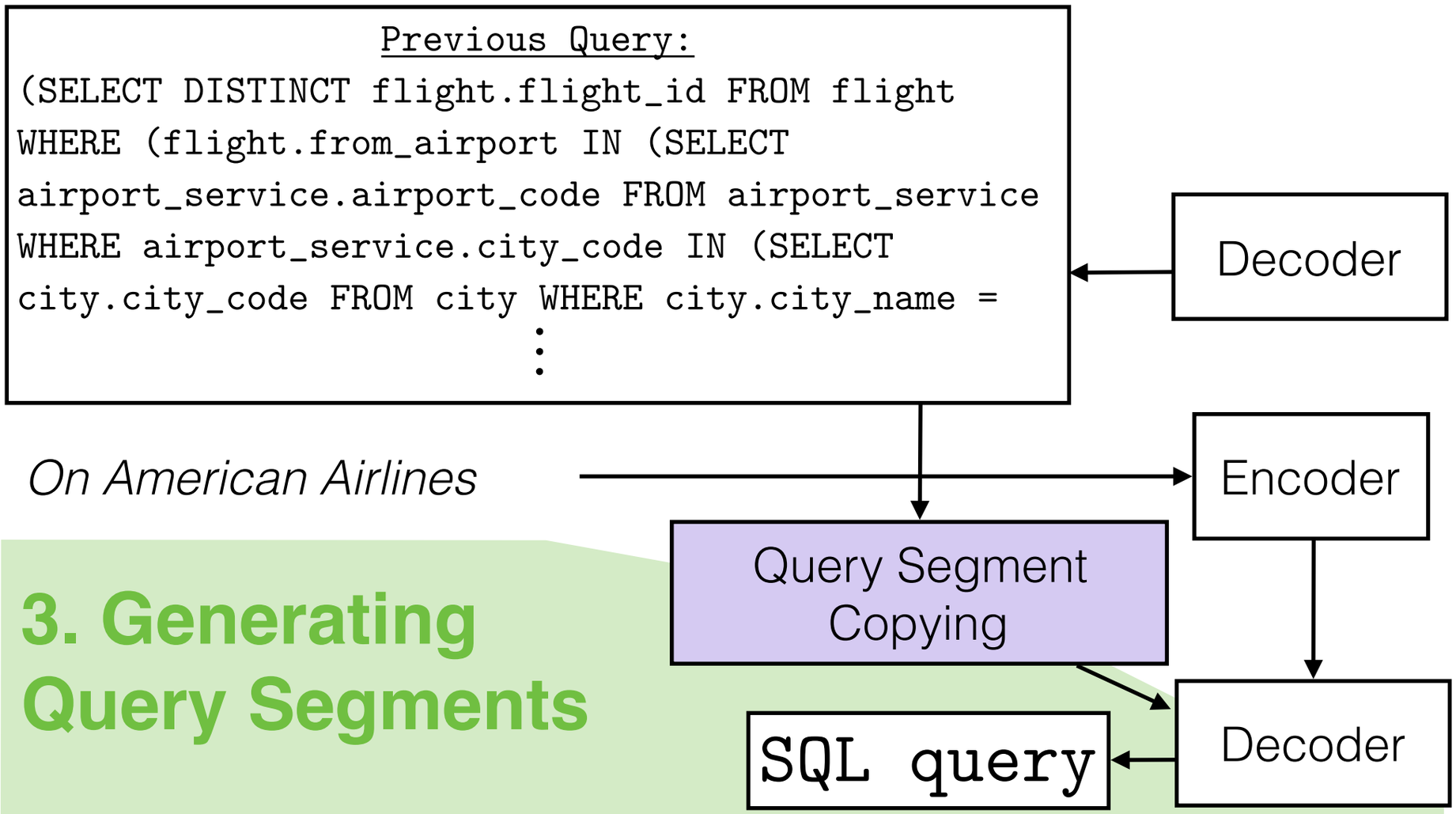
Decoder

2. Segment Encoding




```
city.city_name = 'SEATTLE'
```

Query Segment Copying



3. Generating Query Segments

Probability of query segment computed using its vector state

Challenges

Challenges

- How do we make use of **previous requests**?
- How do we make use of **previously generated queries**?

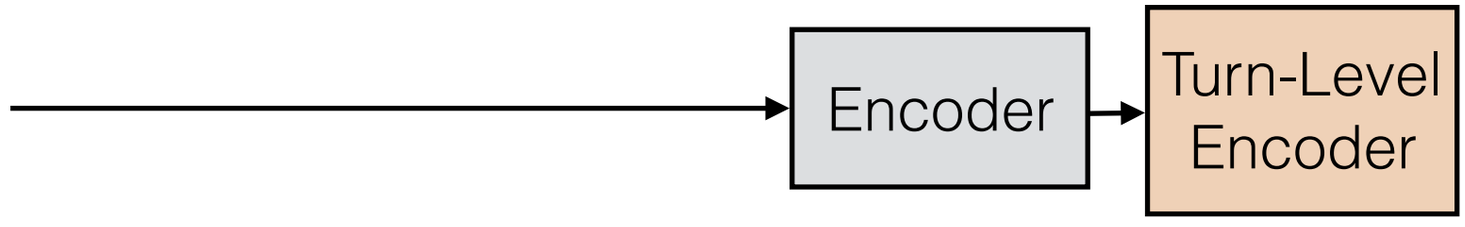
→ **Turn-level encoder**

→ **Query segment copying**

[Suhr et al. 2018]

Inference

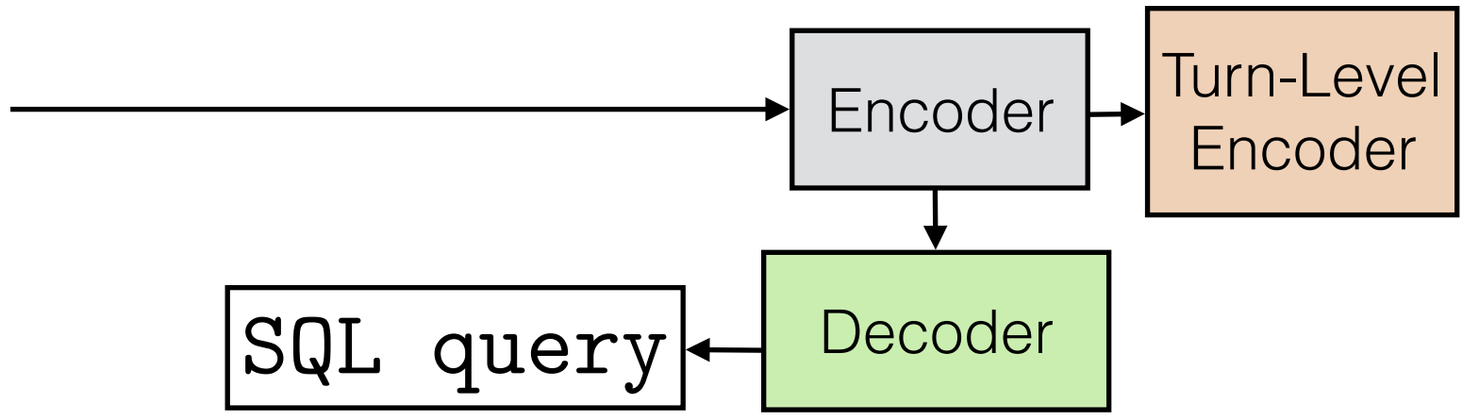
Show me flights from Seattle to Boston next Monday



[Suhr et al. 2018]

