Semantic Parsing

Hsuan-Tung Peng, Andy Su

2020/03/03

Outline

- ❑ What is Semantic Parsing?

- ❑ (Dong et al, 2016): Language to Logical Form with Neural Attention

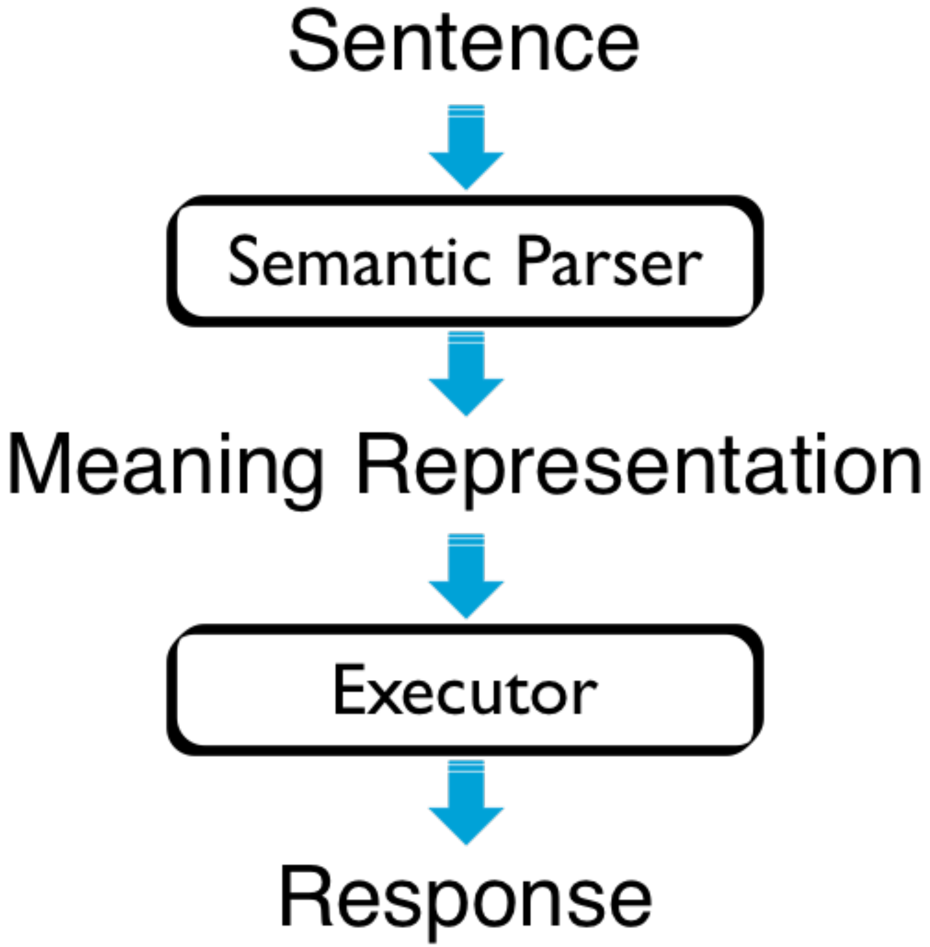
- ❑ (Suhr et al, 2018): Learning to Map ***Context-Dependent*** Sentences to Executable Formal Queries

Semantic Parsing

- Translate ***natural language utterances (NLUs)*** to ***meaning representation (MR)***

$$f : \text{sentence} \rightarrow \text{logical form}$$

- Why do we want to do semantic parsing?



Language to Meaning

Semantic Parsing

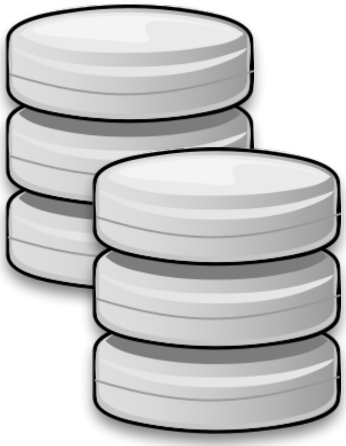
Recover **complete** meaning representation

More informative

Example Task

Database Query

What states border Texas?



Oklahoma
New Mexico
Arkansas
Louisiana

Language to Meaning

Semantic Parsing

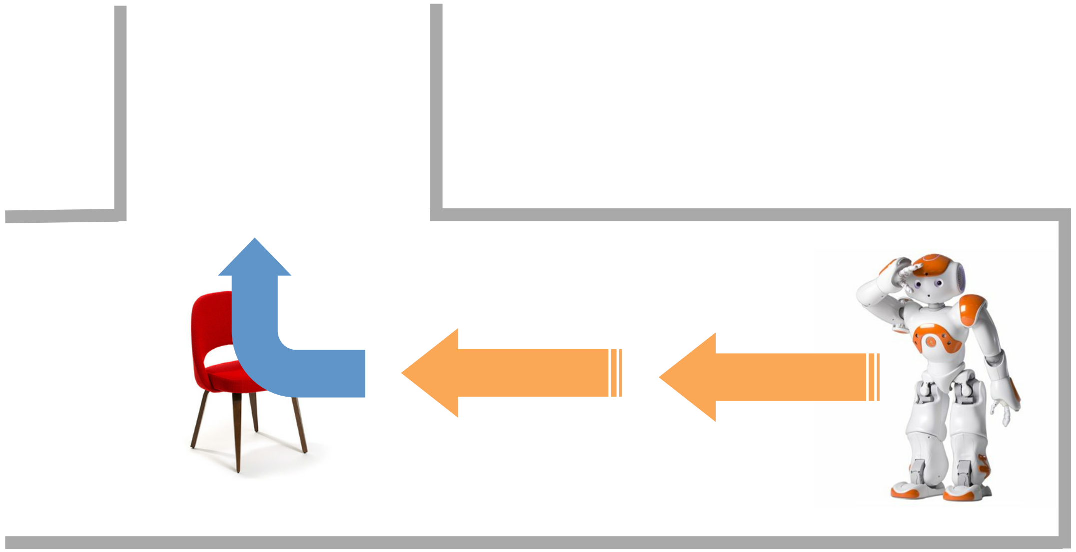
Recover **complete** meaning representation

More informative

Example Task

Instructing a Robot

at the chair,
turn right



Language to Meaning

<u>Complete</u> meaning is sufficient to complete the task

- Convert to database query to get the answer
- Allow a robot to do planning

Semantic Parsing Example

GEOQuery

This is a standard semantic parsing benchmark which contains 880 queries to a database of U.S. geography.

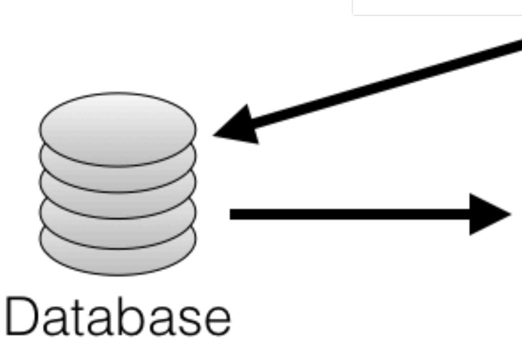
Training Dataset	Test Dataset
600 queries	280 queries

Question:

which state has the most rivers running through it?

Logical form:

```
argmax $0
(state:t $0)
(count $1 (and
            (river:t $1)
            (loc:t $1 $0)))
```



Answer:

Alaska

[Zelle and Mooney 1996; Tang and Mooney 2001]

Semantic Parsing Example

JOBS

This benchmark dataset contains 640 queries to a database of job listings

Training Dataset	Test Dataset
500 queries	140 queries

Question:

what microsoft jobs do not require a bscs?

Logical form:

answer(company(J,'microsoft'), job(J), not((req deg(J,'bscs'))))

[Zelle and Mooney 1996; Tang and Mooney 2001]

Semantic Parsing Example

ATIS

This dataset has 5,410 queries to a flight booking system.

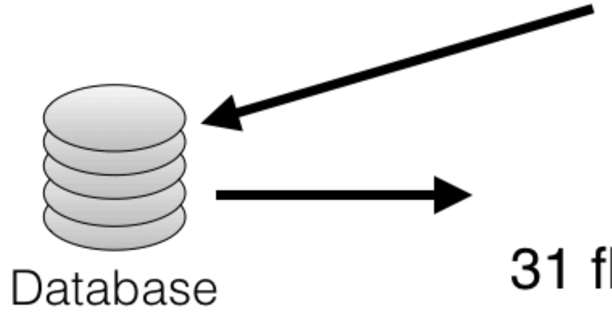
Training Dataset	Development Dataset	Test Dataset
4480 instances	480 instances	450 instances

Request:

Show me flights from Pittsburgh to Seattle

Logical form:

lambda $0 e
(and (flight $0)
(from $0 pittsburgh:ci)
(to $0 seattle:ci))



Result:

31 flights available

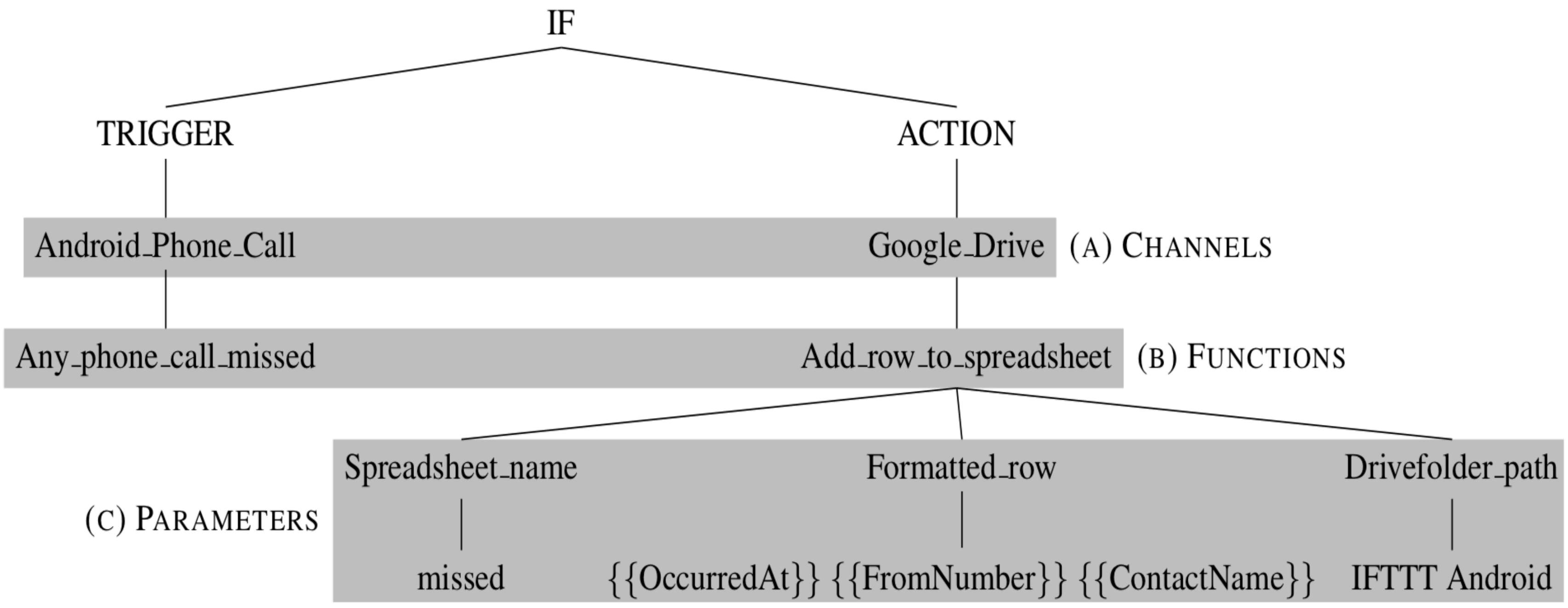
[Hemphill et al. 1990; Dahl et al. 1994]

Semantic Parsing Example

IFTTT (If this then that)

This us dataset extracting a large number of recipes from if-this-then-that website[1].

Training Dataset	Development Dataset	Test Dataset
77495 instances	5171 instances	4294 instances



Archive your missed calls from Android to Google Drive

1. https://ifttt.com

[Quirk *et al.*, 2015]

Training Data

(sentence, implementation) pairs

What is the largest city in Hawaii? → answer(A, largest(A, city(A), loc(A, B), const(B, stateid(hawaii))))

What is the capital of California? → answer(A, capital(A), loc(A, B), const(B, stateid(california))))

Evaluation

- Exact-match accuracy of code
- Compare results of executing code on database

Traditional Semantic Parsing

- Rely on high-quality lexicons, manually-built templates, and features which are domain specific.

- Complex discrete learning algorithms

- Difficult to engineer: few people can do it and it takes a lot of time. (Examples annotated with semantics are expensive)

Neural Semantic Parsing

Any better idea for semantic parsing?

Can we treat the mapping from sentence to logical form as a machine translation problem?

Semantic Parsing vs. MT

Common	Difference
Both involve translating from one semantic representation into another.	But in machine translation, the target semantic representation is not machine-readable! Rather, it is human-readable.
Both involve complex structures, often related in complex ways.	MT has larger dataset. Semantic parsing is expensive to generate dataset ⇒ much smaller dataset
	Logical forms are more structured (so explicitly modeling the compositional structure could help)

Goals of Neural Semantic Parsing

- ❑ Reduce reliance on domain knowledge
- ❑ Use NNs to replace manually designed features
- ❑ Build a general-purpose parser: easy to adapt across domains and meaning representations

Language to Logical Form with Neural Attention

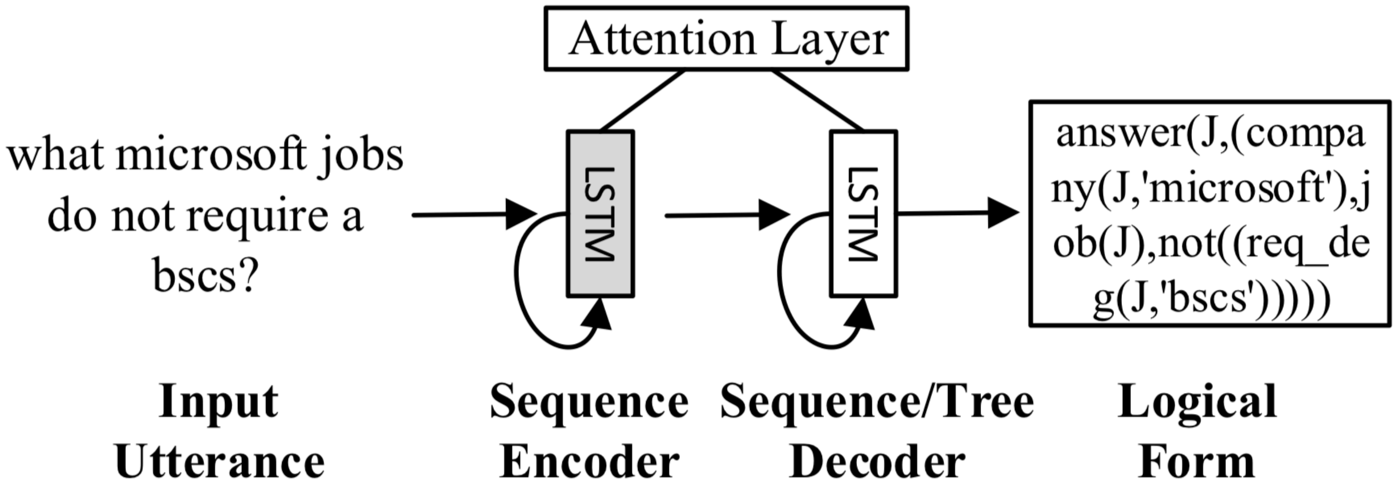
Li Dong and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
li.dong@ed.ac.uk, mlap@inf.ed.ac.uk

Problem Formulation

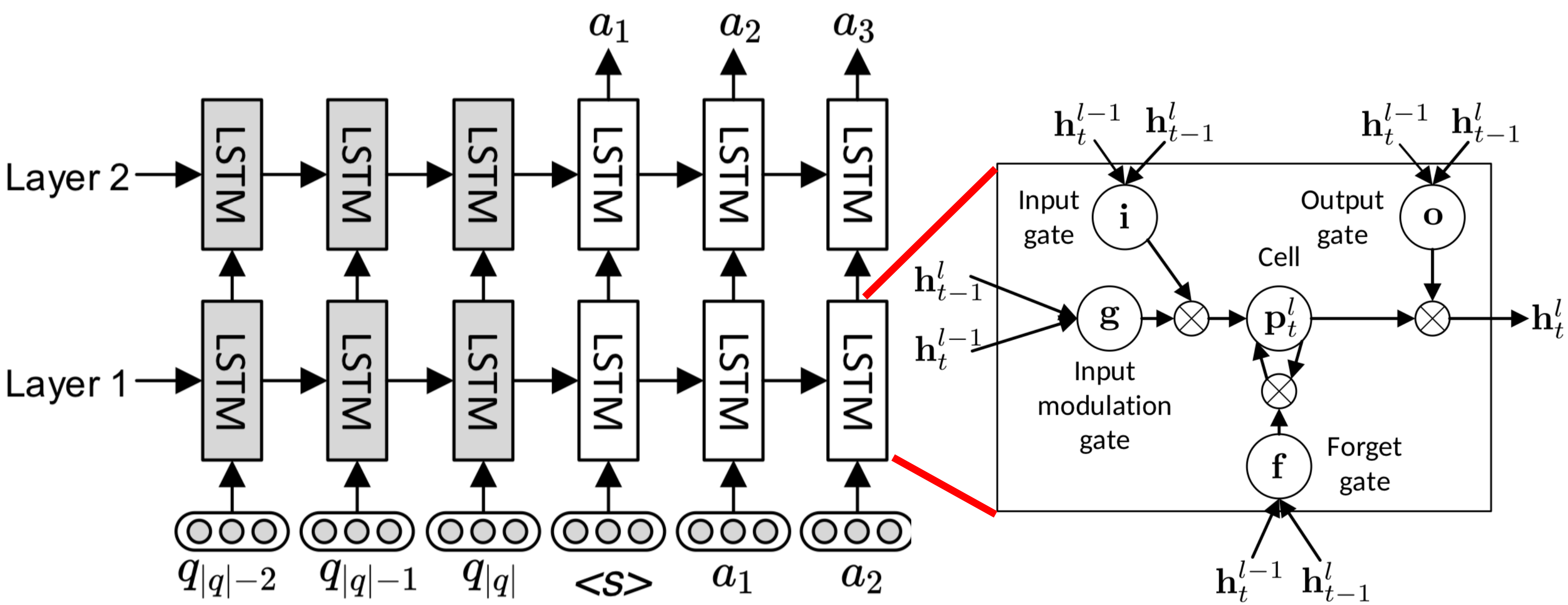
- Goal:
  Learn a model which maps natural language input $q = q_1, \ldots, q_{|q|}$ to a logical form representation of its meaning $a = a_1, \ldots, a_{|a|}$.