Inference

Show me flights from Seattle to Boston next Monday

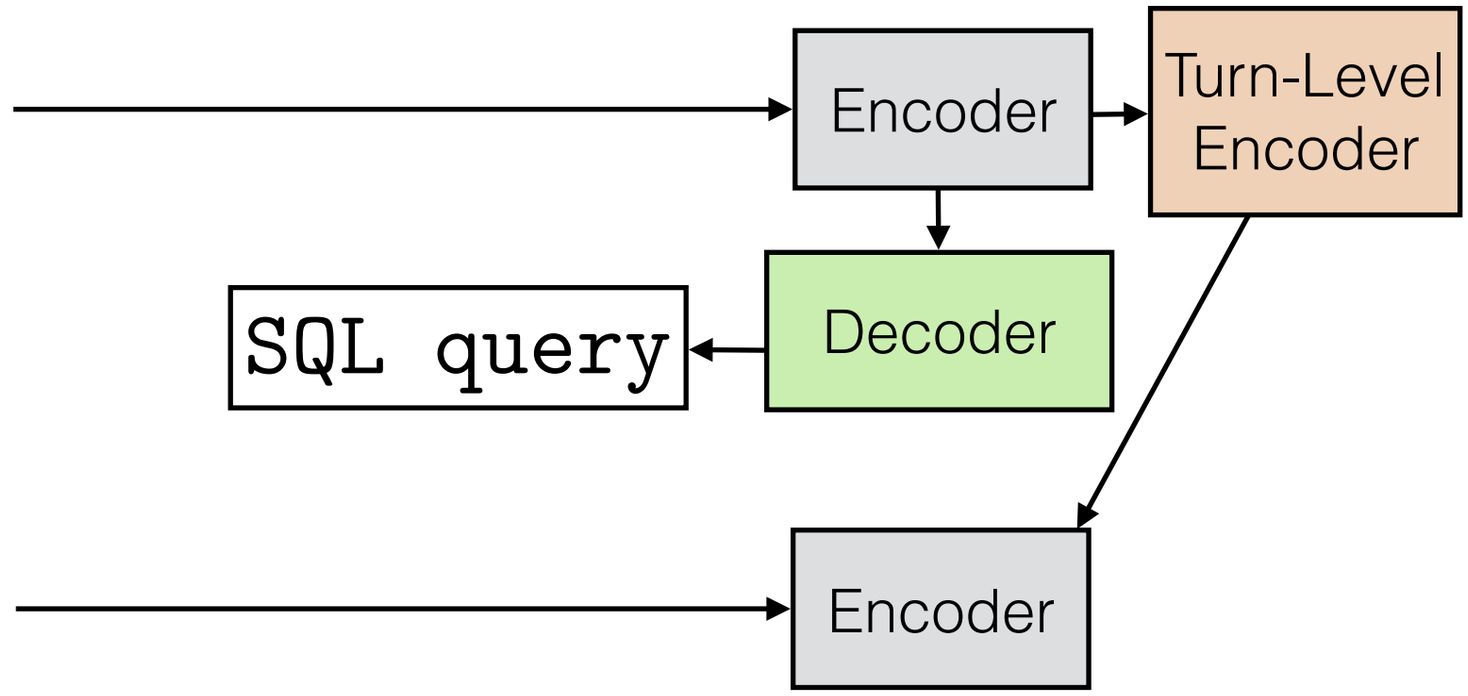


[Suhr et al. 2018]

Inference

Show me flights from Seattle to Boston next Monday

On American Airlines

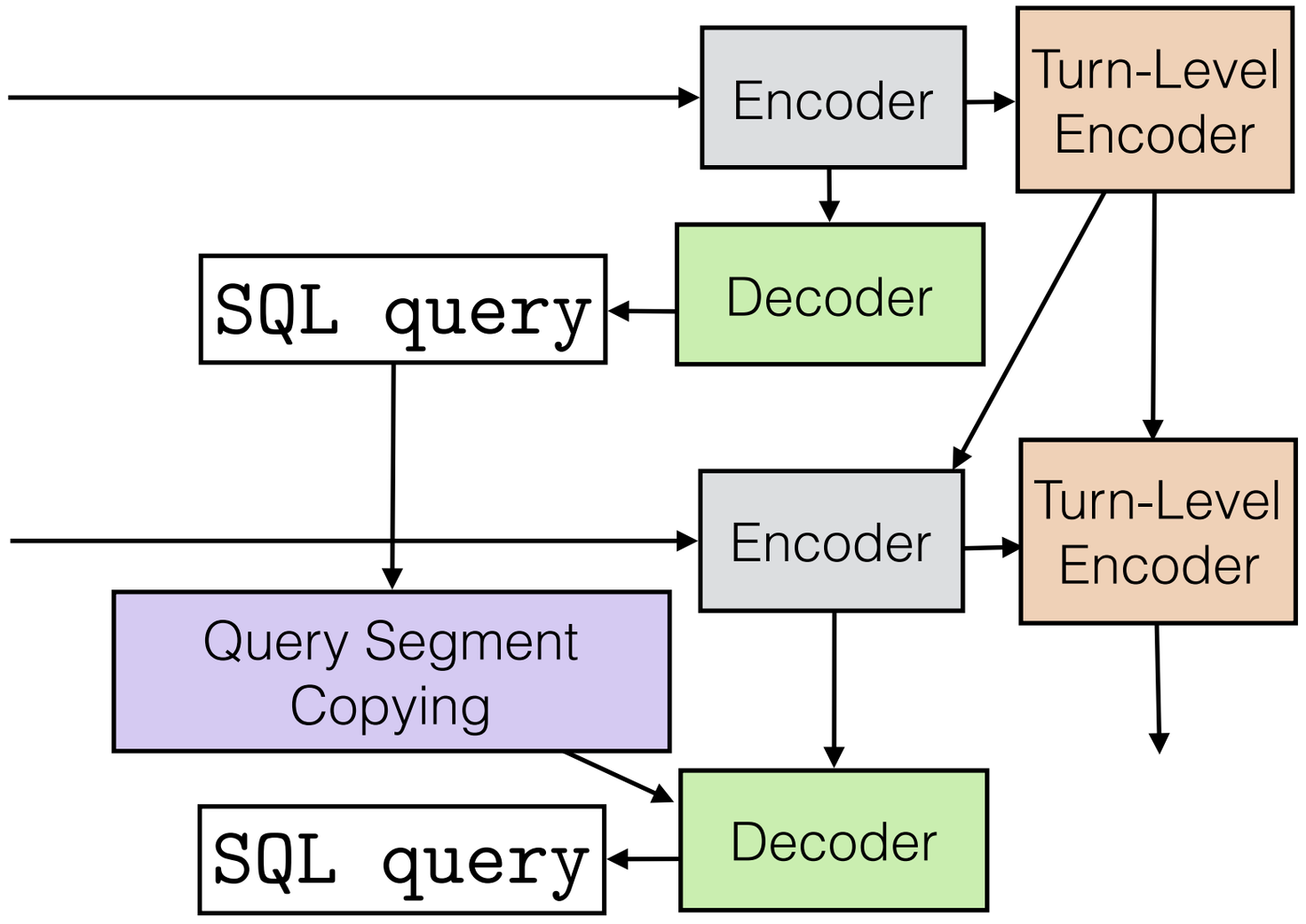


[Suhr et al. 2018]

Inference

Show me flights from Seattle to Boston next Monday

On American Airlines

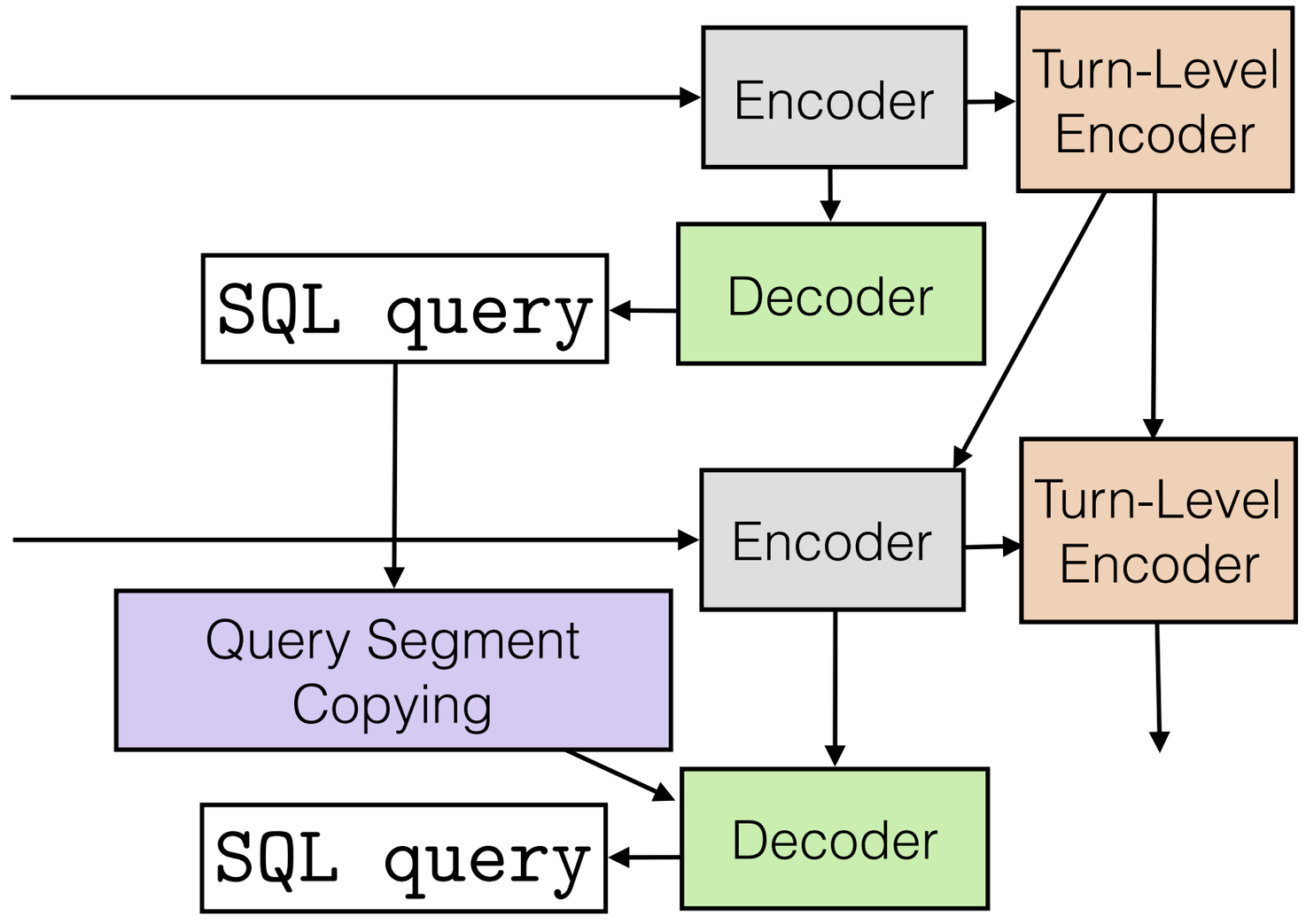


[Suhr et al. 2018]