- Conditional probability: $p(a|q) = \prod_{t=1}^{|a|} p(a_t | a_{<t}, q)$

- Framework of Neural Semantic Parsing with Attention



Working Principle of Seq2Seq Neural Semantic Parsing



$$\mathbf{g}_t^0 = \mathbf{W}_q \mathbf{e}(q_t) \qquad \mathbf{h}_t^0 = \mathbf{W}_a \mathbf{e}(a_{t-1})$$

$$\mathbf{W}_q \in \mathbb{R}^{n \times |V_q|} \qquad \mathbf{W}_a \in \mathbb{R}^{n \times |V_a|}$$

$$p(a_t | a_{<t}, q) = \mathrm{softmax}_{a_t}(\mathbf{W}_o \mathbf{h}_t^l)$$

$$\mathbf{W}_o \in \mathbb{R}^{|V_a| \times n}$$

$$\mathbf{h}_t^l = f_{LSTM}(\mathbf{h}_{t-1}^l, \mathbf{h}_t^{l-1})$$

$$\mathbf{p}_t^l = f \odot \mathbf{p}_{t-1}^l + \mathbf{i} \odot \mathbf{g}$$

$$\mathbf{h}_t^l = \mathbf{o} \odot \tanh(\mathbf{p}_t^l)$$

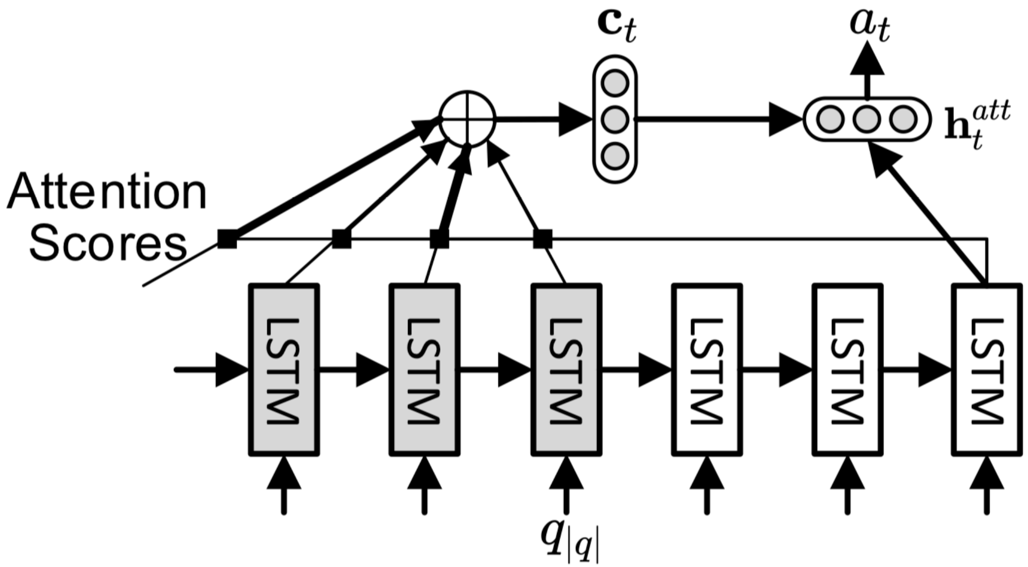
$$\begin{pmatrix} \mathbf{i} \\ \mathbf{f} \\ \mathbf{o} \\ \mathbf{g} \end{pmatrix} = \begin{pmatrix} sigm \\ sigm \\ sigm \\ tanh \end{pmatrix} W^l \begin{pmatrix} \mathbf{h}_t^{l-1} \\ \mathbf{h}_{t-1}^l \end{pmatrix}$$

Attention Mechanism for Neural Semantic Parsing

$$\mathbf{c}_t = \sum_{k=1}^{|q|} r_{t,k} \mathbf{h}_k^L$$

$$r_{t,k} \propto exp\{\mathbf{h}_t^L, \mathbf{h}_k^L\}$$

$$\sum_{j=1}^{|q|} r_{i,j} = 1$$



$\mathbf{h}_1^L, ..., \mathbf{h}_{|q|}^L$ are the top layer hidden vectors of the encoder

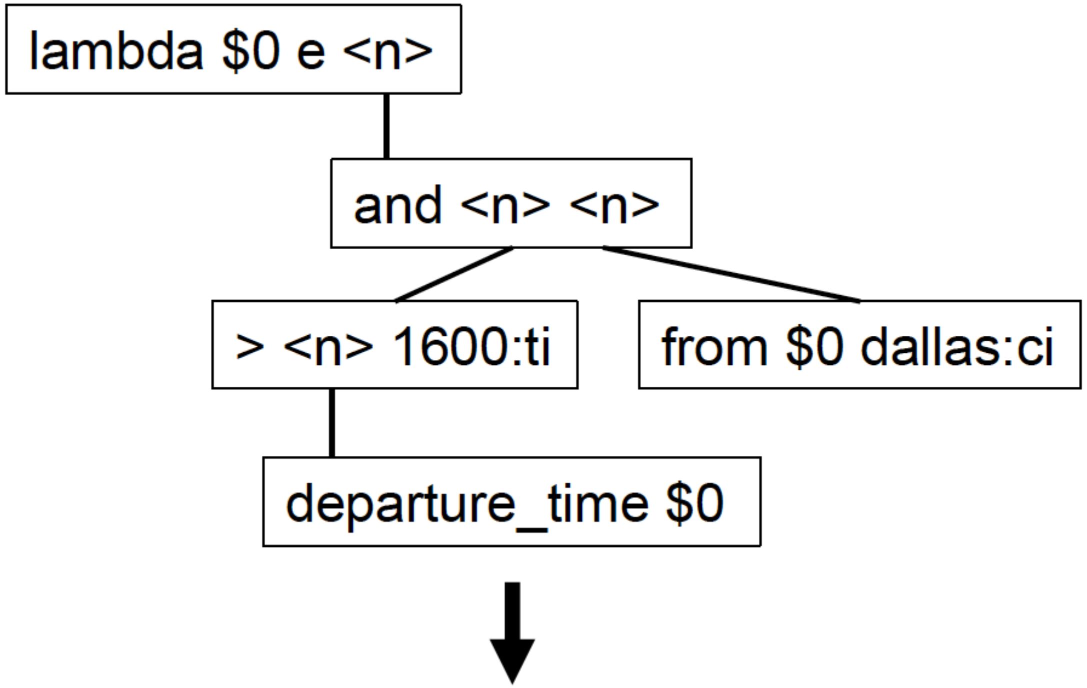
$$\mathbf{h}_t^{att} = \tanh(\mathbf{W}_1 \mathbf{h}_t^L + \mathbf{W}_2 \mathbf{c}_t)$$

$$p(a_t | a_{<t}, q) = \text{softmax}_{a_t}(\mathbf{W}_o \mathbf{h}_t^{att})$$

Drawbacks of Seq2Seq Model

Ignore the hierarchical structure of logical forms

Use ``( )“ to linearize logical form

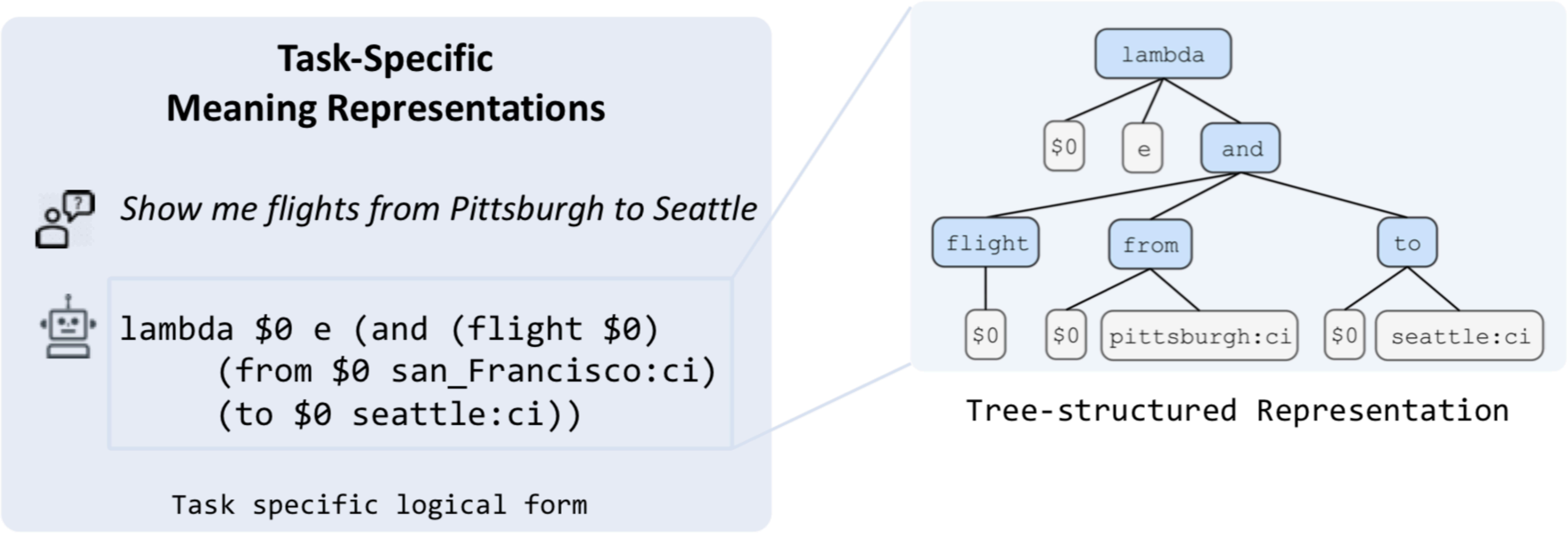


lambda $0 e (and (> (departure_time $0) 1600:ti) (from $0 dallas:ci))

Structure-aware Decoding for Semantic Parsing

Motivation:
Utilize the rich syntactic structure of target meaning representations

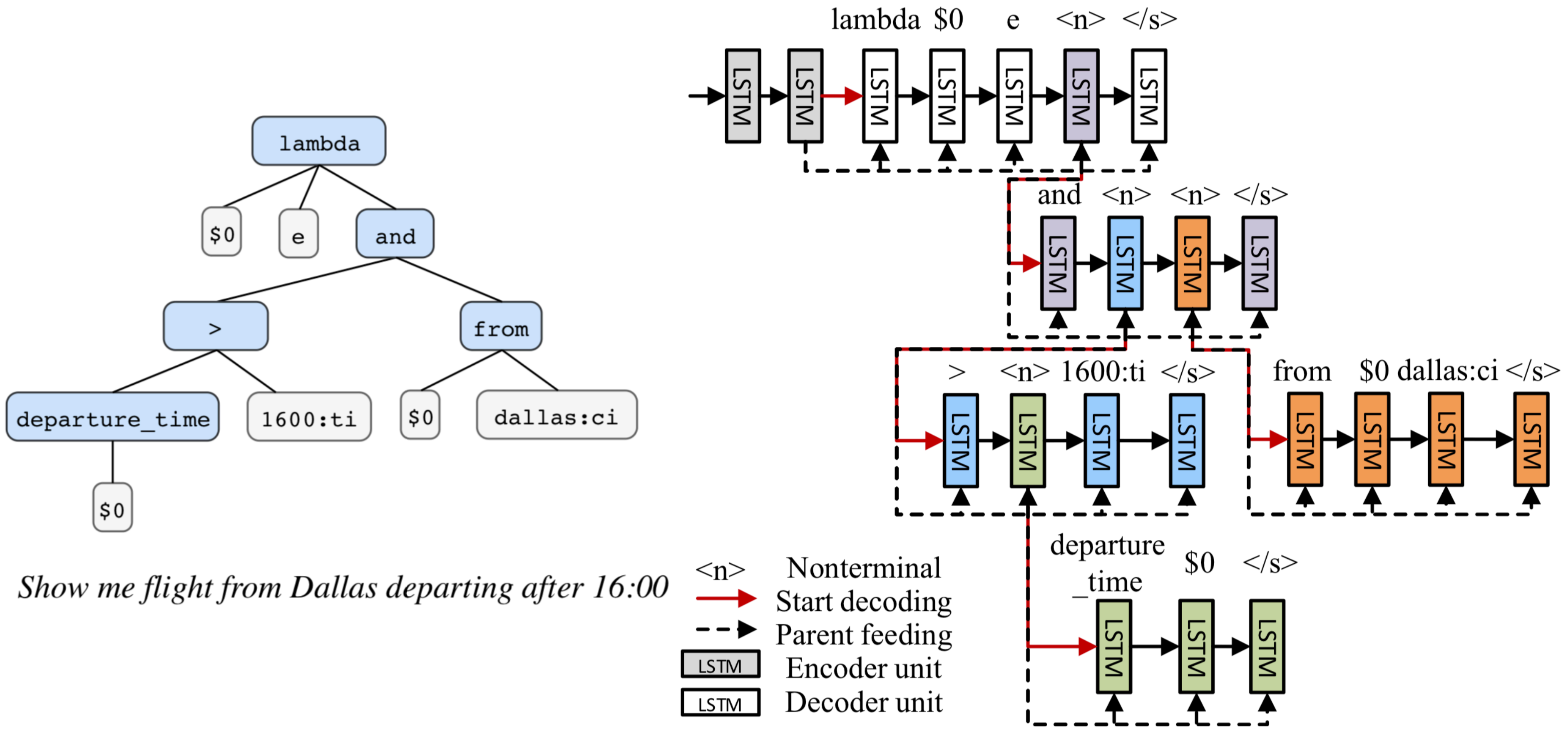
Seq2Tree:
Generate from top-down using hierarchical sequence-to-sequence model



Sequence to Tree Model (Seq2Tree)

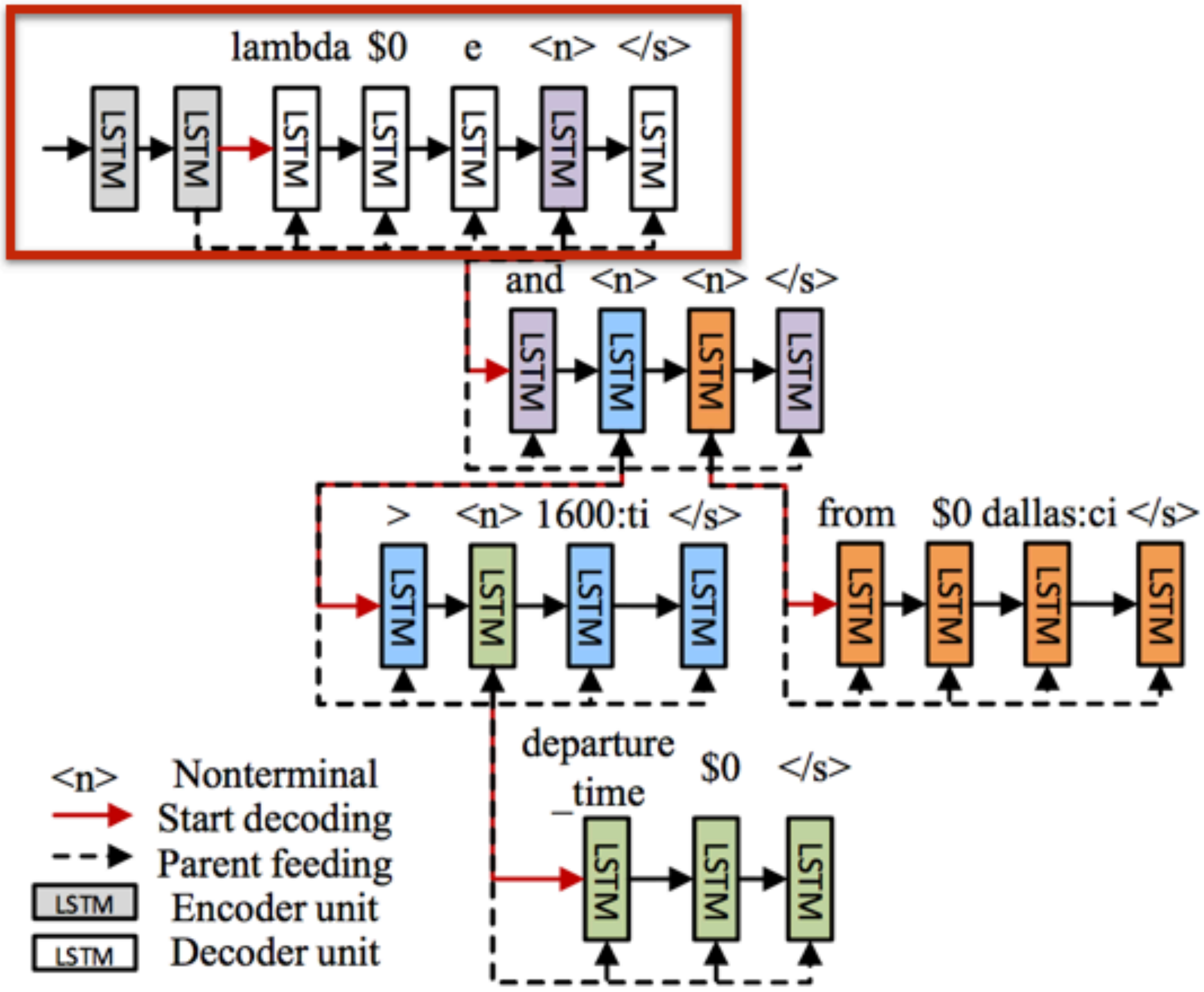
Sequence-to-tree Decoding Process:

– Each level of a parse tree is a sequence of terminals and nonterminals
– Use a LSTM decoder to generate the sequence
– For each nonterminal node, expand it using the LSTM decoder



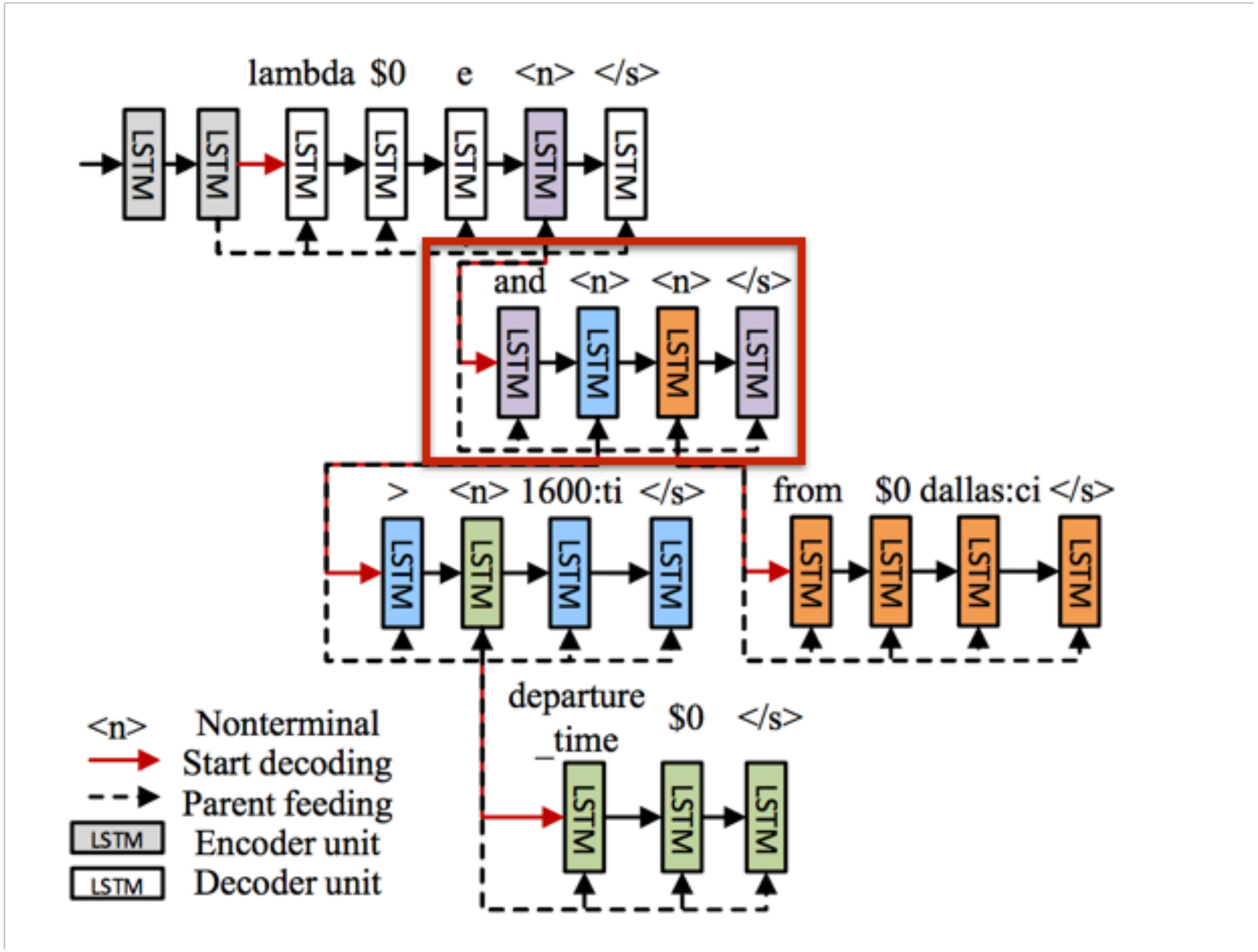
Flights from Dallas leaving after 4 in the afternoon

(lambda $0 e <n>)



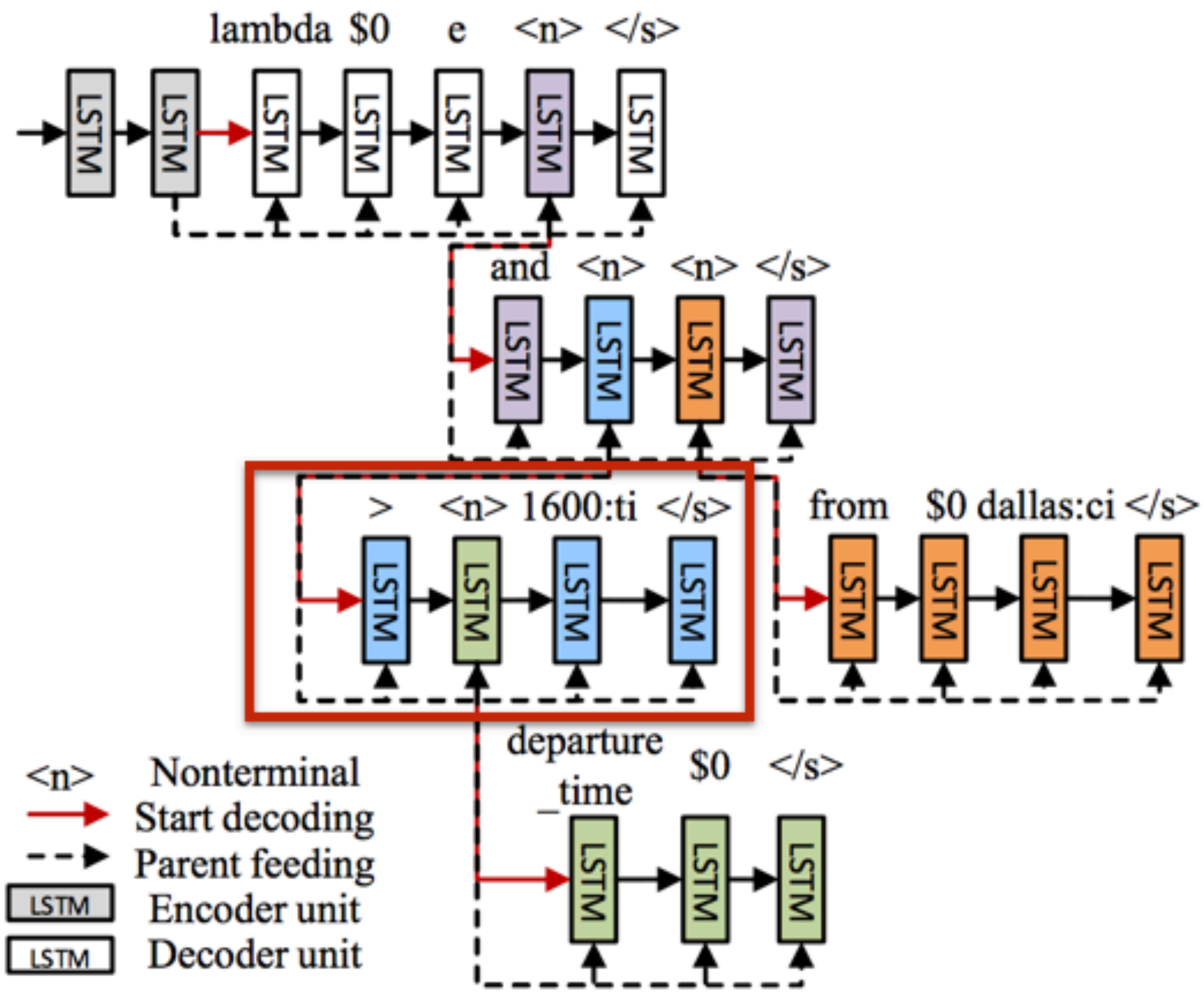
Flights from Dallas leaving after 4 in the afternoon

(lambda $0 e
(and **<n>** <n>))



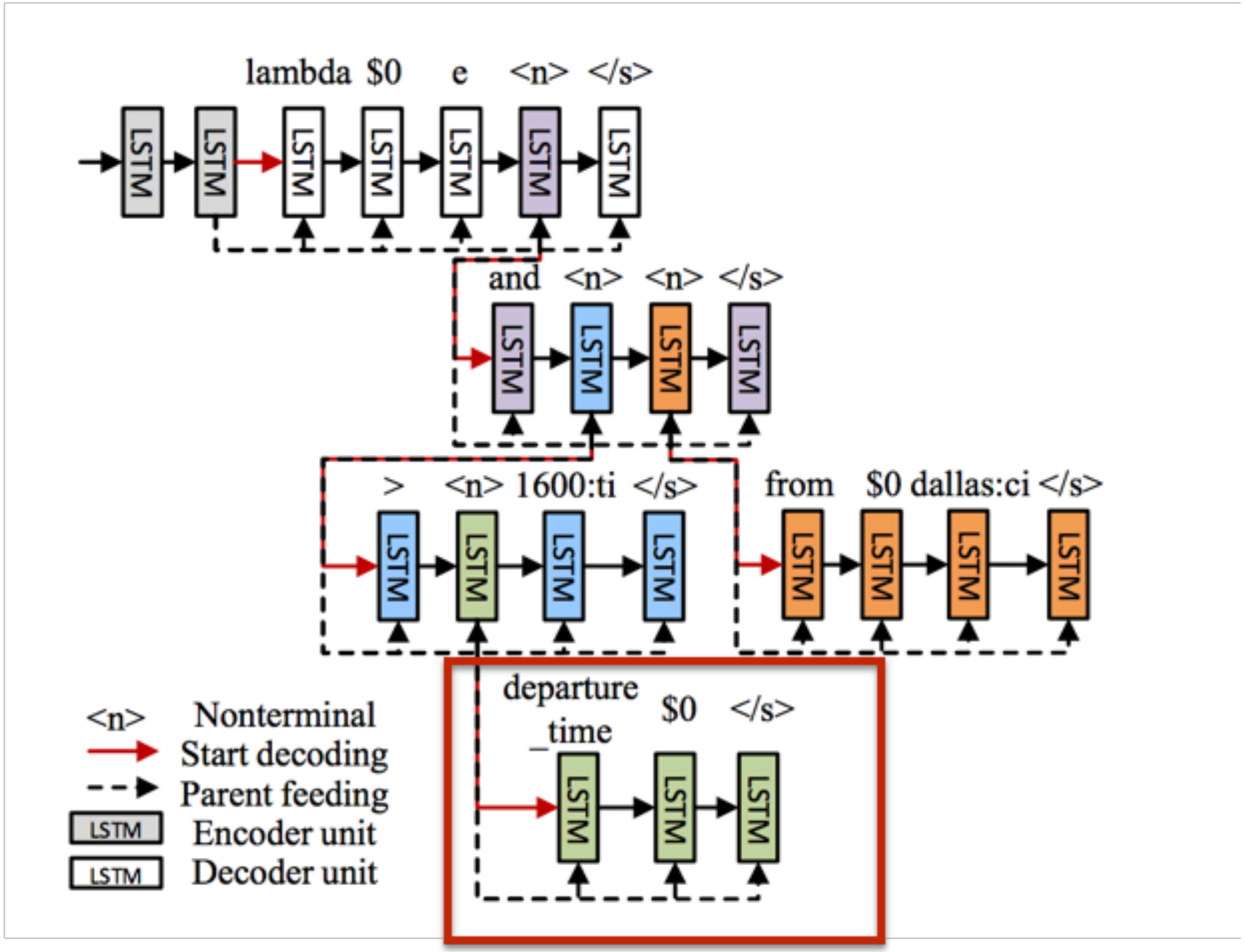
Flights from Dallas leaving after 4 in the afternoon

(lambda $0 e
  (and
    (> <n> 1600:ti)
    <n>))



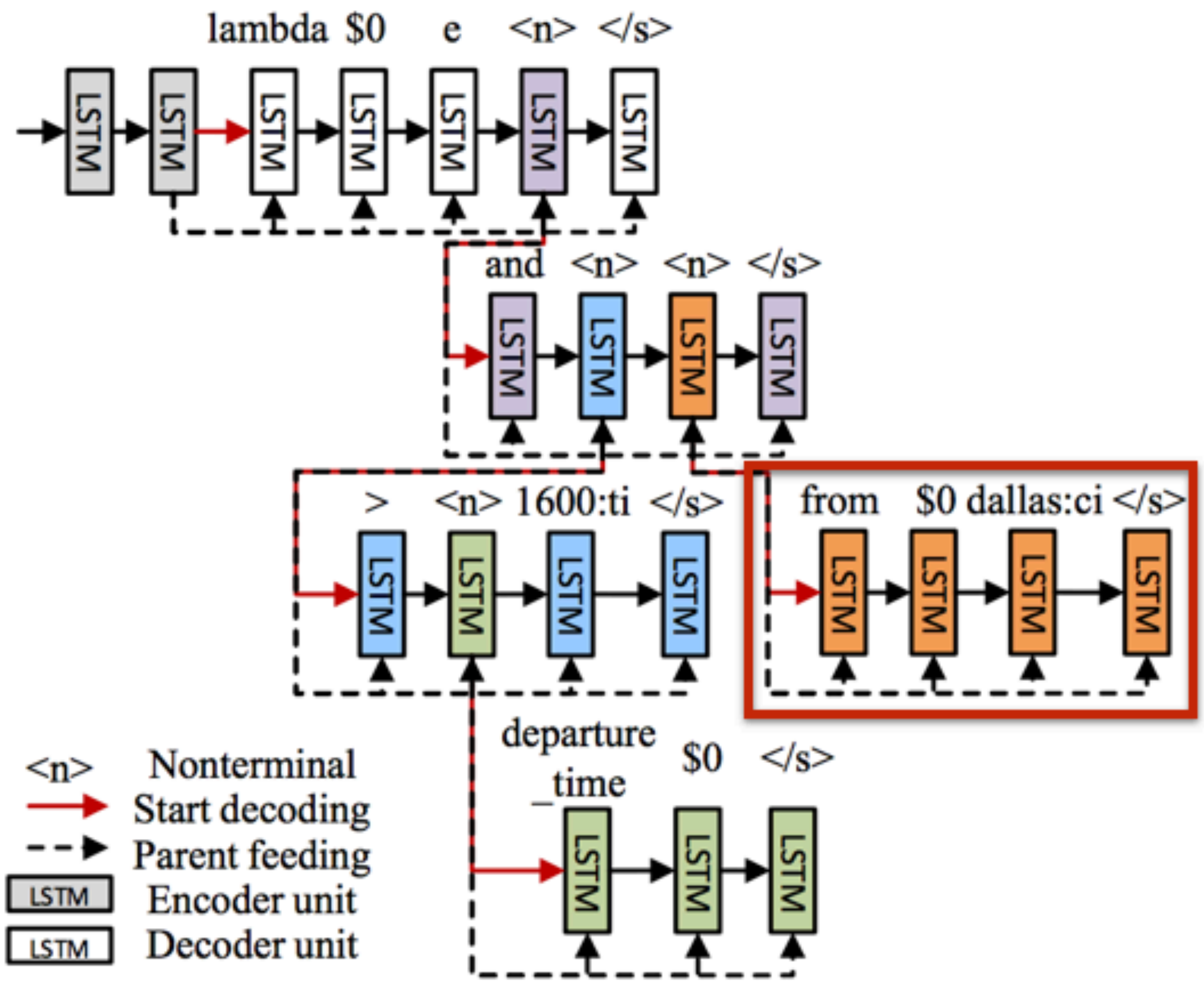
Flights from Dallas leaving after 4 in the afternoon

(lambda $0 e
  (and
    (> (departure_time $0) 1600:ti)
    <n>))



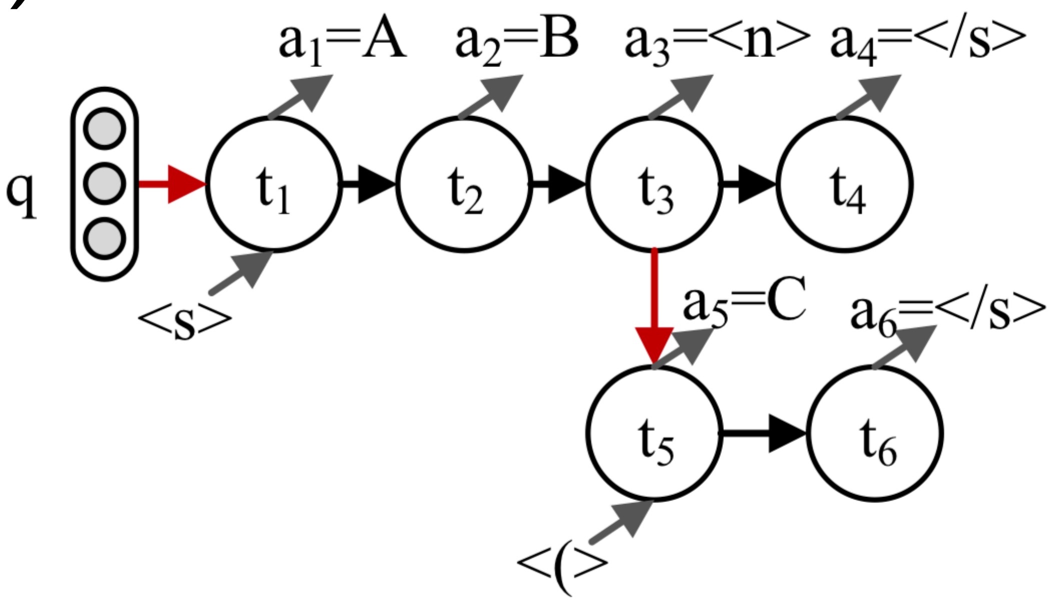
Flights from Dallas leaving after 4 in the afternoon

(lambda $0 e
  (and
    (> (departure_time $0) 1600:ti)
    (from $0 dallas:ci)))



Seq2Tree: Decoding Example

Logic form: ***A B (C)***



$$p(a|q) = p(a_1 a_2 a_3 a_4 | q)\, p(a_5 a_6 | a_{\leq 3}, q)$$

- Need not explicitly model matching parentheses
- Syntactically valid trees
- Allows parent feeding

Model Training

- Goal:

  Maximize the likelihood of the generated logical forms given natural language utterances as input

- Objective function:

$$\text{minimize} \sum_{(q,a)\in\mathcal{D}} -\log p(a|q)$$

Algorithm 1 Decoding for SEQ2TREE

Input: q: Natural language utterance
Output: $\hat{a}$: Decoding result

$\triangleright$ *Push the encoding result to a queue*
$Q.init(\{hid : \mathsf{SeqEnc}(q)\})$
$\triangleright$ *Decode until no more nonterminals*
while $(c \leftarrow Q.pop()) \neq \varnothing$ **do**
  $\triangleright$ *Call sequence decoder*
  $c.child \leftarrow \mathsf{SeqDec}(c.hid)$
  $\triangleright$ *Push new nonterminals to queue*
  for $n \leftarrow$ nonterminal in $c.child$ **do**
    $Q.push(\{hid : \mathsf{HidVec}(n)\})$
$\hat{a} \leftarrow$ convert decoding tree to output sequence

Inference

- At test time, we predict the logical form for an input utterance q by:

$$\hat{a} = \arg\max_{a'} p(a'|q)$$

- It is impractical to iterate over all possible results to obtain the optimal prediction. ⇒ Greedy search or beam search to generate tokens.

Argument Identification

- Motivation:

  Many NLUs contain entities or numbers and they are usually the arguments in logical form

  - Unavoidably rare
  - Do not appear in the training set at all

- Goal:

  Identify entities and numbers in input questions and replace them with their type names and unique IDs.

Argument Identification

- Procedure:
  1. Pre-processing training data:

     e.g. "***jobs with a salary of 40000***" → "***jobs with a salary of*** $\underline{\textbf{\textit{num}}_0}$"

     and its logical form: "***job(ANS), salary greater than(ANS, 40000, year)***"

     → "***job(ANS), salary greater than(ANS,*** $\underline{\textbf{\textit{num}}_0}$***, year)***"

  2. Perform training using pre-processed data

  3. At inference time, we also mask entities and numbers with their types and IDs.

  4. Once we obtain the decoding result, a post-processing step recovers all the markers $\underline{\textbf{\textit{type}}_i}$ to their corresponding logical constants.

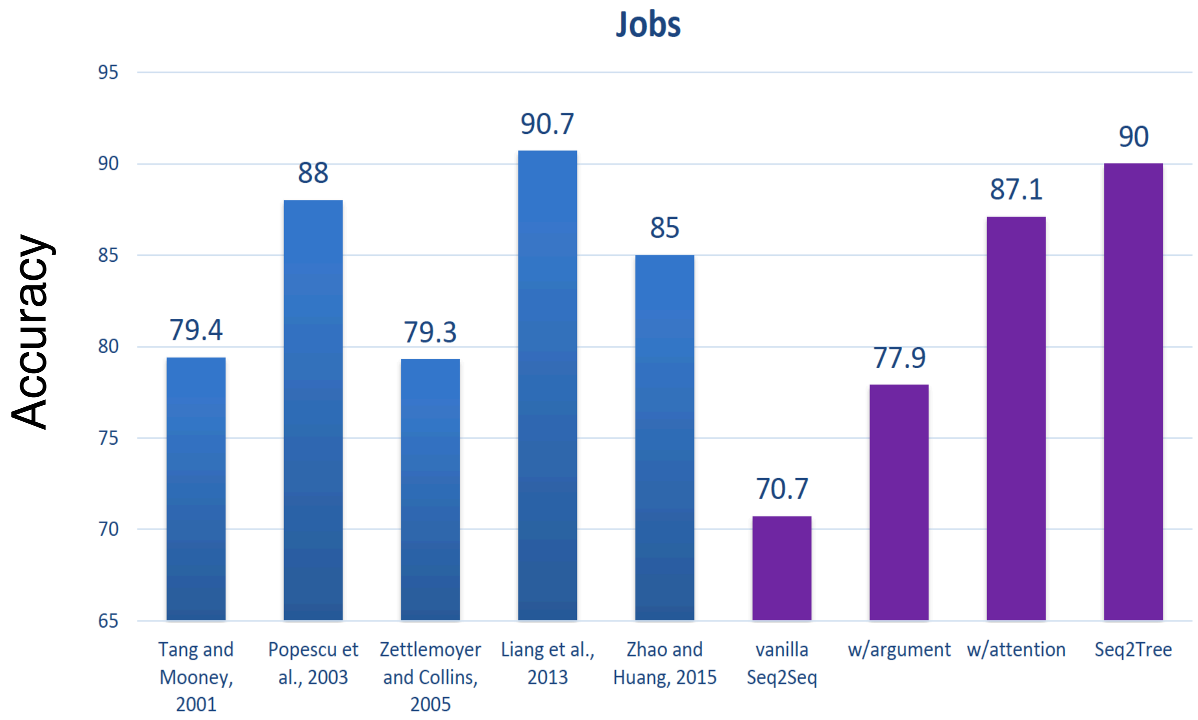
Evaluation Metrics

- For ***GEO, JOBS, ATIS***, accuracy is defined as:

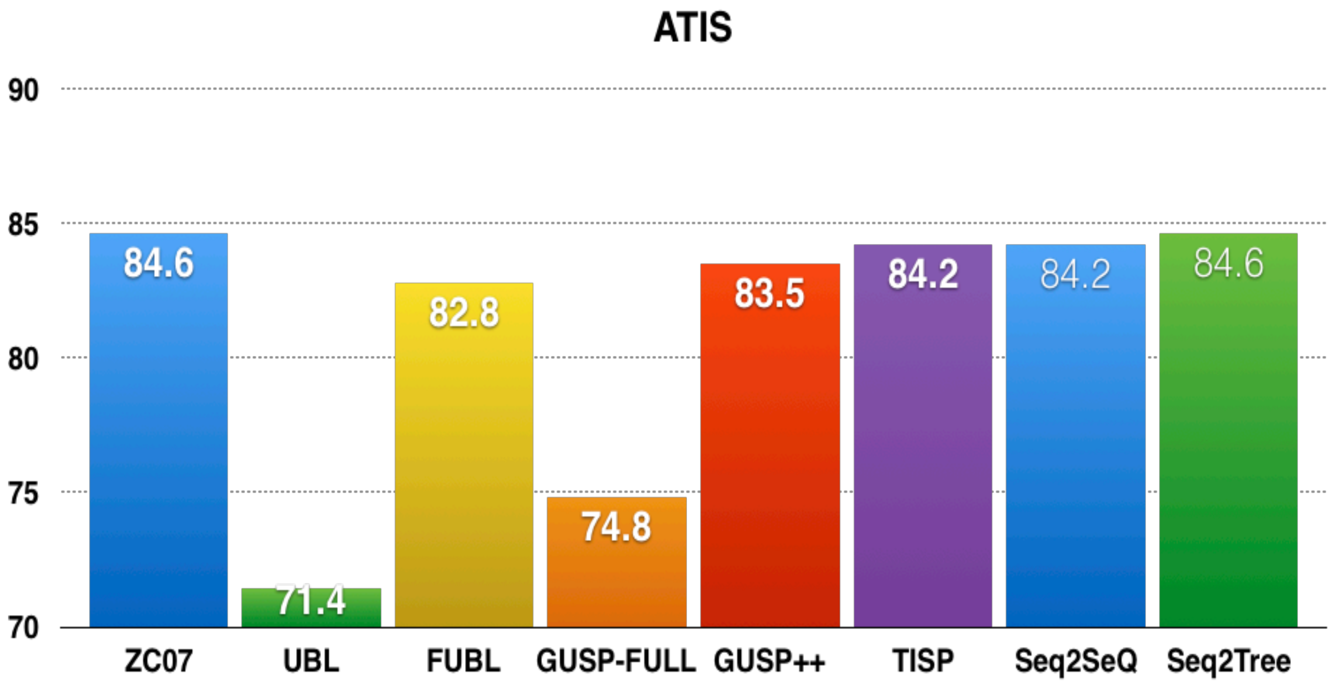
$$\frac{\text{sentences that are correctly parsed to gold standard}}{\text{total number of sentences}}$$

- For ***IFTTT***, dataset is extremely noisy and measuring accuracy is problematic. Consider three metrics: the accuracy of correct "**channels**", the accuracy of "**channels+funcs**", and F1 score.

Results

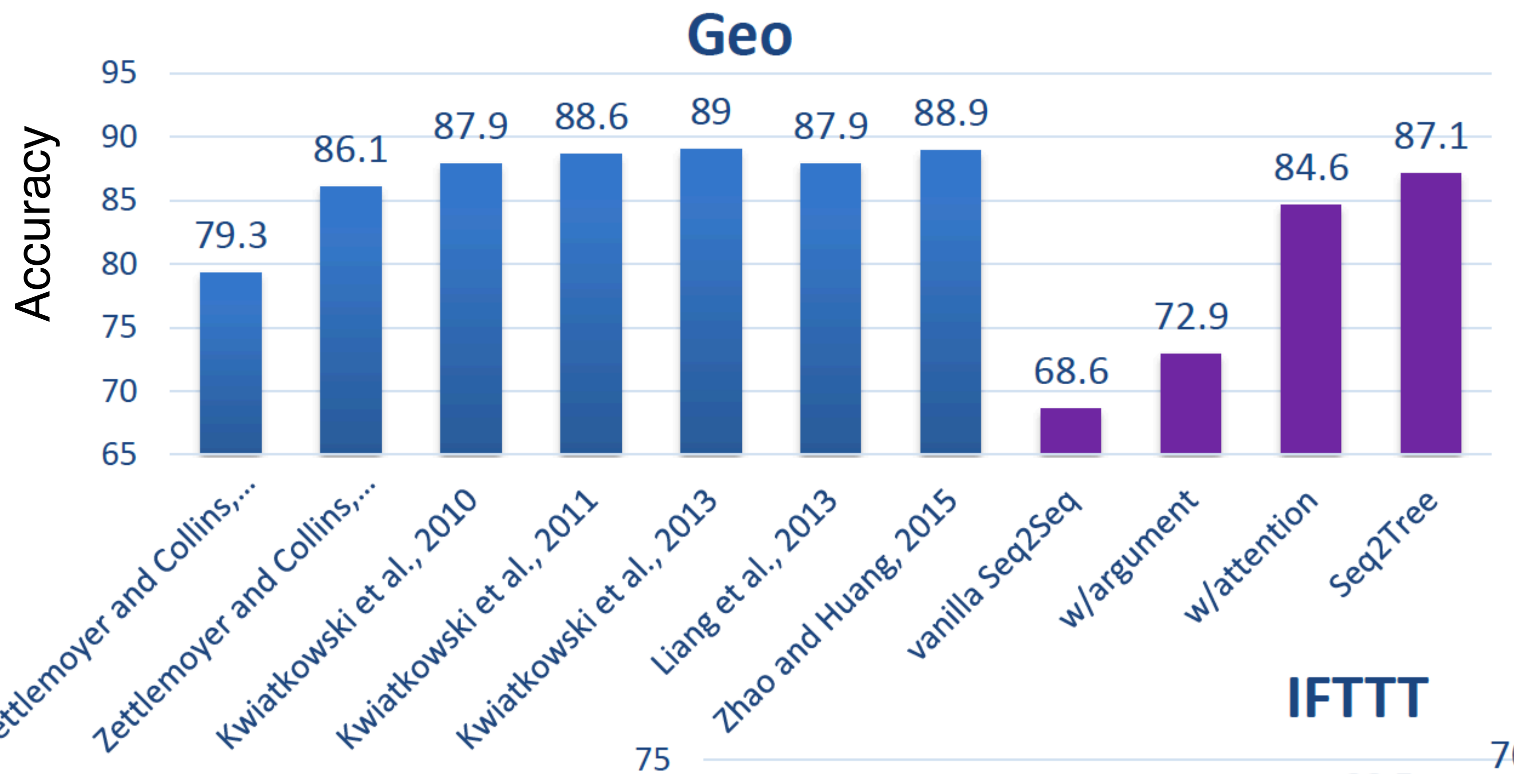


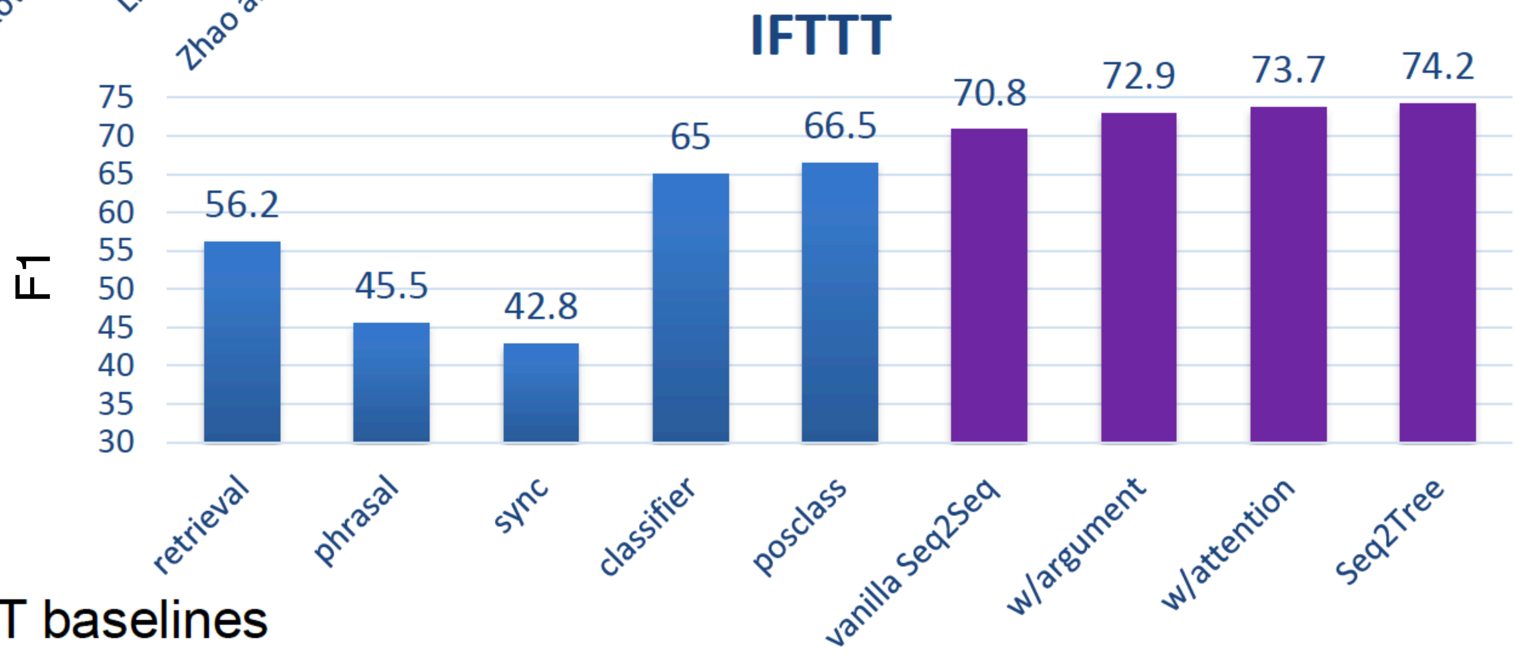
Jobs
Accuracy
95
90
85
80
75
70
65
79.4
88
79.3
90.7
85
70.7
77.9
87.1
90
Tang and Mooney, 2001
Popescu et al., 2003
Zettlemoyer and Collins, 2005
Liang et al., 2013
Zhao and Huang, 2015
vanilla Seq2Seq
w/argument
w/attention
Seq2Tree