Inference

Show me flights from Seattle to Boston next Monday

On American Airlines



- Gradually build computation graph for entire interaction
- Directly predict context-dependent representations

[Suhr et al. 2018]

Learning

- Training data: interactions with request-SQL pairs
- Objective: minimize token-level cross-entropy loss

[Suhr et al. 2018]

Interaction-Level Learning

- Each batch consists of an entire interaction
- Build one computation graph, and backpropagate through entire interaction
- Each batch contains a different number of output sequences
- This introduces several learning challenges

[Suhr et al. 2018]

Interaction Loss

Loss for an interaction y

$$\mathcal{L} = \frac{1}{\sum_{i=1}^{n} |\overline{y}_i|} \sum_{i=1}^{n} \sum_{k=1}^{|\overline{y}_i|} \mathcal{L}(y_{i,k})$$

Average over number of tokens in interaction

Sum of token-level loss for all utterances in batch

Problem: sequences in longer interactions have relatively lower gradient due to token-level averaging

[Suhr et al. 2018]

Batch Re-Weighting

Loss for an interaction y with batch re-weighting

$$\mathcal{L} = \frac{n}{B} \frac{1}{\sum_{i=1}^{n} |\overline{y}_i|} \sum_{i=1}^{n} \sum_{k=1}^{|\overline{y}_i|} \mathcal{L}(y_{i,k})$$

Re-normalized based on length of interaction

Solution: gradient will have similar magnitude regardless of interaction length

[Suhr et al. 2018]

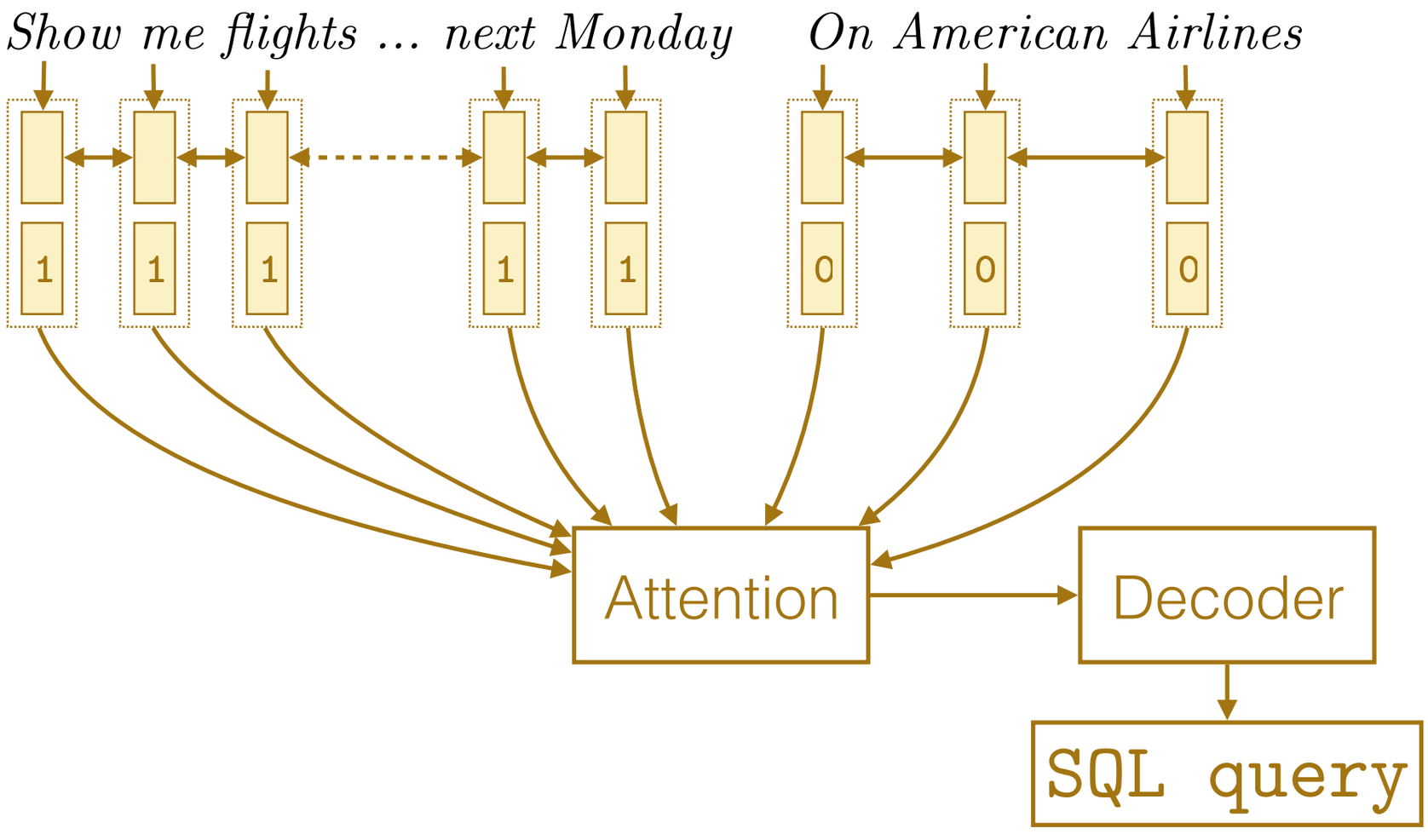
Attention on Interaction History

- Attention is effective for sequence-to-sequence models
- Our model: attend over last few utterances
- **Problem:** hidden states don't encode which utterance they are coming from

[Suhr et al. 2018]

Attention on Interaction History

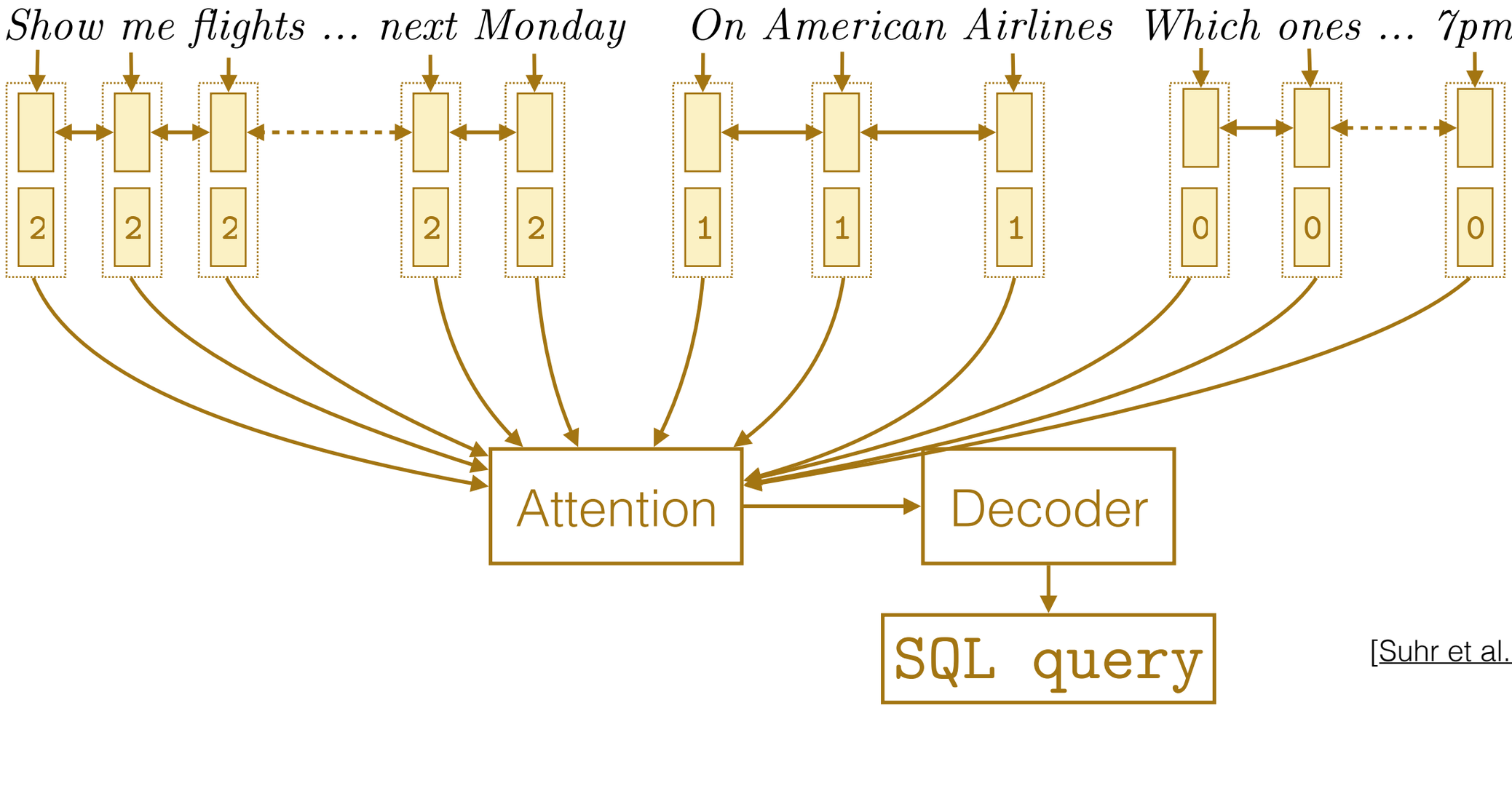
Solution: add positional embeddings to input hidden states in attention computations