ATIS
90
85
80
75
70
84.6
71.4
82.8
74.8
83.5
84.2
84.2
84.6
ZC07
UBL
FUBL
GUSP-FULL
GUSP++
TISP
Seq2SeQ
Seq2Tree

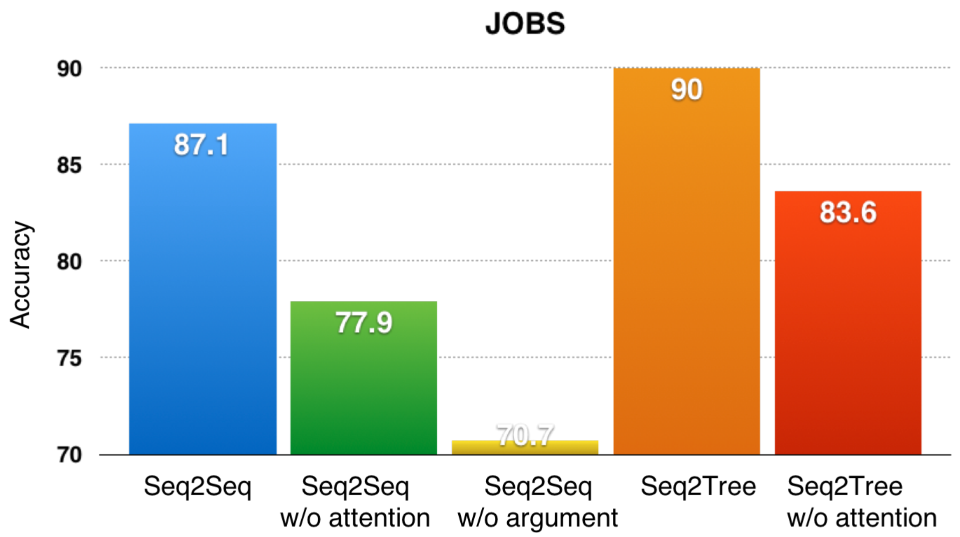
Results

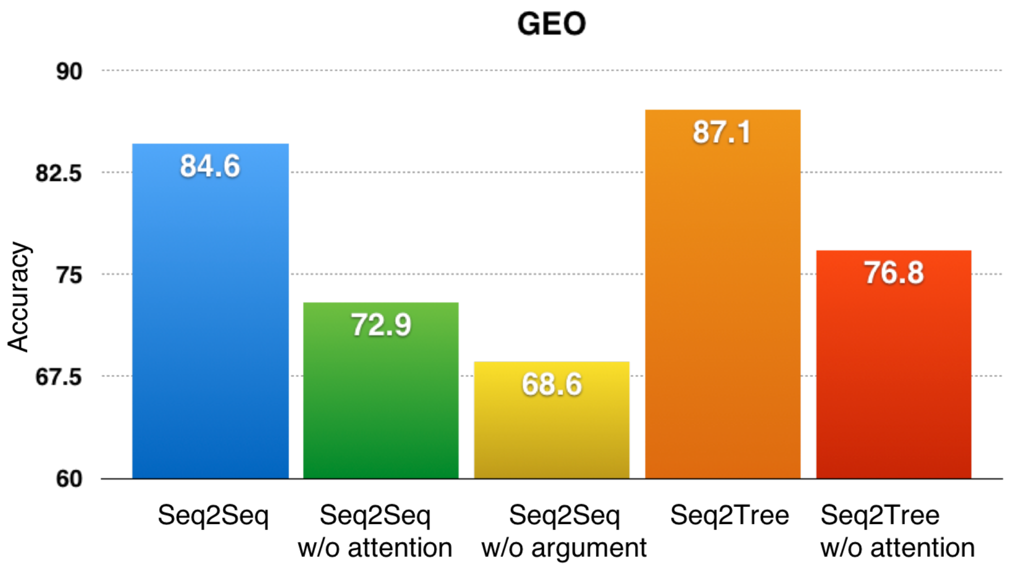


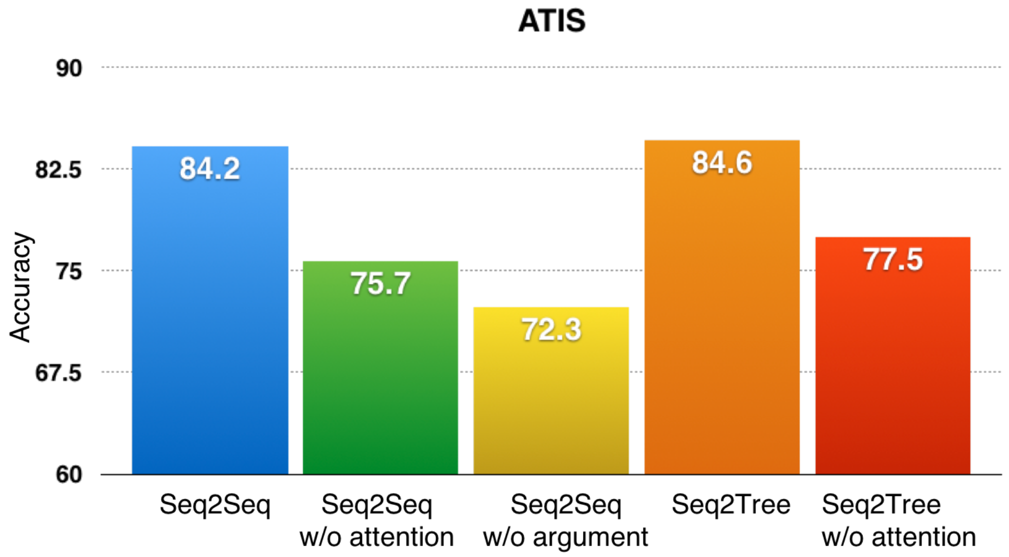


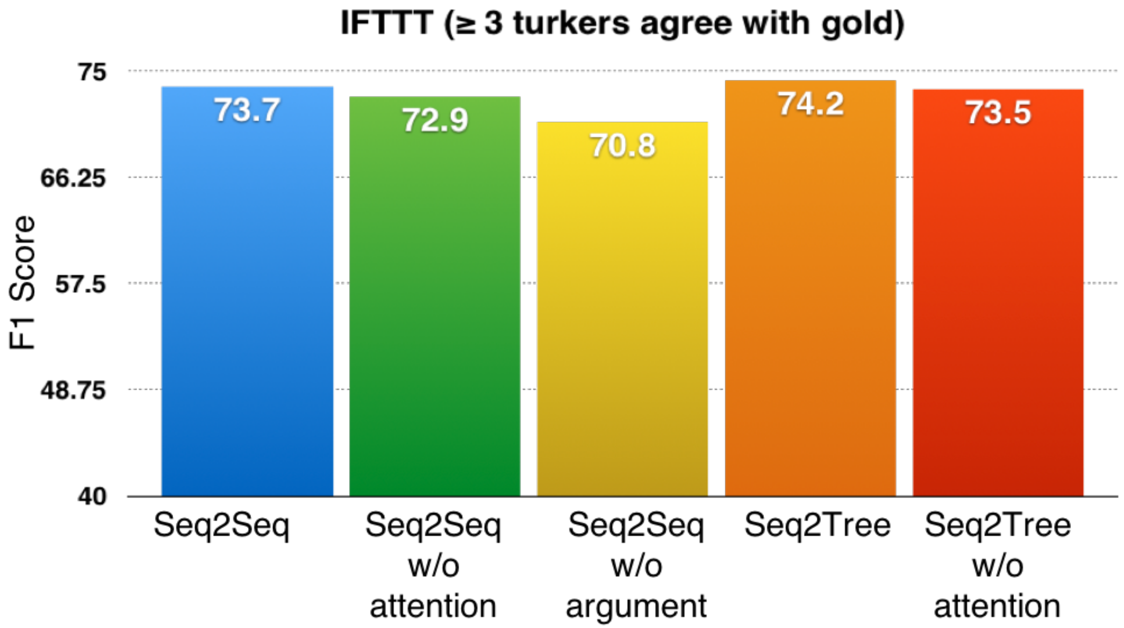
(Quirk et al., 2015): IFTTT baselines

Ablation Study

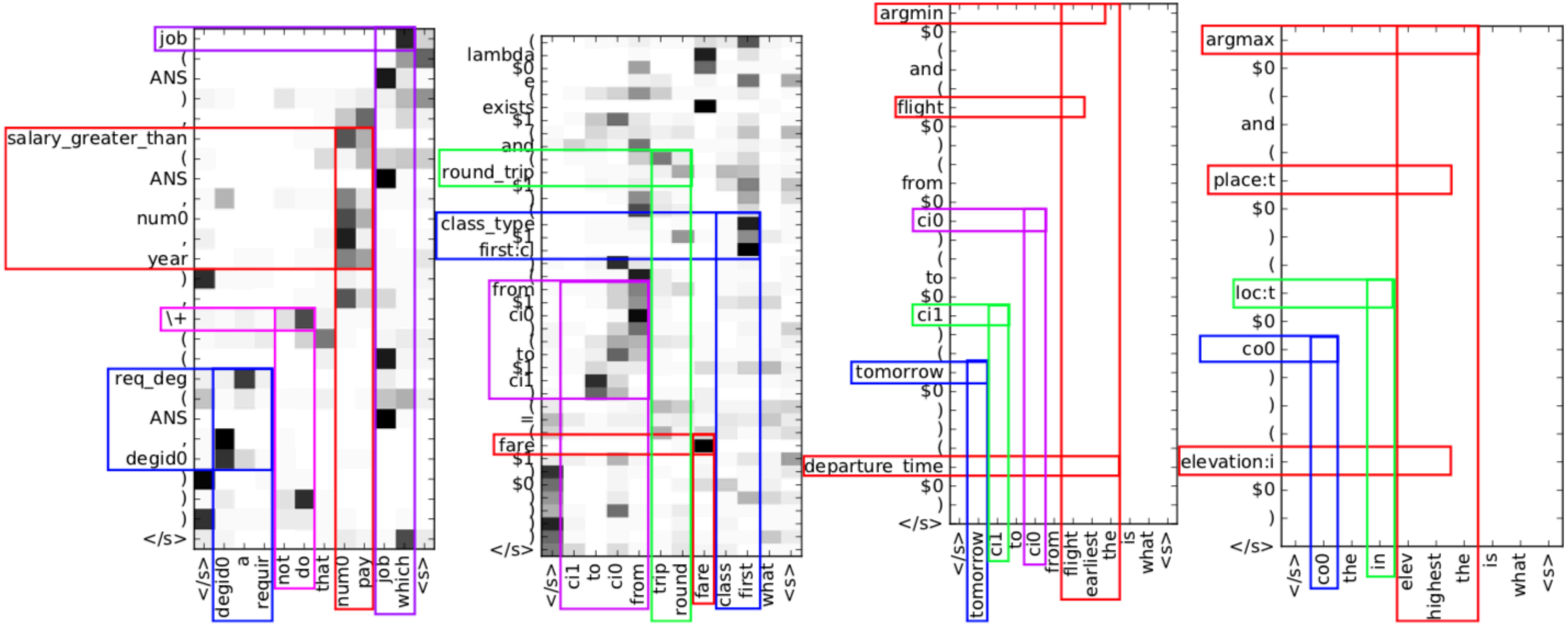








Attention helps!



(a) which jobs pay num0 that do not require a degid0

(b) what's first class fare round trip from ci0 to ci1

(c) what is the earliest flight from ci0 to ci1 tomorrow

(d) what is the highest elevation in the co0

Figure 6: Alignments (same color rectangles) produced by the attention mechanism (darker color represents higher attention score). Input sentences are reversed and stemmed. Model output is shown for SEQ2SEQ (a, b) and SEQ2TREE (c, d).

Result Summary

- ❑ Overall, Seq2Tree is superior to Seq2Seq.

- ❑ Adding attention substantially improves performance on all three datasets.

- ❑ Argument identification is critical for small scale datasets.

Error Analysis

❑ Under-Mapping

Doesn’t take alignment history into consideration

❑ Argument Identification

- Some mentions are incorrectly identified as arguments, e.g. ***may*** is sometimes identified as a month when it is simply a modal verb
- Some argument mentions are ambiguous, e.g. 6 o’clock can be used to express either 6 am or 6 pm

❑ Rare Words:

Some question words are rare in the training set, which makes it hard to estimate reliable parameters for them.

Conclusions

- ❑ This paper presented an encoder-decoder neural network model for mapping natural language descriptions to their meaning representations

- ❑ Encode natural language utterances into vectors and generate their corresponding logical forms as sequences or trees using recurrent neural networks with LSTM units

- ❑ Experimental results show that enhancing the model with a hierarchical tree decoder and an attention mechanism improves performance across the board

So far, we only consider parsing NLUs independently.

But how to parse context-dependent sentences?

➡ Semantic Parsing for Context-Dependent Sentences

Learning to Map Context-Dependent Sentences to Executable Formal Queries

Goal:

- Language understanding in **long sequence** of interactions
- Learn to translate human utterance to executable queries
- **Use contextual information**

Background Study

Related works

- "Learning to Parse Database Queries Using Inductive Logic Programming" (Zelle, Mooney, 1996)
- "Language to Logical Form with Neural Attention" (Dong, Lapata, 2016)

- Single turn interaction

Background Study

Context-Dependent Prior research work

- SCONE (Long et al. 2016): only focus on specific interaction phenomena

- ATIS (Zettlemoyer and Collins 2009): use different representations, and require extra training and annotation of data.

A Case Study:

Query a database with natural Language

A Case Study

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

A Case Study

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

A Case Study

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))));
```

A Case Study

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

Result `Found 31 Flights:` ✈️✈️✈️✈️✈️✈️✈️✈️✈️✈️✈️✈️✈️✈️

A Case Study

User *On American Airlines*

Result `Found 2764 Flights:`

A Case Study

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

Result `Found 31 Flights:`

User *On American Airlines*

Result `Found 5 Flights:`

A Case Study

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.`

A Case Study

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:`

Challenges

- For long history of interactions:
  - Relevant but elided information was mentioned many turns before
  - User may change focus during interaction

A Case Study

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*

Key Element

- Implicit mechanism for carrying information from beginning to end of interaction

Key Element

- Implicit mechanism for carrying information from beginning to end of interaction

- **Interaction History** dependency
    - Previous user request (**natural language form**)
    - Previous generated queries (**SQL form**)

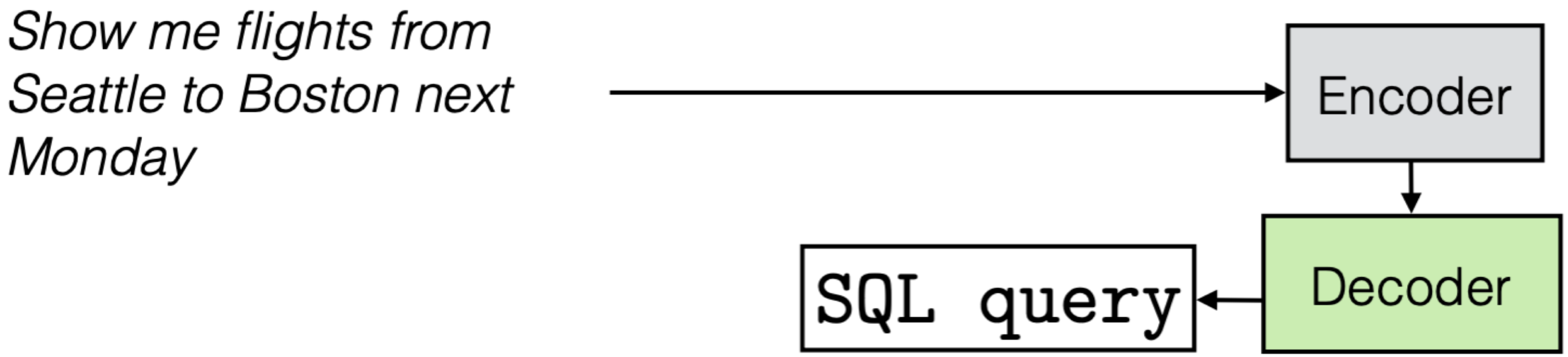
Solutions

Solutions Ingredients:

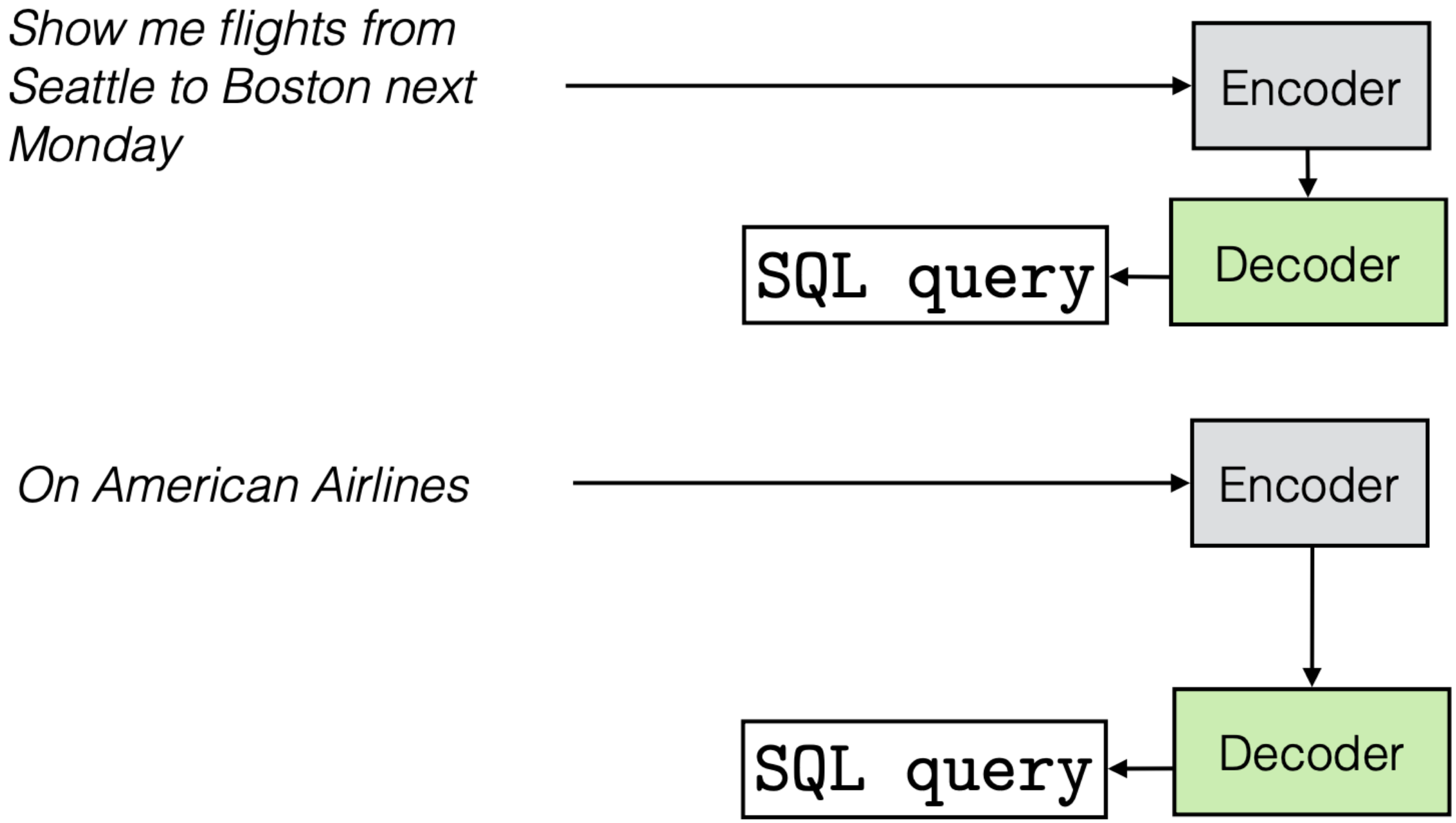
a) Incorporate previous request

b) Incorporate previous queries

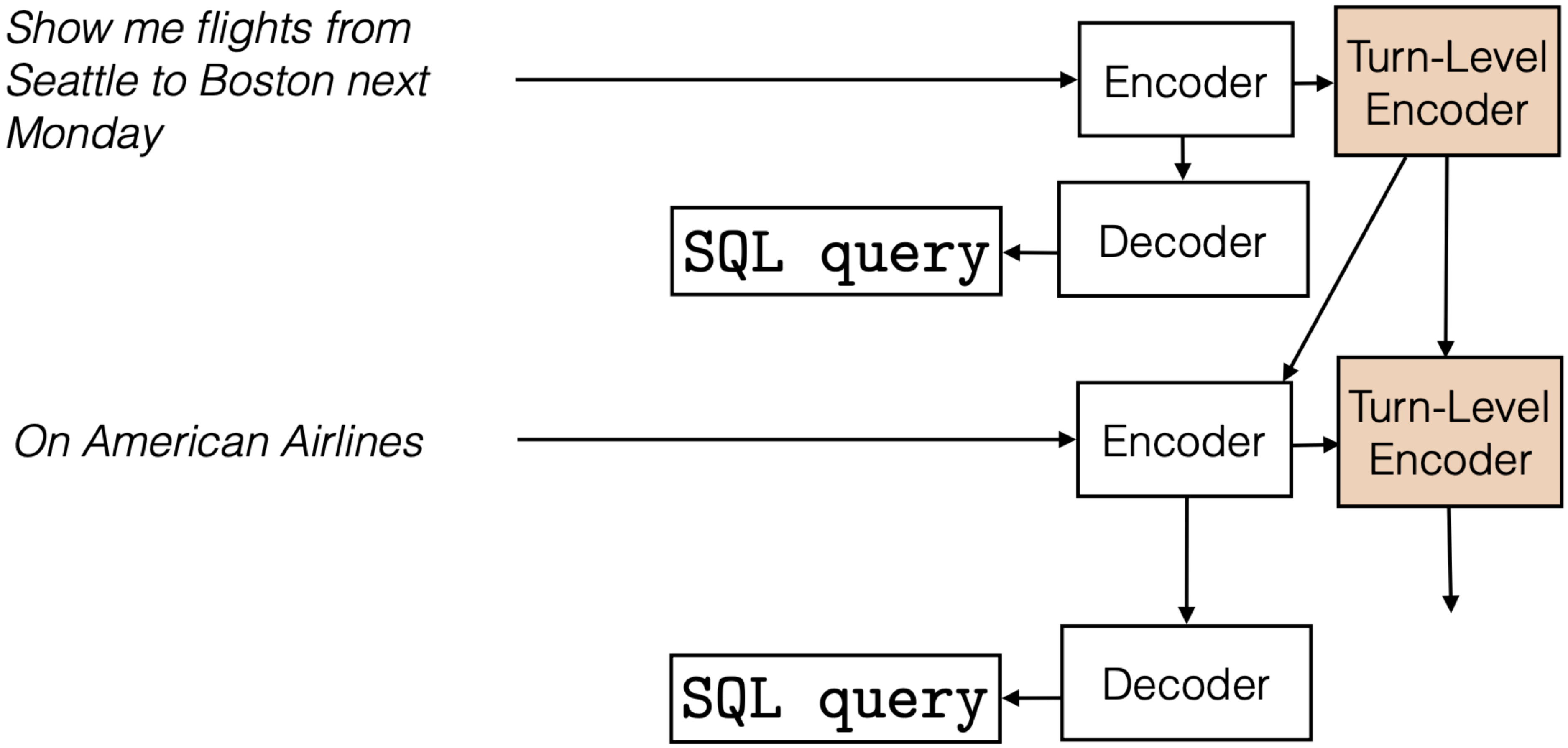
Idea


Show me flights from