[Suhr et al. 2018]

Attention on Interaction History

Solution: add positional embeddings to input hidden states in attention computations



[Suhr et al. 2018]

Learning to Copy Segments

- **Goal:** model should learn to copy as much as possible, when appropriate

- **Challenge:** tokens are not annotated with whether they are generated or copied

- **Solution:** heuristic extraction and alignment of segments

[Suhr et al. 2018]

Segment Extraction and Alignment

Previous Query:

```
(SELECT DISTINCT flight.flight_id
FROM flight WHERE
(flight.from_airport IN (SELECT
airport_service.airport_code FROM
airport_service WHERE
airport_service.city_code IN (SELECT
city.city_code FROM city WHERE
city.city_name =
                    ⋮
```



```
city.city_name = 'SEATTLE'
city.city_name = 'BOSTON'
date_day.year = 1993
date_day.month_number = 2
date_day.day_number = 8
                    ⋮
```

Includes all constituents of the SQL tree

[Suhr et al. 2018]

Segment Extraction and Alignment

Request

On American Airlines

Gold Query

```
(SELECT DISTINCT flight.flight_id
FROM flight ...
city.city_name = 'SEATTLE'))) ...
```

Available Segments

```
1. city.city_name = 'SEATTLE'
2. city.city_name = 'BOSTON'
3. date_day.year = 1993
4. date_day.month_number = 2
5. date_day.day_number = 8
                ⋮
```

[Suhr et al. 2018]

Segment Extraction and Alignment

Request

On American Airlines

Gold Query

```
(SELECT DISTINCT flight.flight_id
FROM flight ...
city.city_name = 'SEATTLE'))) ...
```

Available Segments

```
1. city.city_name = 'SEATTLE'
2. city.city_name = 'BOSTON'
3. date_day.year = 1993
4. date_day.month_number = 2
5. date_day.day_number = 8
                ⋮
```

[Suhr et al. 2018]

Segment Extraction and Alignment

Request

On American Airlines

Gold Query

```
(SELECT DISTINCT flight.flight_id
FROM flight ...
        SEGMENT#1          ))) ...
```

Available Segments

```
1. city.city_name = 'SEATTLE'
2. city.city_name = 'BOSTON'
3. date_day.year = 1993
4. date_day.month_number = 2
5. date_day.day_number = 8
                  ⋮
```

Replace identified segment with reference

[Suhr et al. 2018]

Segment Extraction and Alignment

Request

From Seattle to Denver

Gold Query

```
(SELECT DISTINCT flight.flight_id
FROM flight ...
city.city_name = 'SEATTLE'))) ...
```

Available Segments

```
1. city.city_name = 'SEATTLE'
2. city.city_name = 'BOSTON'
3. date_day.year = 1993
4. date_day.month_number = 2
5. date_day.day_number = 8
                ⋮
```

[Suhr et al. 2018]

Segment Extraction and Alignment

Request

From Seattle to Denver

Gold Query

```
(SELECT DISTINCT flight.flight_id
FROM flight ...
city.city_name = 'SEATTLE'))) ...
```

Available Segments

```
1. city.city_name = 'SEATTLE'
2. city.city_name = 'BOSTON'
3. date_day.year = 1993
4. date_day.month_number = 2
5. date_day.day_number = 8
                ⋮
```

Don’t replace: entity present in current request

[Suhr et al. 2018]

Segment Extraction and Alignment

Request

From Seattle to Denver

Gold Query

```
(SELECT DISTINCT flight.flight_id
FROM flight ...
city.city_name = 'SEATTLE'))) ...
```

Available Segments

```
1. city.city_name = 'SEATTLE'
2. city.city_name = 'BOSTON'
3. date_day.year = 1993
4. date_day.month_number = 2
5. date_day.day_number = 8
                ⋮
```

Don’t replace: entity present in current request

[Suhr et al. 2018]

Experiments

- **Seq2Seq w/o history**
  seq2seq on current utterance only

- **Seq2Seq + history**
  seq2seq by concatenating last four utterances

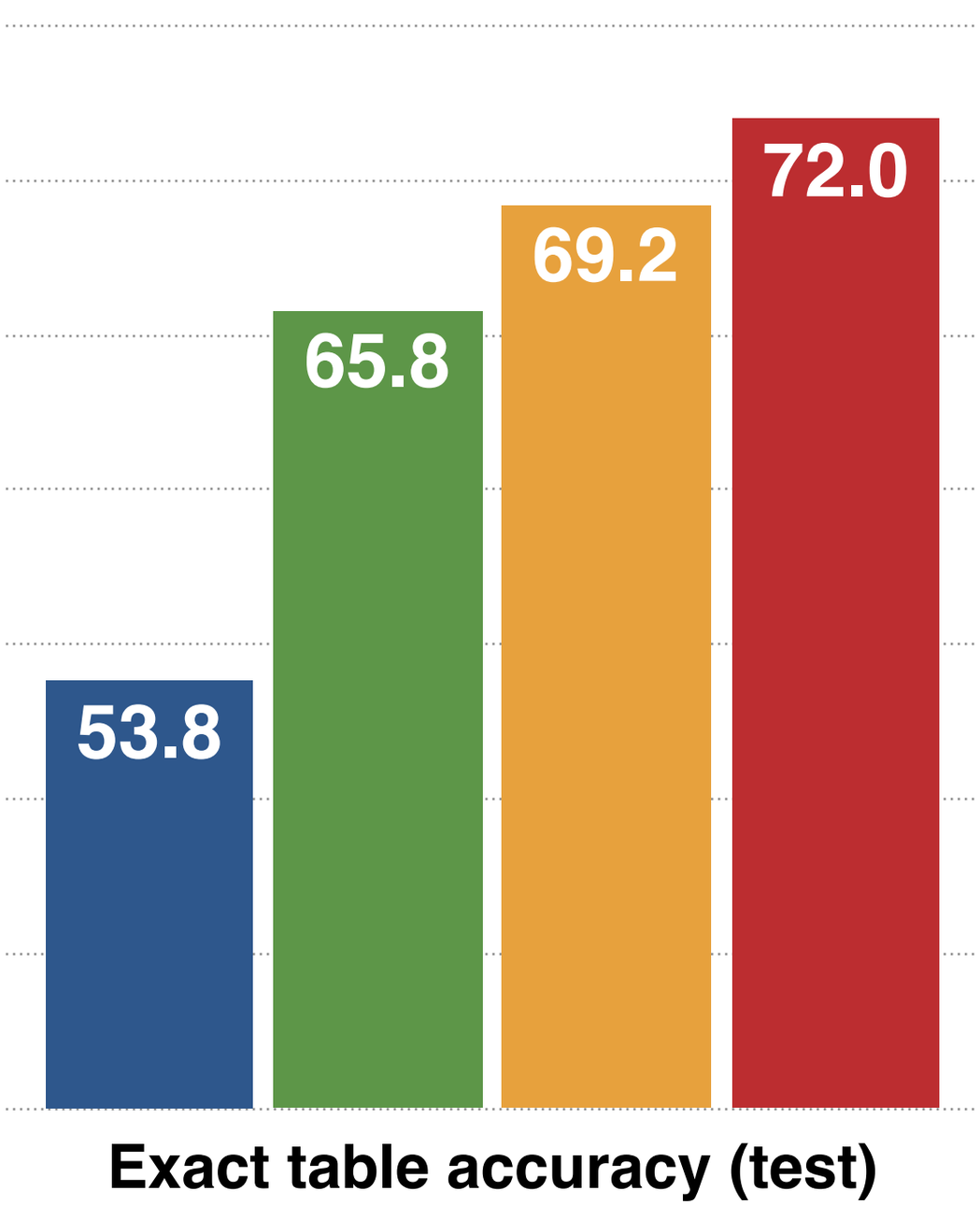
- **Full model**
  use turn-level encoder
  and query segment copying

Evaluation metric:
Exact table accuracy

- **Measure effect of error propagation:**
  full model with access to gold previous query

[Suhr et al. 2018]