Seattle to Boston next
Monday
Encoder
Decoder
SQL query

Idea

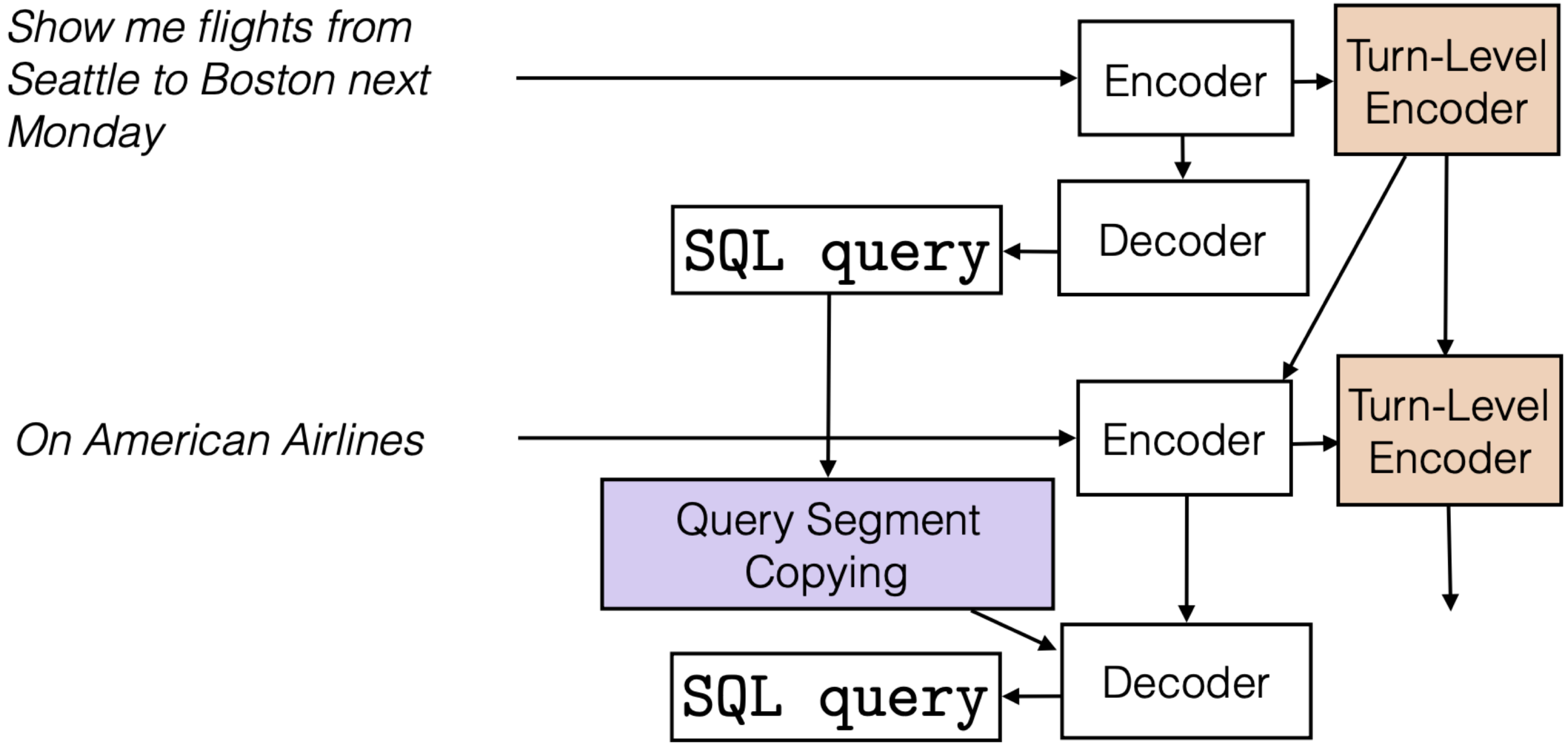

Show me flights from Seattle to Boston next Monday
Encoder
Decoder
SQL query
On American Airlines
Encoder
Decoder
SQL query

Idea



Mechanism 1 **Previous Requests: Turn-level Encoder**

Idea



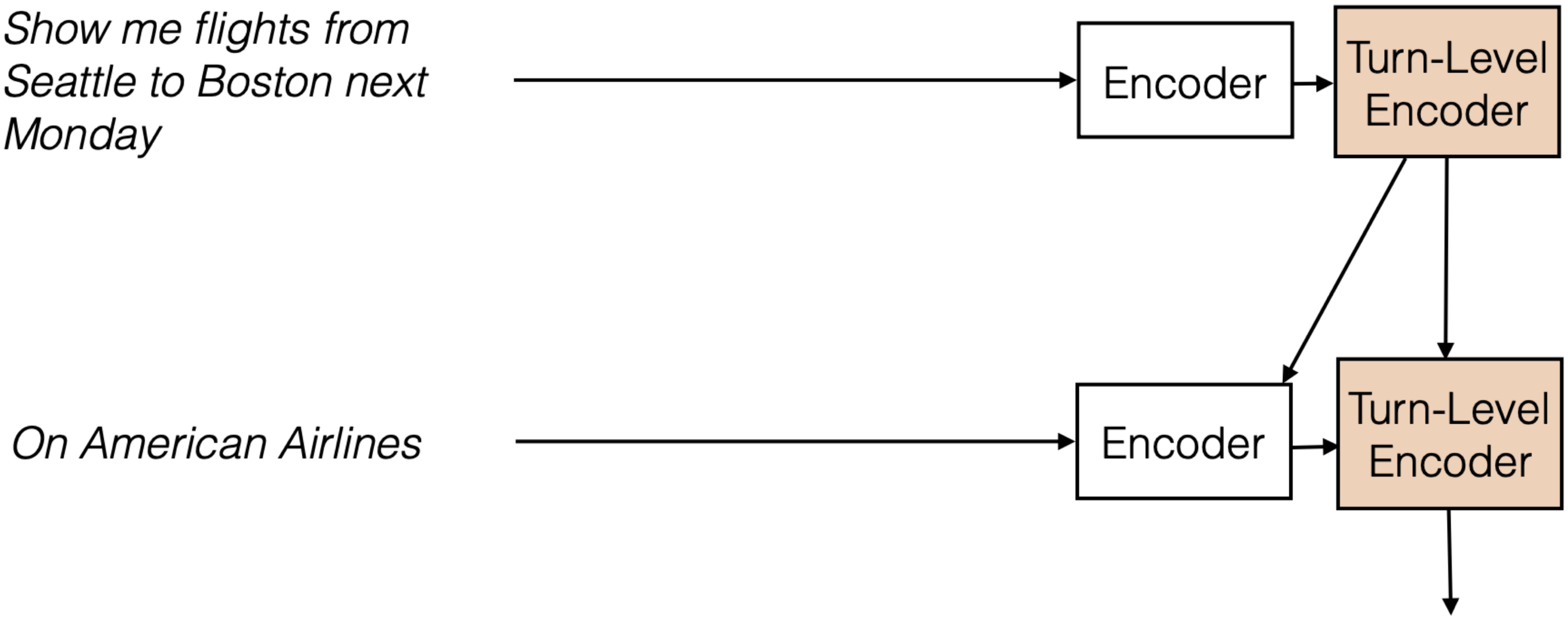
Mechanism 1 **Previous Requests: Turn-level Encoder**

Mechanism 2 **Previous Queries: Query Segment Copying**

Mechanism #1:

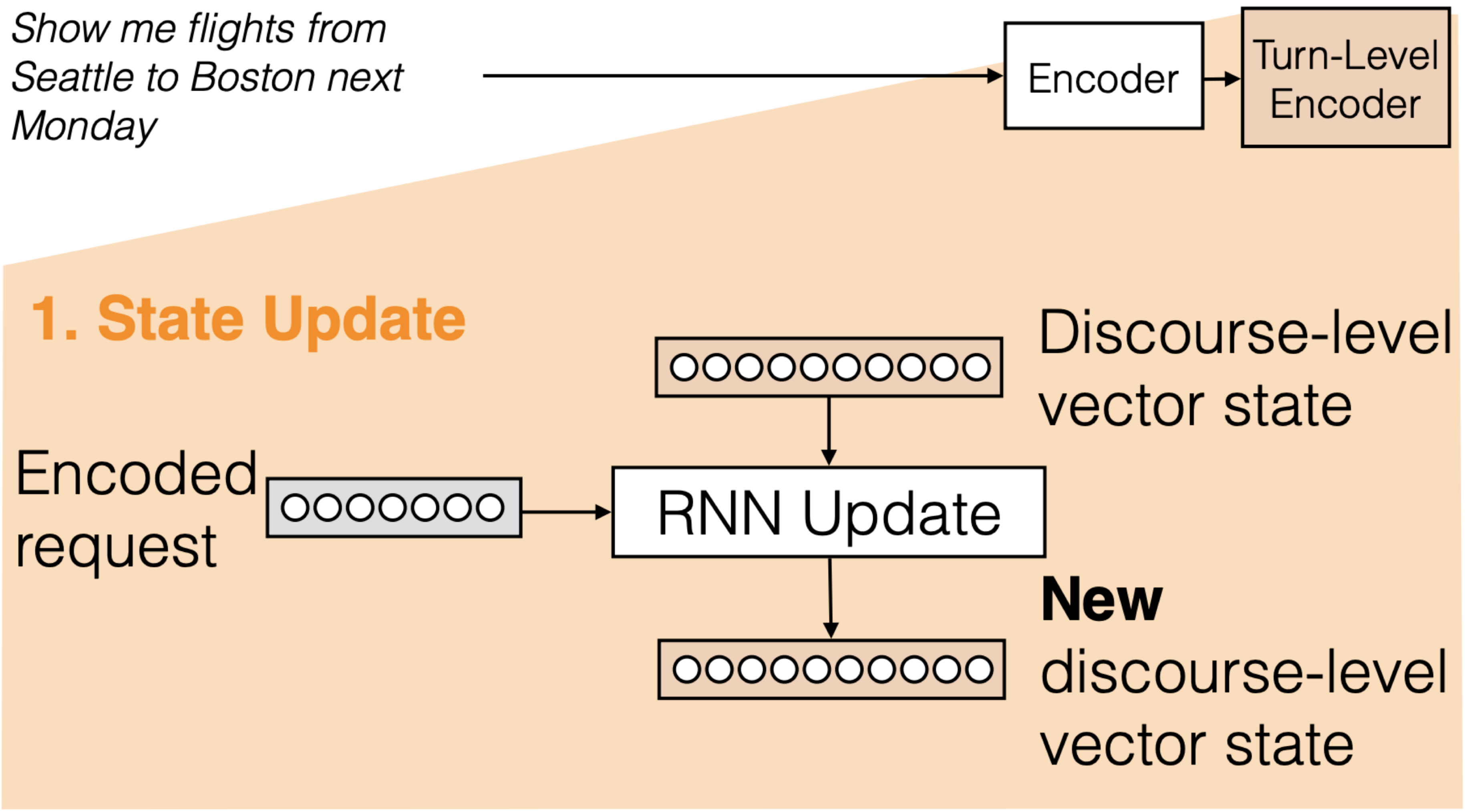
Incorporating Previous Request

Incorporating Previous Requests

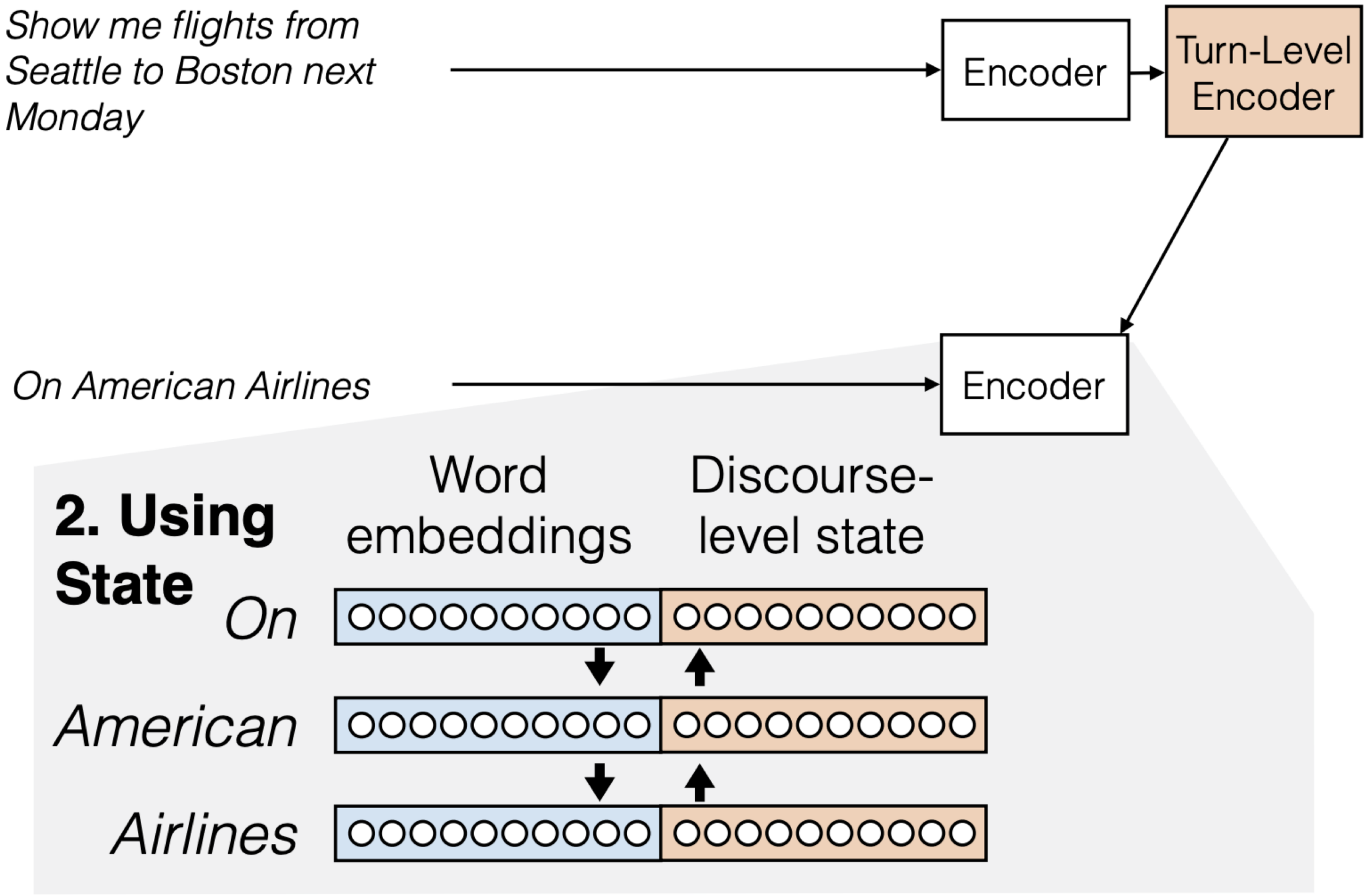


Mechanism 1 **Previous Requests: Turn-level Encoder**

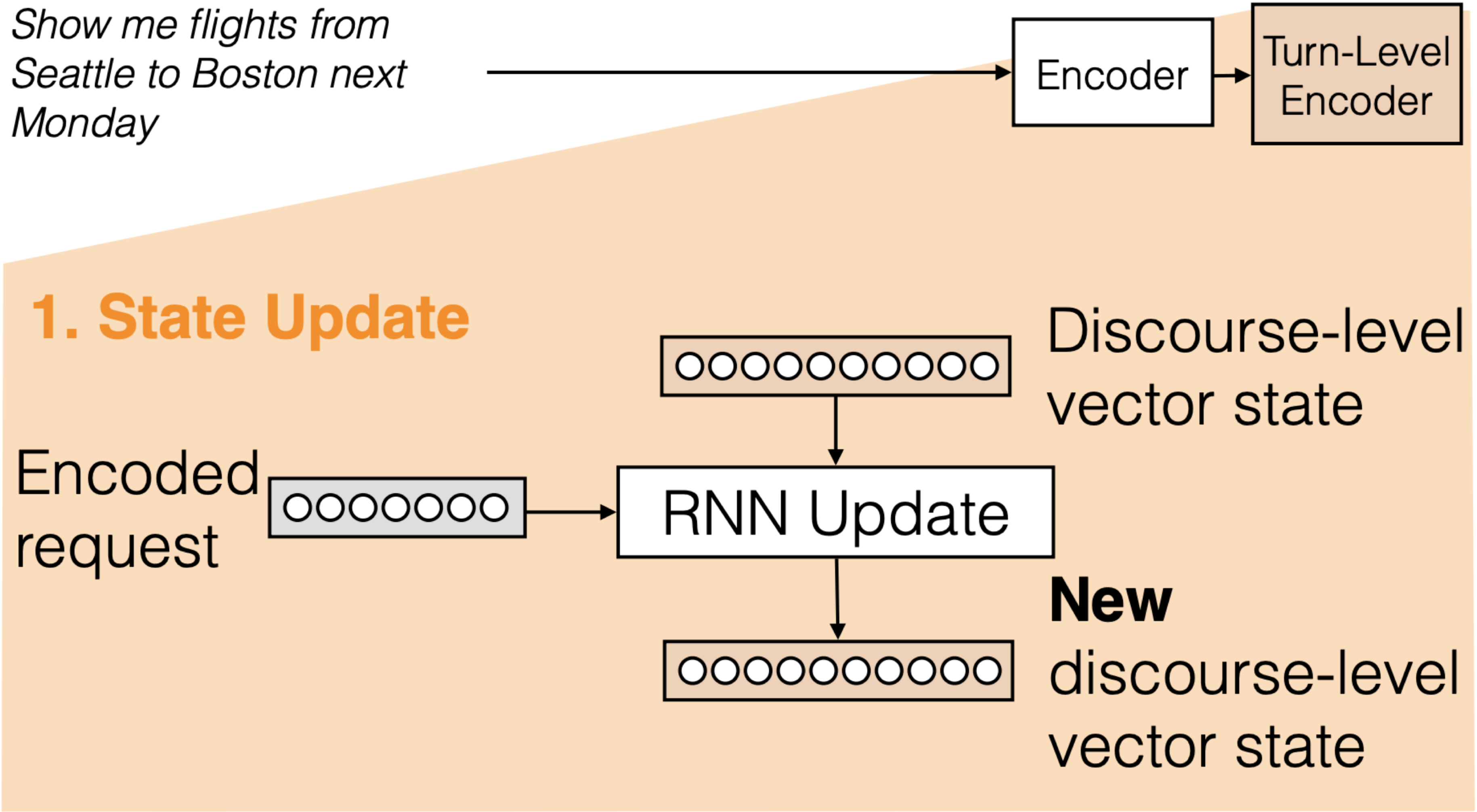
Incorporating Previous Requests



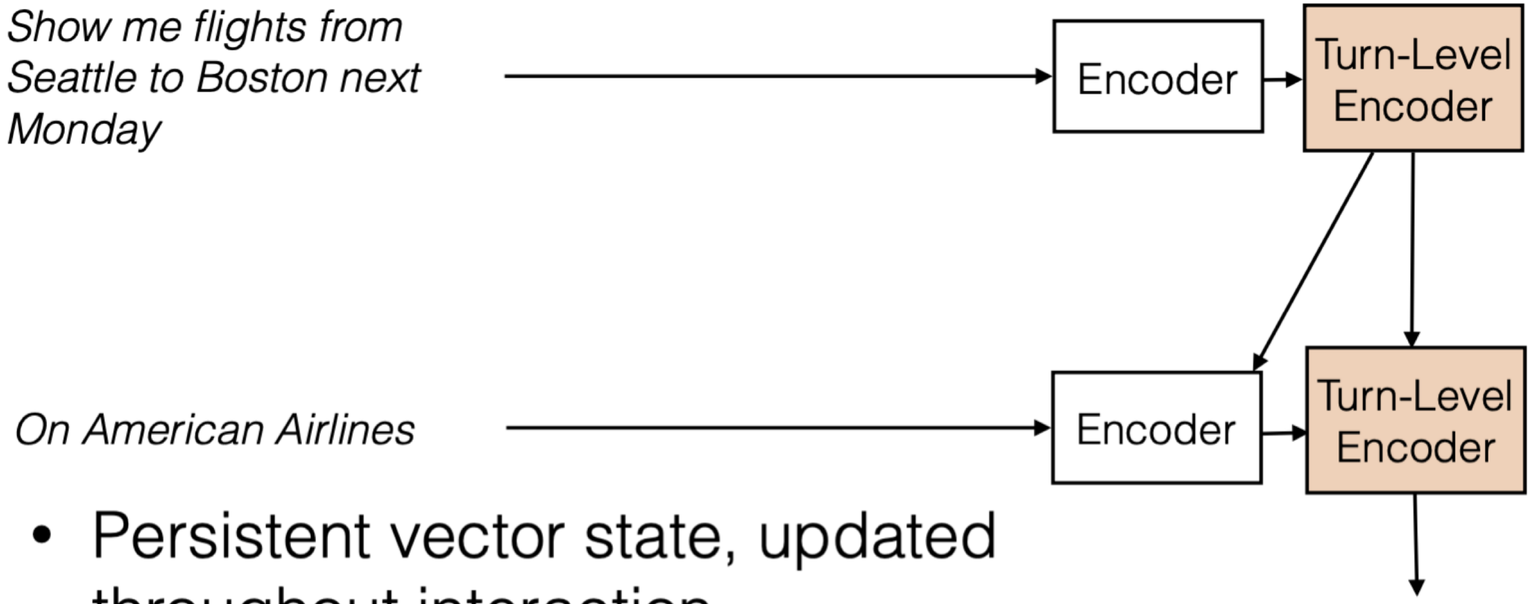
Incorporating Previous Requests



Incorporating Previous Requests



Incorporating Previous Requests



- Persistent vector state, updated throughout interaction
- Encode information from beginning to end of interaction
- Completely learned

Incorporating Previous Requests

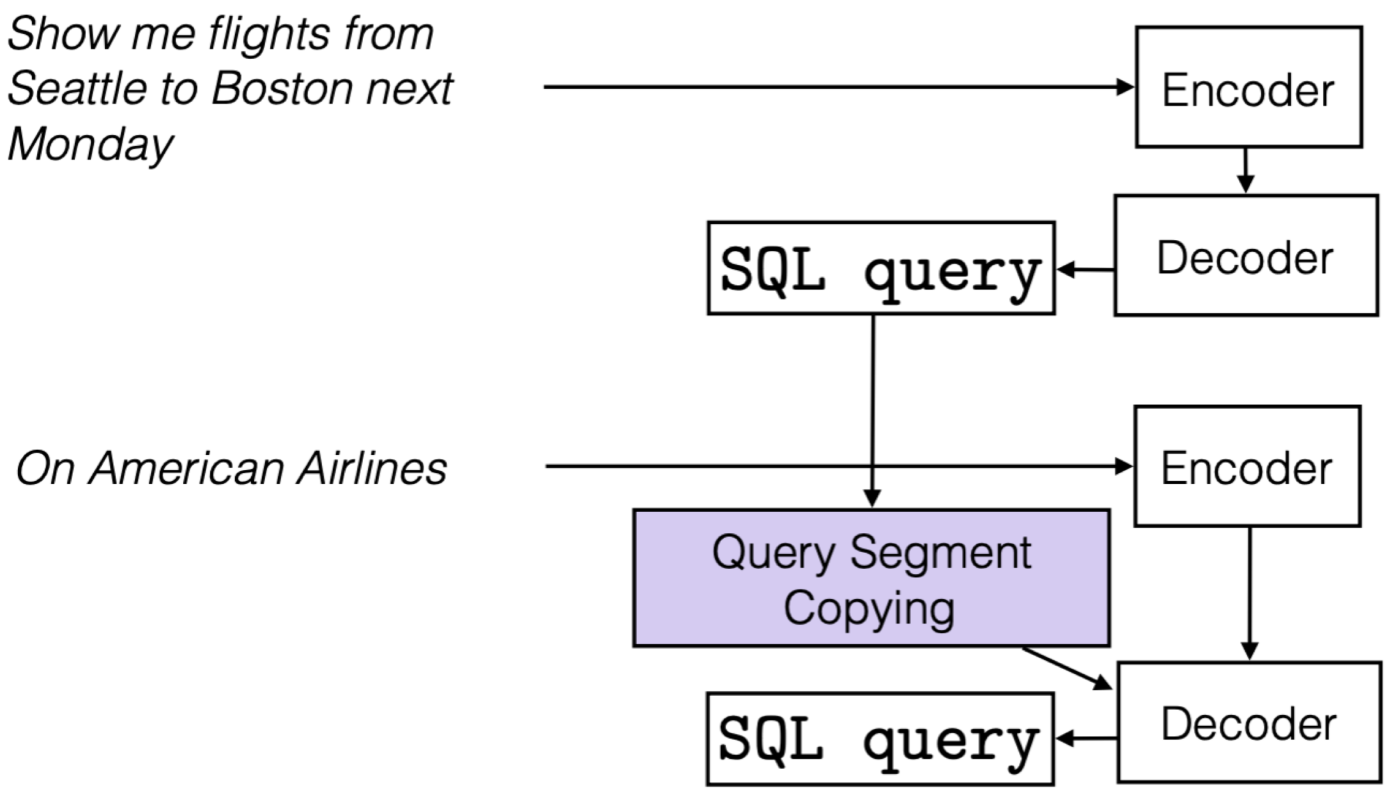
$$h^E_{i,j} = LSTM^E\left(\left[\phi\left(x_{i,j}\right); h^I_{i-1}\right]; h_{i,j-1}\right)$$

- x_i is the current utterance
- ϕ is the embedding
- h^I_{i-1} is the discourse state following utterance x_{i-1}
- $h^I_i = LSTM^I\left(h^E_{i\ |x_i|}; h^I_{i-1}\right)$

Mechanism #2:

Incorporating Previous Query

Incorporating Previous **Query**



Mechanism 2 **Previous Queries: Query Segment Copying**

Incorporating Previous **Query**

<u>Previous Query:</u>