Test Results

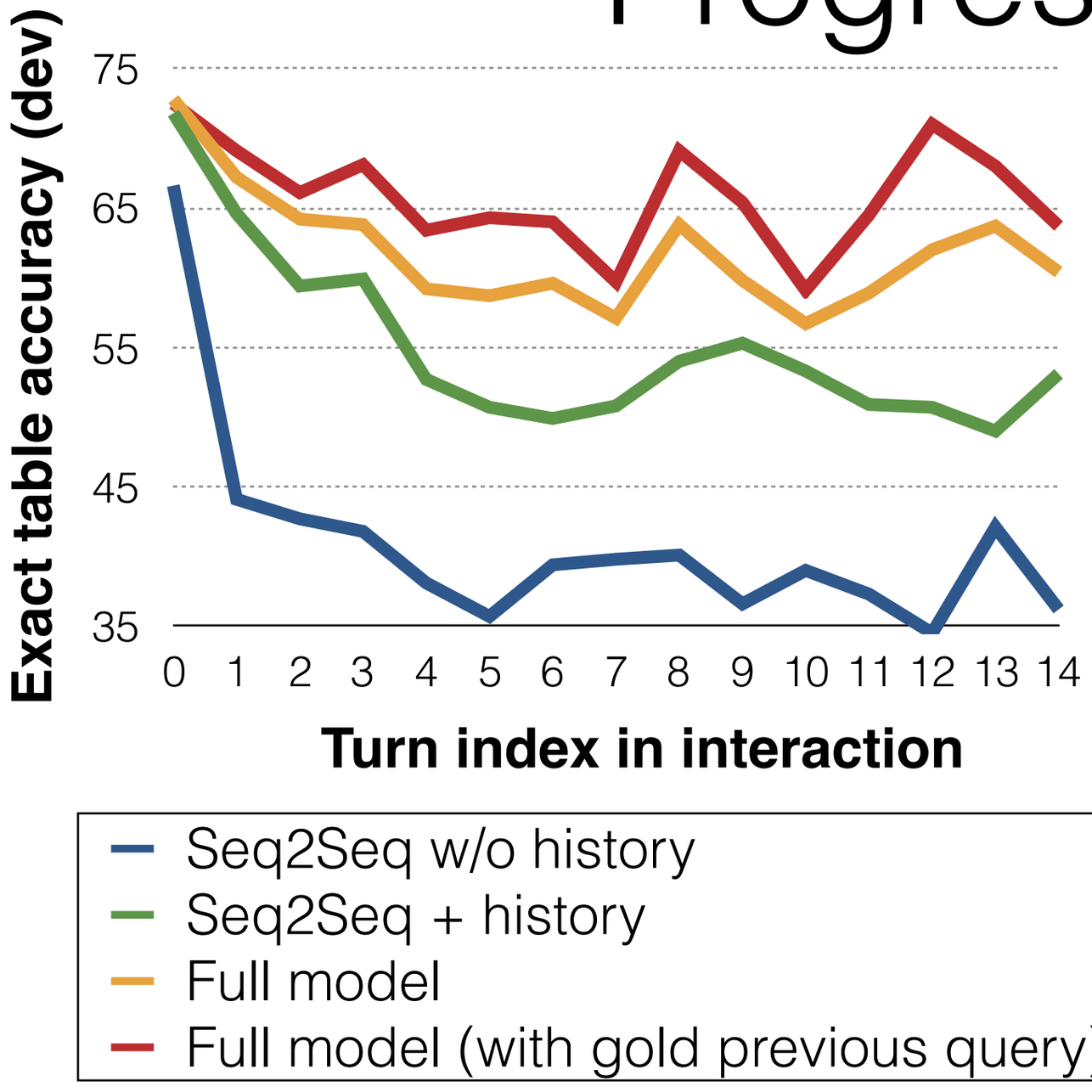


- Seq2Seq w/o history
- Seq2Seq + history
- Full model
- Full model (with gold previous query)

- Using interaction history is critical
- Error propagation contributes about 3% performance drop

[Suhr et al. 2018]

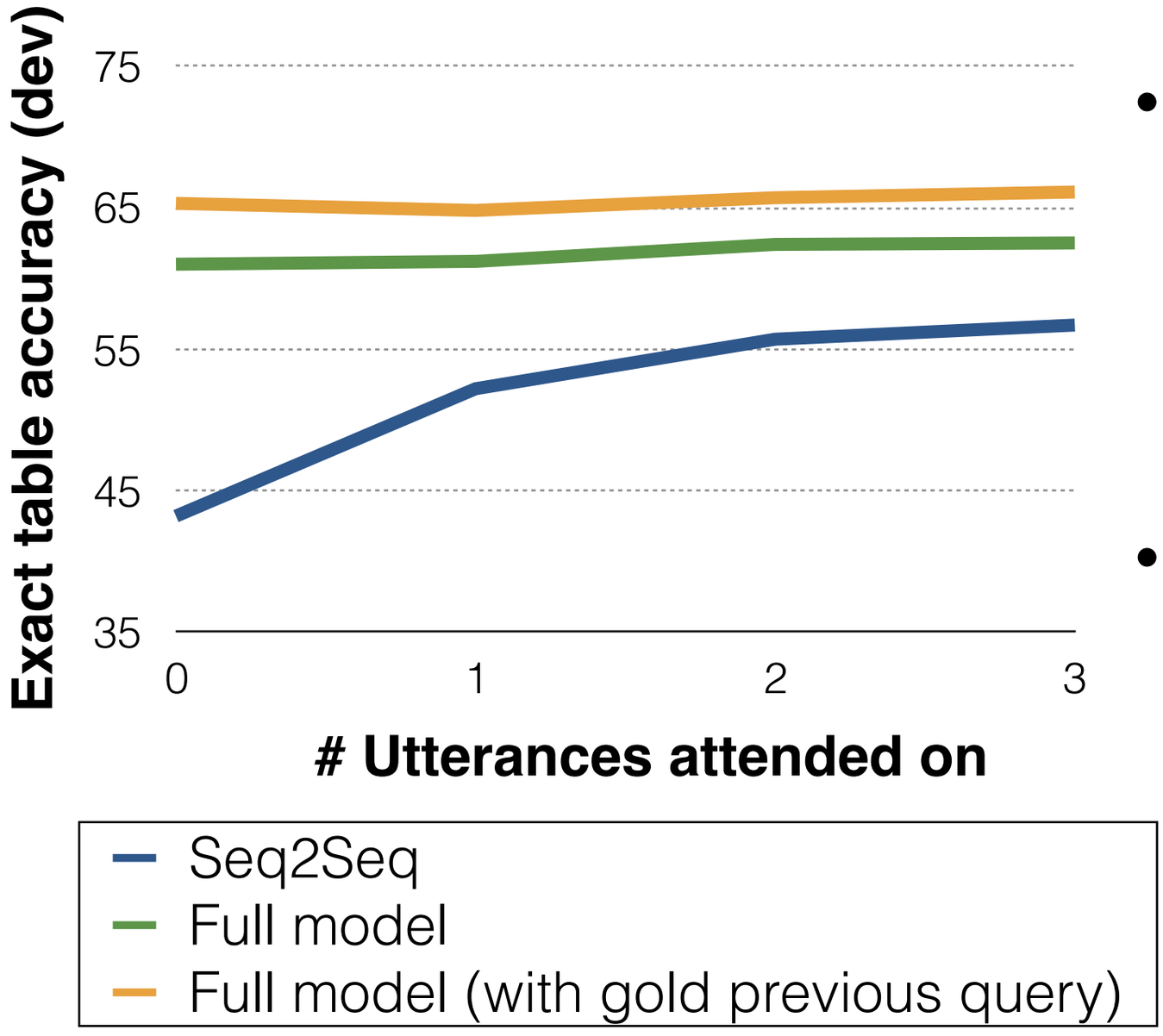
Performance as Interactions Progress



- Without interaction history, performance drops immediately
- Our model: relatively stable

[Suhr et al. 2018]

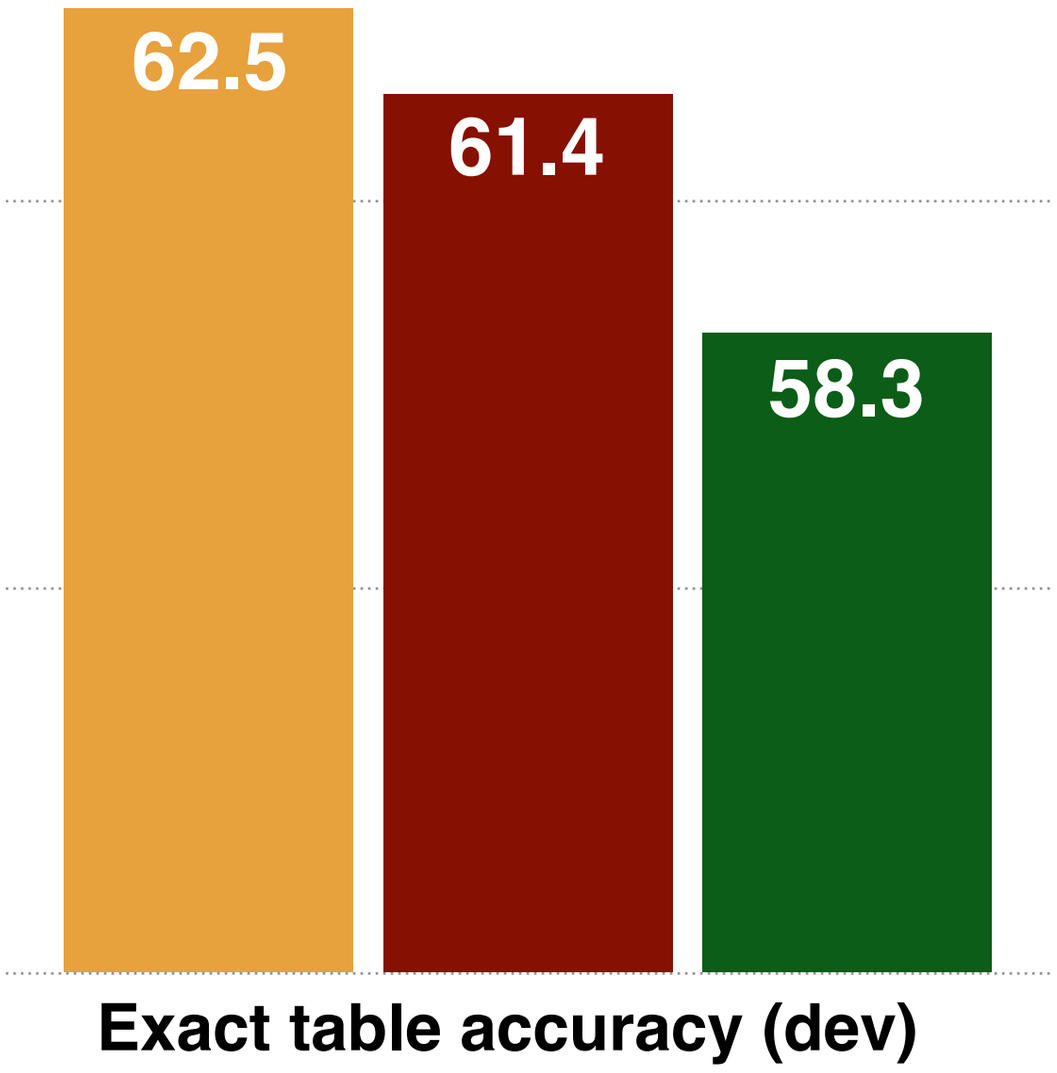
Attention Analysis



- With our mechanisms, can still perform well while only attending on current utterance
- Seq2seq models need to attend over more utterances — inefficient

[Suhr et al. 2018]

Mechanism Ablations

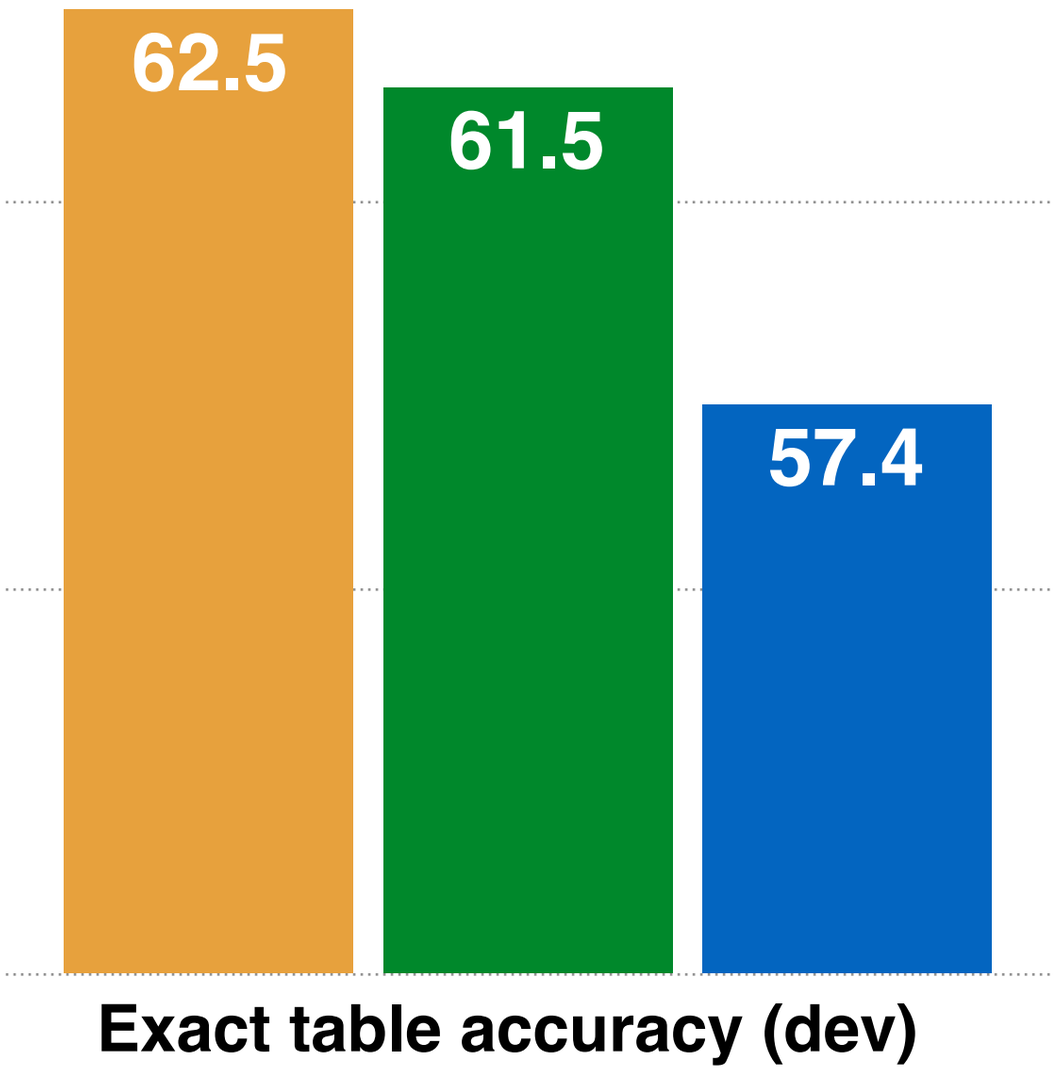


Both mechanisms contribute to performance

- Full model
- w/o turn-level encoder
- w/o query segment copying

[Suhr et al. 2018]

Turn-Level Encoder Ablations



Learning problems addressed by batch re-weighting and input positional embeddings

- Full model
- w/o batch re-weighting
- w/o input positional embeddings

[Suhr et al. 2018]

Error Propagation

User *Which ones arrive around 7pm?*

SQL Query

```
( SELECT DISTINCT flight.flight_id FROM flight WHERE
( flight.from_airport IN ( SELECT
airport_service.airport_code FROM airport_service WHERE
airport_service.city_code IN ( SELECT city.city_code FROM
city WHERE city.city_name = 'ATLANTA' ) ) AND
( flight.to_airport IN ( SELECT airport_service.airport_code
FROM airport_service WHERE airport_service.city_code IN
( SELECT city.city_code FROM city WHERE city.city_name =
'BALTIMORE' ) ) AND ( flight.flight_days IN ( SELECT
days.days_code FROM days WHERE days.day_name IN ( SELECT
date_day.day_name FROM date_day WHERE date_day.year = 1991
AND date_day.month_number = 9 AND date_day.day_number =
6 ) ) AND ( flight.arrival_time >= 1630 AND
flight.arrival_time <= 1730 ) ) ) ) ) ;
```

[Suhr et al. 2018]

Error Propagation

User *Which ones arrive around 7pm?*

SQL Query

Error: looking for flights around 5pm

```
flight.arrival_time >= 1630 AND
flight.arrival_time <= 1730
```

[Suhr et al. 2018]

Error Propagation

User *Which kind of airplane is that?*

SQL Query

```
( SELECT DISTINCT aircraft.aircraft_code FROM aircraft WHERE
aircraft.aircraft_code IN ( SELECT
equipment_sequence.aircraft_code FROM equipment_sequence
WHERE equipment_sequence.aircraft_code_sequence IN ( SELECT
flight.aircraft_code_sequence FROM flight WHERE
( flight.arrival_time >= 1630 AND flight.arrival_time <=
1730  AND ( flight.from_airport IN ( SELECT
airport_service.airport_code FROM airport_service WHERE
airport_service.city_code IN ( SELECT city.city_code FROM
city WHERE city.city_name = 'ATLANTA' ) )
```

⋮

[Suhr et al. 2018]

Code and Data

- Code available: https://github.com/clic-lab/atis
- Preprocessed data available:
  - https://github.com/clic-lab/atis_data
  - Requires LDC93S5, LDC94S19, and LDC95S26
  - Email suhr@cs.cornell.edu for access

[Suhr et al. 2018]

Outline

1. **Case study:** natural language interface to a database
    A. Previous instructions
    B. Previous interpretations of instructions
2. **Situated interaction context**
3. **Open questions**

Situated Understanding

→ *Empty out the leftmost beaker of purple chemical*

Then, add the contents of the first beaker to the second

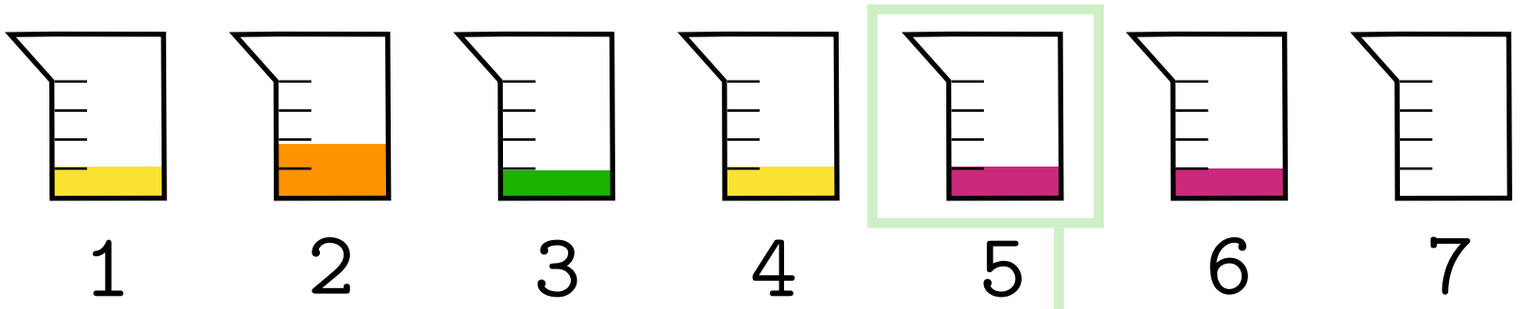
Mix it

Then, drain 1 unit from it

Same for 1 more unit

[Long et al. 2016]

Situated Understanding



→ *Empty out the leftmost beaker of purple chemical*

Then, add the contents of the first beaker to the second

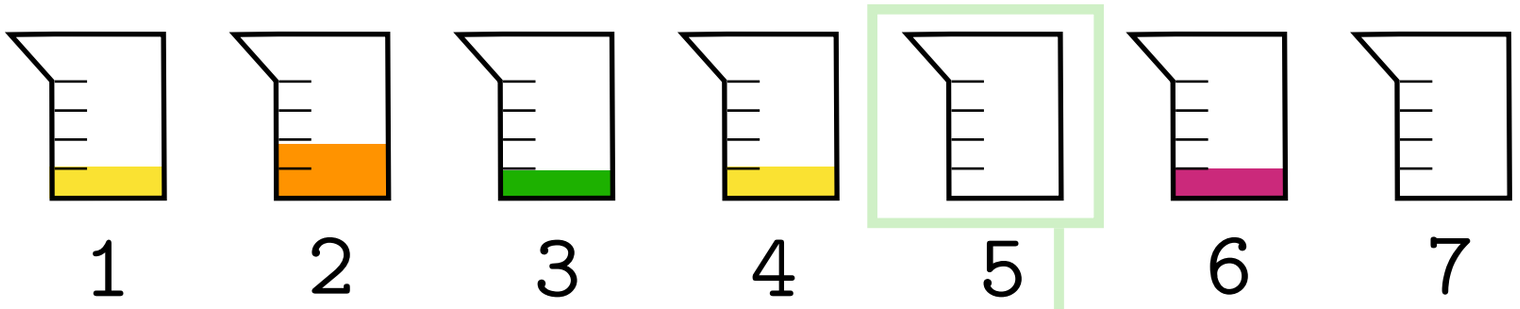
Mix it

Then, drain 1 unit from it

Same for 1 more unit

[Long et al. 2016]

Situated Understanding



→ *Empty out the leftmost beaker of purple chemical*

Then, add the contents of the first beaker to the second

Mix it

Then, drain 1 unit from it

Same for 1 more unit

[Long et al. 2016]

Situated Understanding

Empty out the leftmost beaker of purple chemical

→ *Then, add the contents of the first beaker to the second*

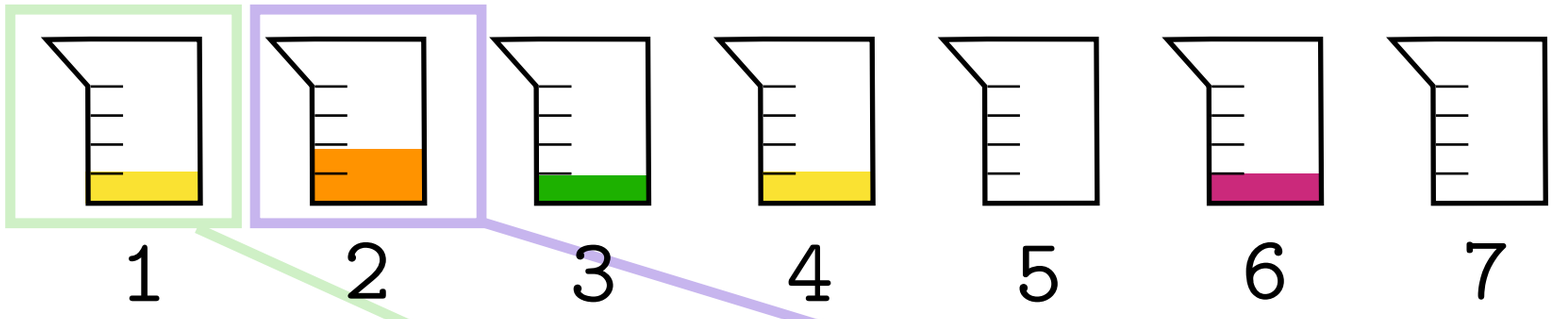
Mix it

Then, drain 1 unit from it

Same for 1 more unit

[Long et al. 2016]

Situated Understanding



Empty out the leftmost beaker of purple chemical

→ *Then, add the contents of the first beaker to the second*

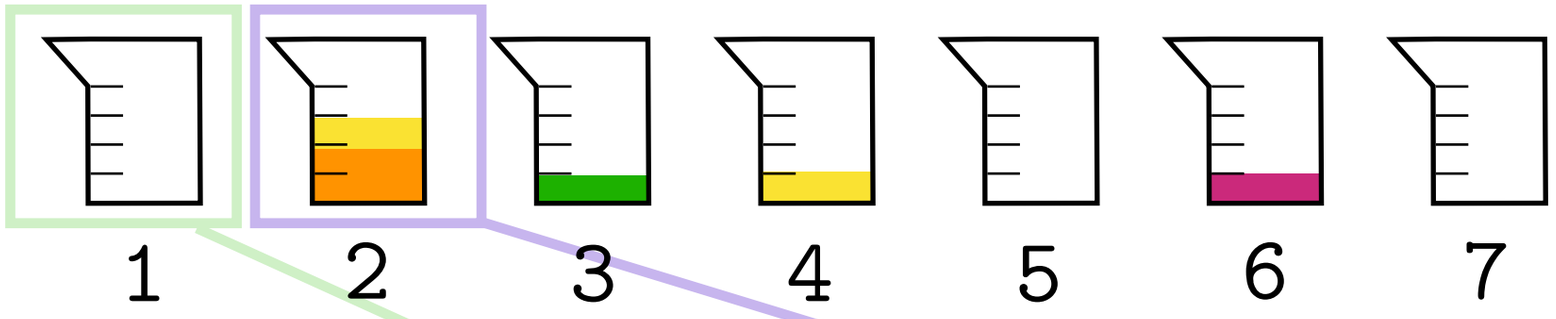
Mix it

Then, drain 1 unit from it

Same for 1 more unit

[Long et al. 2016]

Situated Understanding



Empty out the leftmost beaker of purple chemical

Then, add the contents of the first beaker to the second

Mix it

Then, drain 1 unit from it

Same for 1 more unit

[Long et al. 2016]

Situated Understanding

Empty out the leftmost beaker of purple chemical

Then, add the contents of the first beaker to the second

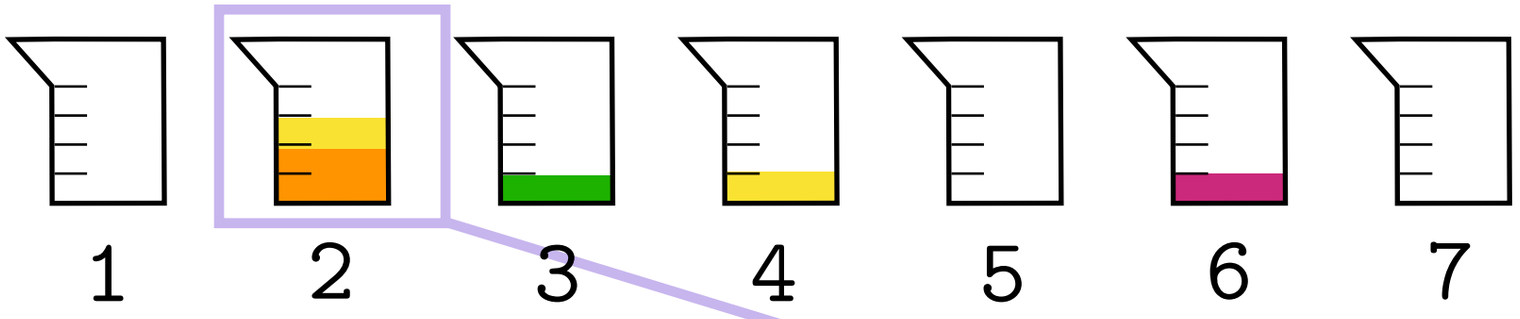
→ *Mix it*

Then, drain 1 unit from it

Same for 1 more unit

[Long et al. 2016]

Situated Understanding



Empty out the leftmost beaker of purple chemical

Then, add the contents of the first beaker to the second

→ *Mix it*

Then, drain 1 unit from it

Same for 1 more unit

[Long et al. 2016]

Situated Understanding

Empty out the leftmost beaker of purple chemical

Then, add the contents of the first beaker to the second

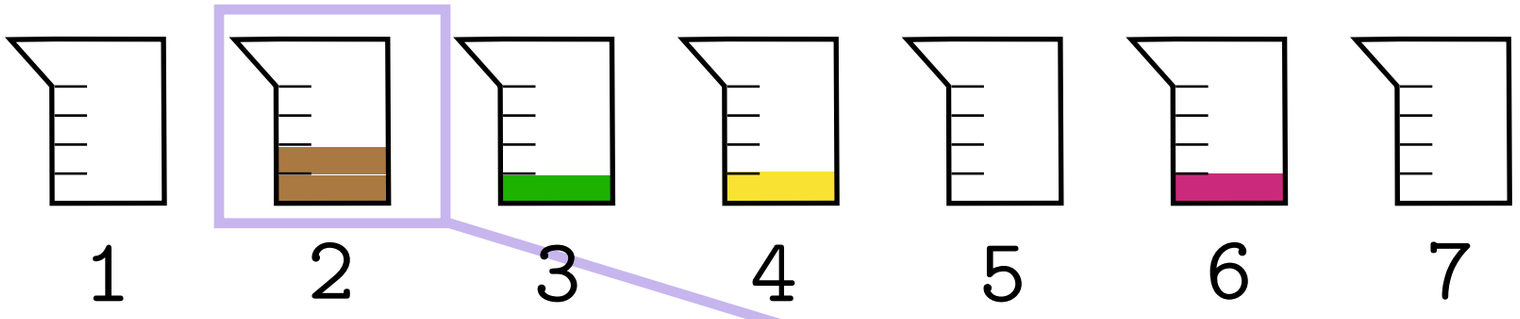
Mix it

→ *Then, drain 1 unit from it*

Same for 1 more unit

[Long et al. 2016]

Situated Understanding



Empty out the leftmost beaker of purple chemical

Then, add the contents of the first beaker to the second

Mix it

→ *Then, drain 1 unit from it*

Same for 1 more unit

[Long et al. 2016]

Situated Understanding

Empty out the leftmost beaker of purple chemical

Then, add the contents of the first beaker to the second

Mix it

Then, drain 1 unit from it

→ *Same for 1 more unit*

[Long et al. 2016]

Situated Understanding

Empty out the leftmost beaker of purple chemical

Then, add the contents of the first beaker to the second

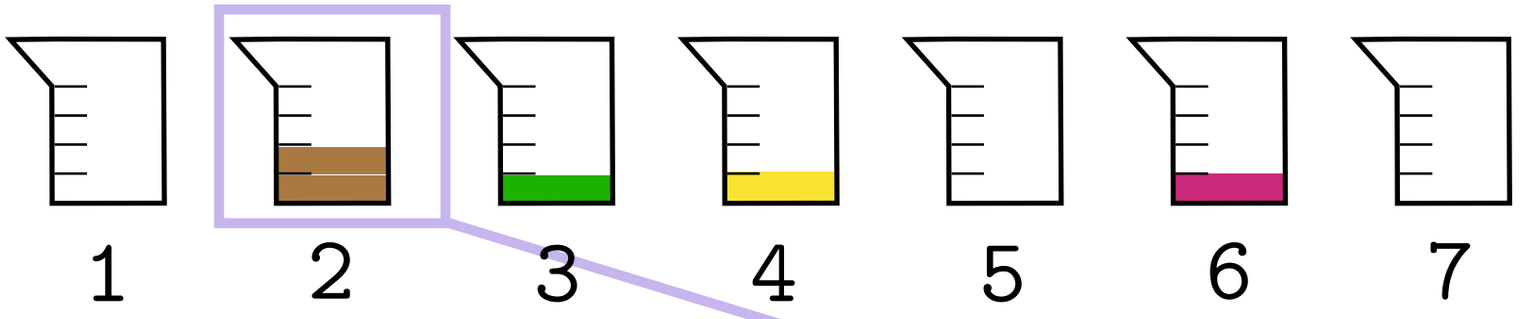
Mix it

Then, drain 1 unit from it

Same for 1 more unit

[Long et al. 2016]

Situated Understanding



Empty out the leftmost beaker of purple chemical

Then, add the contents of the first beaker to the second

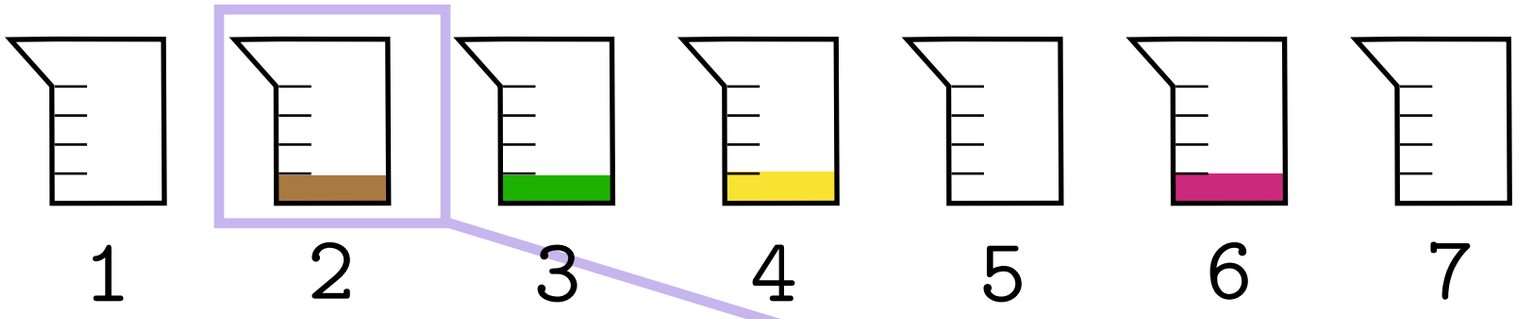
Mix it

Then, drain 1 unit from it

Same for 1 more unit

[Long et al. 2016]

Situated Understanding



Empty out the leftmost beaker of purple chemical

Then, add the contents of the first beaker to the second

Mix it

Then, drain 1 unit from it

Same for 1 more unit

[Long et al. 2016]

Meaning representation grounded in **environment** and **interaction history**

1. Mix the second beaker

```
mix(index(allObjects, 2))
```

2. Pour it into the third beaker

```
pour(prevArg1(1), index(allObjects, 3))
```

[Long et al. 2016, Guu et al. 2017, Fried et al. 2018]

Meaning representation grounded in **environment** and **interaction history**

1. Mix the second beaker

```
mix(index(allObjects, 2))
```

2. Pour it into the third beaker

```
pour(prevArg1(1), index(allObjects, 3))
```

[Long et al. 2016, Guu et al. 2017, Fried et al. 2018]

Meaning representation grounded in **environment** and **interaction history**

1. Mix the second beaker

`mix(index(allObjects, 2))`

2. Pour it into the third beaker

`pour(prevArg1(1), index(allObjects, 3))`

[Long et al. 2016, Guu et al. 2017, Fried et al. 2018]

Meaning representation grounded in **environment** and **interaction history**

1. Mix the second beaker

`mix(index(allObjects, 2))`

2. Pour it into the third beaker

`pour(prevArg1(1), index(allObjects, 3))`

[Long et al. 2016, Guu et al. 2017, Fried et al. 2018]

Suhr and Artzi 2018

- **Sequential understanding:** attention over previous instructions
- **Grounding in environment:** attention over environments, updated as actions are executed
- **Talk: Wednesday 18 July, 11:20 (Session 7A)**

[Other work on situated understanding: Artzi and Zettlemoyer 2013, Mei et al. 2015, Misra et al. 2017]

[Suhr and Artzi 2018]

Outline

1. **Case study:** natural language interface to a database
    A. Previous instructions
    B. Previous interpretations of instructions
2. **Situated interaction context**
3. **Open questions**

Open Questions

- **Learning by interacting:** system questions and user clarifications as a learning signal

- **Robustness:** training to recover from errors

- **Reproducibility and evaluation:** working without static corpora

- **Latent decisions:** learning to distinguish between meaning derived from current utterance vs. interaction history

Learning by Interacting

- **Existing work:** make use of dialogues to learn context-independent semantic parsers

[Artzi and Zettlemoyer 2011, Thomason et al. 2015, Padmakumar et al. 2017, Gur et al 2018]

Learning by Interacting

System *How can I help you?*

User *I would like to fly from Atlanta, Georgia to London, England on September 24th in the early evening. I would like to return on October 1st departing from London in the late morning.*

[Artzi and Zettlemoyer 2011]

Learning by Interacting

System *How can I help you?*

User *I would like to fly from Atlanta, Georgia to London, England on September 24th in the early evening. I would like to return on October 1st departing from London in the late morning.*

System *Leaving what city?*

System failed to recognize city constraint

[Artzi and Zettlemoyer 2011]

Learning by Interacting

System *How can I help you?*

User *I would like to fly from Atlanta, Georgia to London, England on September 24th in the early evening. I would like to return on October 1st departing from London in the late morning.*

System *Leaving what city?*

User *Atlanta, Georgia*

Can learn from user's response

[Artzi and Zettlemoyer 2011]

Learning by Interacting

- **Existing work:** make use of dialogues to learn context-independent semantic parsers

- **Limitations:** simple domains; only in context-independent setting

[Artzi and Zettlemoyer 2011, Thomason et al. 2015, Padmakumar et al. 2017, Gur et al 2018]

Robustness

- During training, models should learn how users react to its mistakes
- Learn to recover from mistakes and avoid error propagation
- **Existing work:** simulated users
- Can collect unlimited number of interactions, but restricted to synthetic language

[Schatzmann et al. 2006, Pietquin and Dutoit 2006, Schatzmann et al. 2007, Keizer et al. 2010, Li et al. 2016, El Asri et al. 2016, Crook and Marin 2017]

Reproducibility and Evaluation

- **Difficult to compare:** No fixed test set when evaluating with live interactions
- Human judgment of understanding is subjective
- **Utterance-level evaluation:** annotation effort and expertise required
- **Interaction-level evaluation:** goal completion

[Stent et al. 2005, Yang et al. 2012, Liu et al. 2017]

Latent Decisions

- **Context-independent:** how do output tokens align with input utterance?
- **Context-dependent:** tokens can also align to interaction history
- **Large space of latent decisions:** is each output token derived from interaction history, or current utterance?
- **Existing work:** heuristic approaches that place assumptions on how context is used

Summary

1. **Case study:** natural language interface to a database
    A. Previous instructions
    B. Previous interpretations of instructions
2. **Situated interaction context**
3. **Open questions**