```
(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 =
                    ⋮
```

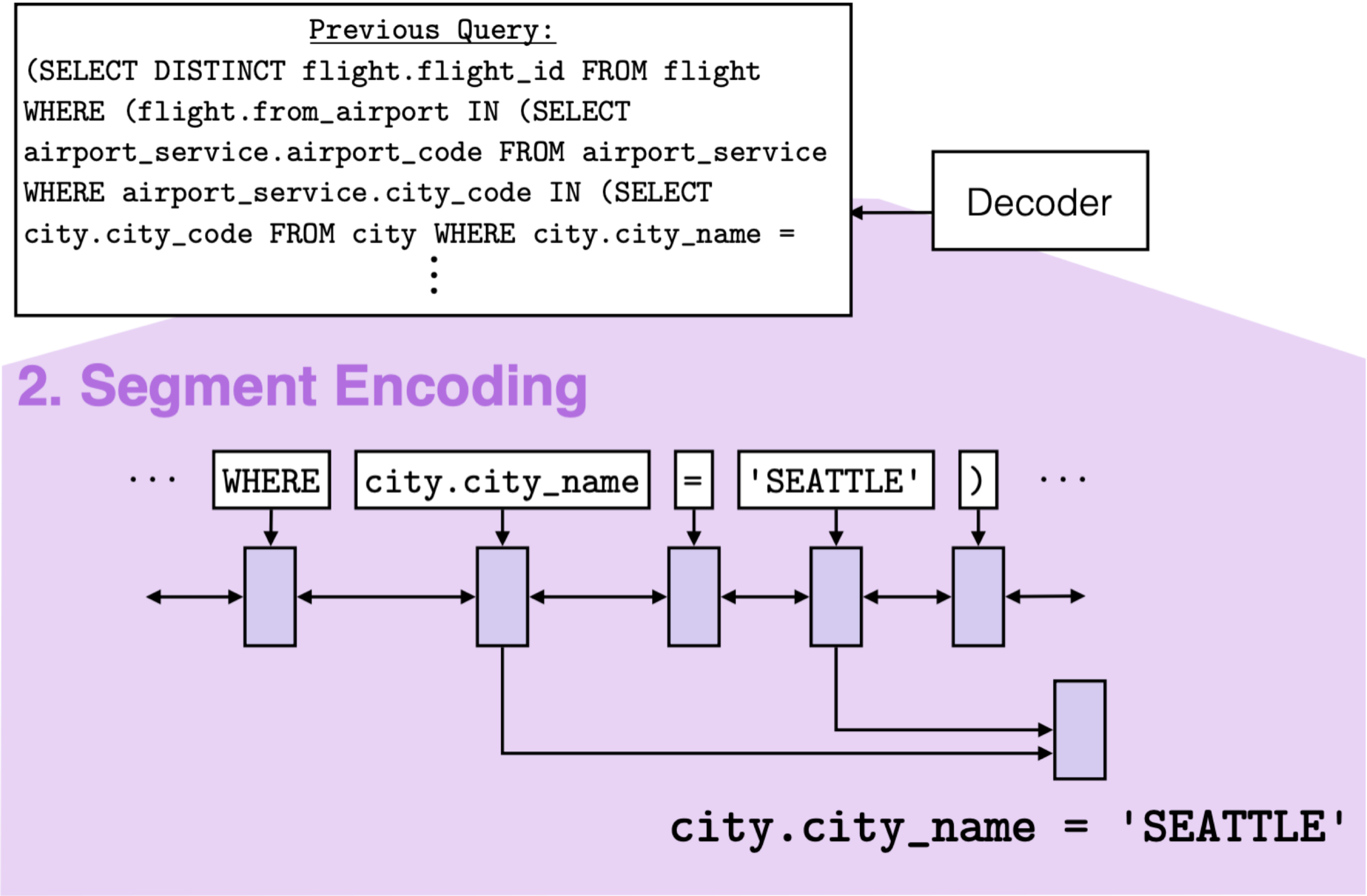
Decoder

1. Segment Extraction

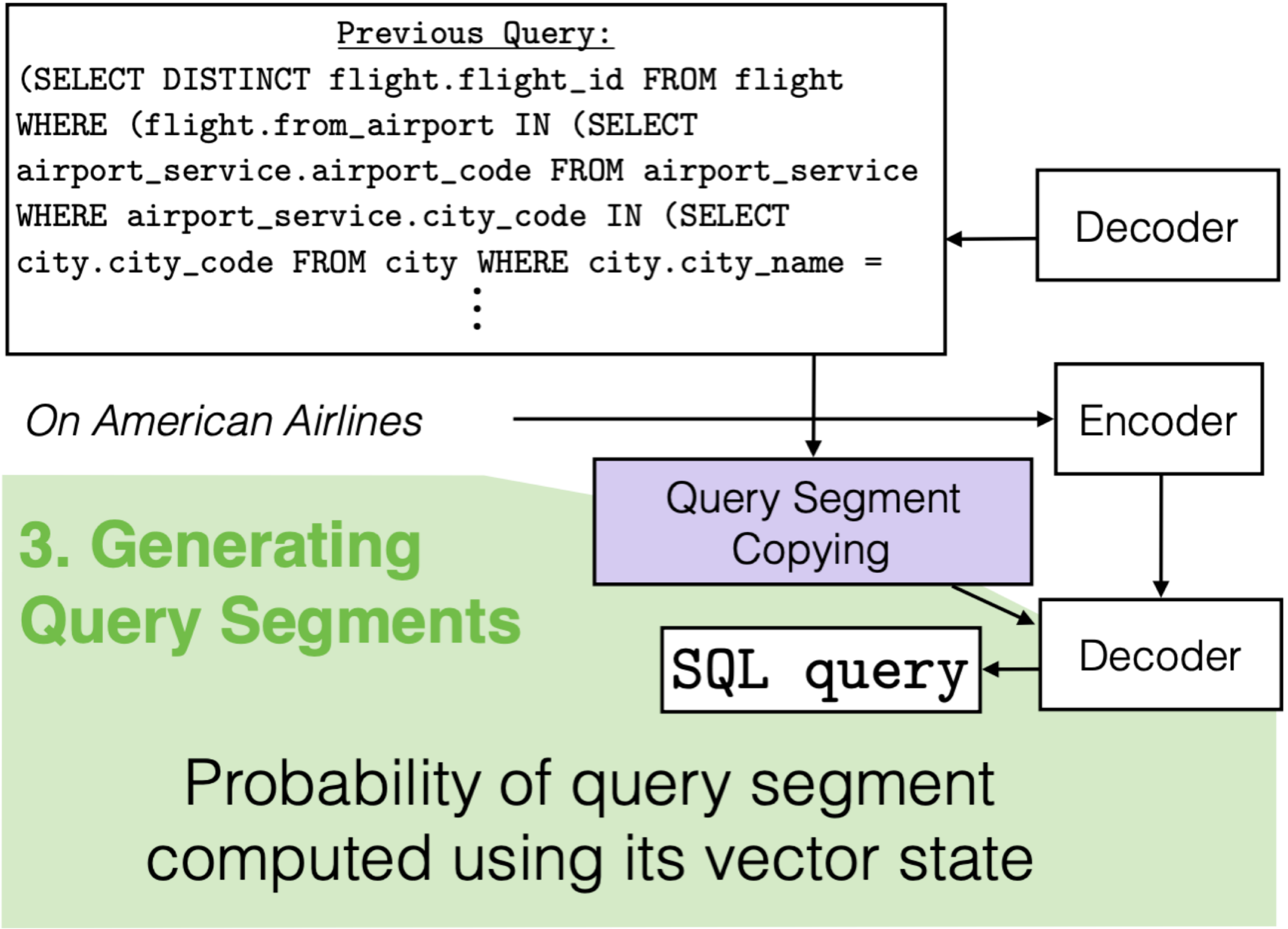
```
city.city_name = 'SEATTLE'
city.city_name = 'BOSTON'
date_day.year = 1993
date_day.month_number = 2
date_day.day_number = 8
          ⋮
```

Deterministic, operates on the SQL tree

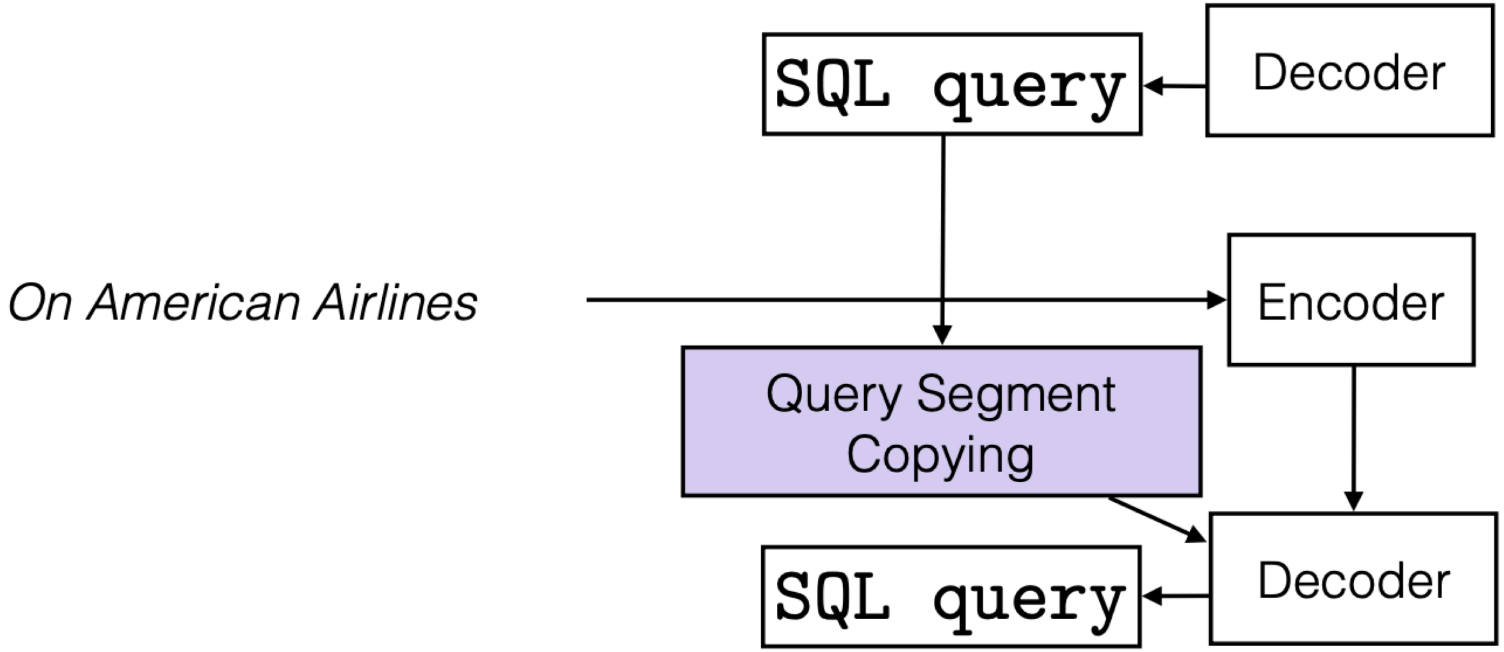
Incorporating Previous **Query**



Incorporating Previous **Query**



Incorporating Previous **Query**

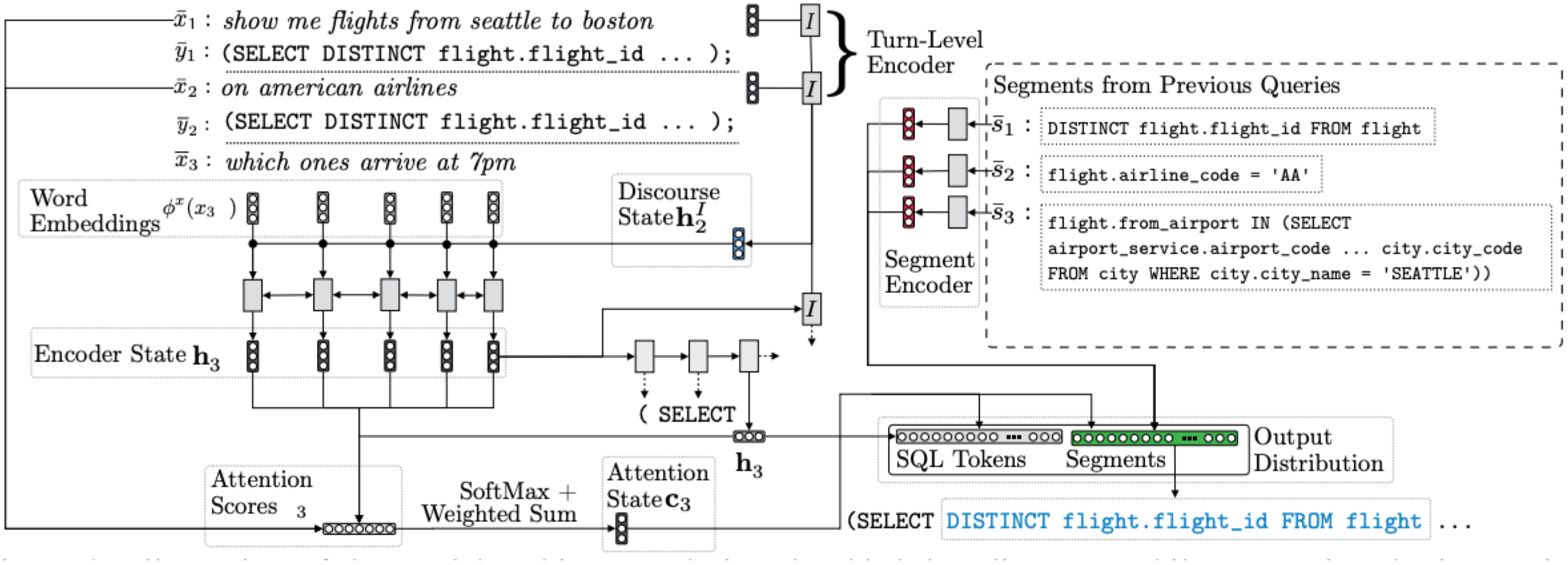


- Explicit mechanism for copying previous constraints
- Encoding and generating segments learned with the rest of the model

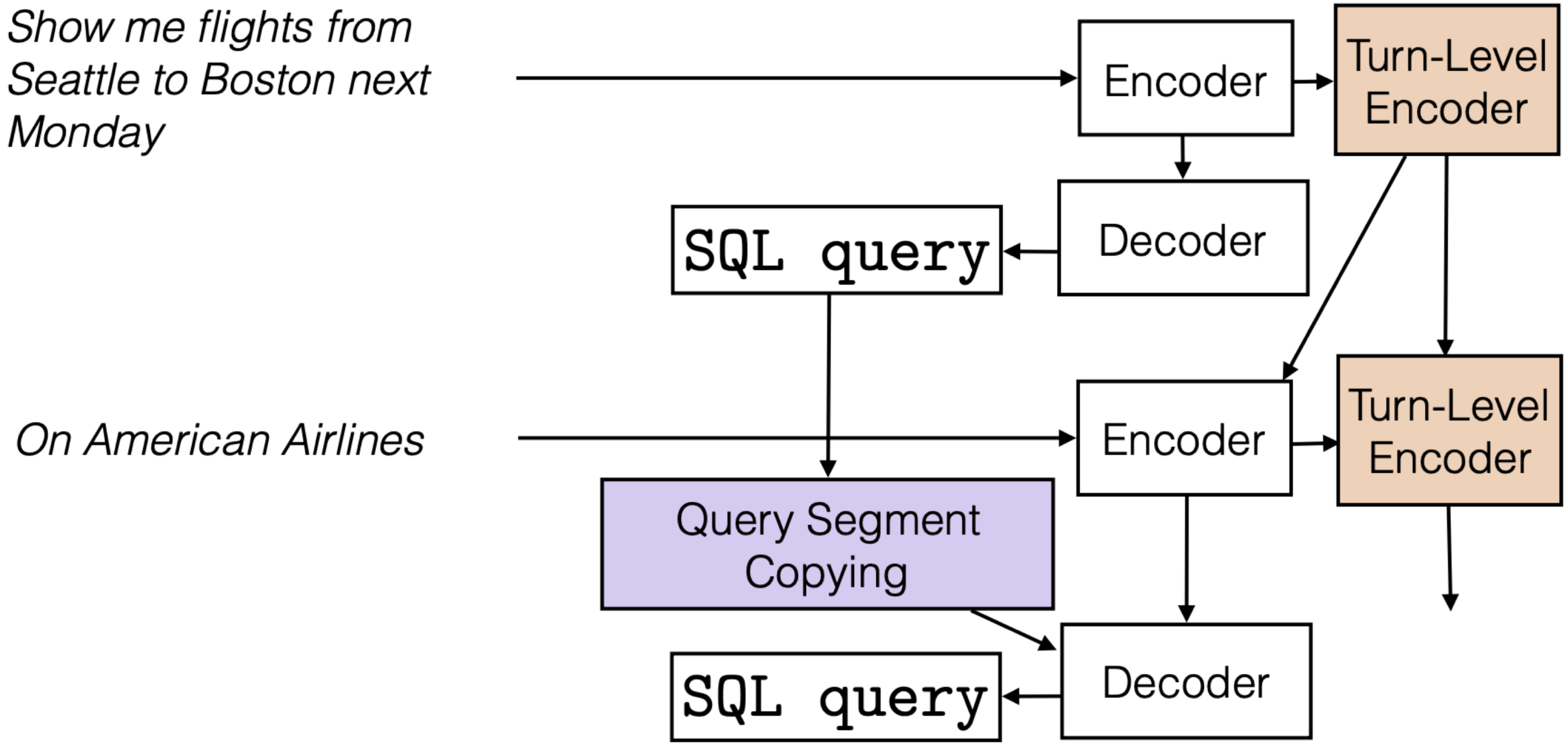
Incorporating Previous Requests

- $h^S = \left[h_l^Q; h_r^Q; \phi^g(y_b)\right]$ represents the hidden state of segment encoding
- ϕ^g is the embedding of the SQL
- $< h_1^Q, h_2^Q \ \dots, h_n^Q >$ are the query level hidden state
- y_b is the generated SQL

System Diagram



Idea



Mechanism 1 **Previous Requests: Turn-level Encoder**

Mechanism 2 **Previous Queries: Query Segment Copying**

Training

End to end training …

1) Training data: interactions with <SQL, request > pairs

2) Loss function: minimize token-level cross-entropy loss (against with gold query)

Evaluation

ATIS Dataset (Hemphill et al; Dahl et al 1994)

1) Flight information, 27 tables, 162K entries

2) Small corpus: <2000 interactions

3) Long interactions: average **7 turns;** maximum: 64 turns

4) Complex and long queries: average 102.9 tokens each;

Models to Evaluate

- **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

- Measures the effect of error propagation:
  - Full model with access to gold previous query

Evaluation Metric: Denotation accuracy

  - Comparing against with retrieved tables executed by generated-query vs gold-query

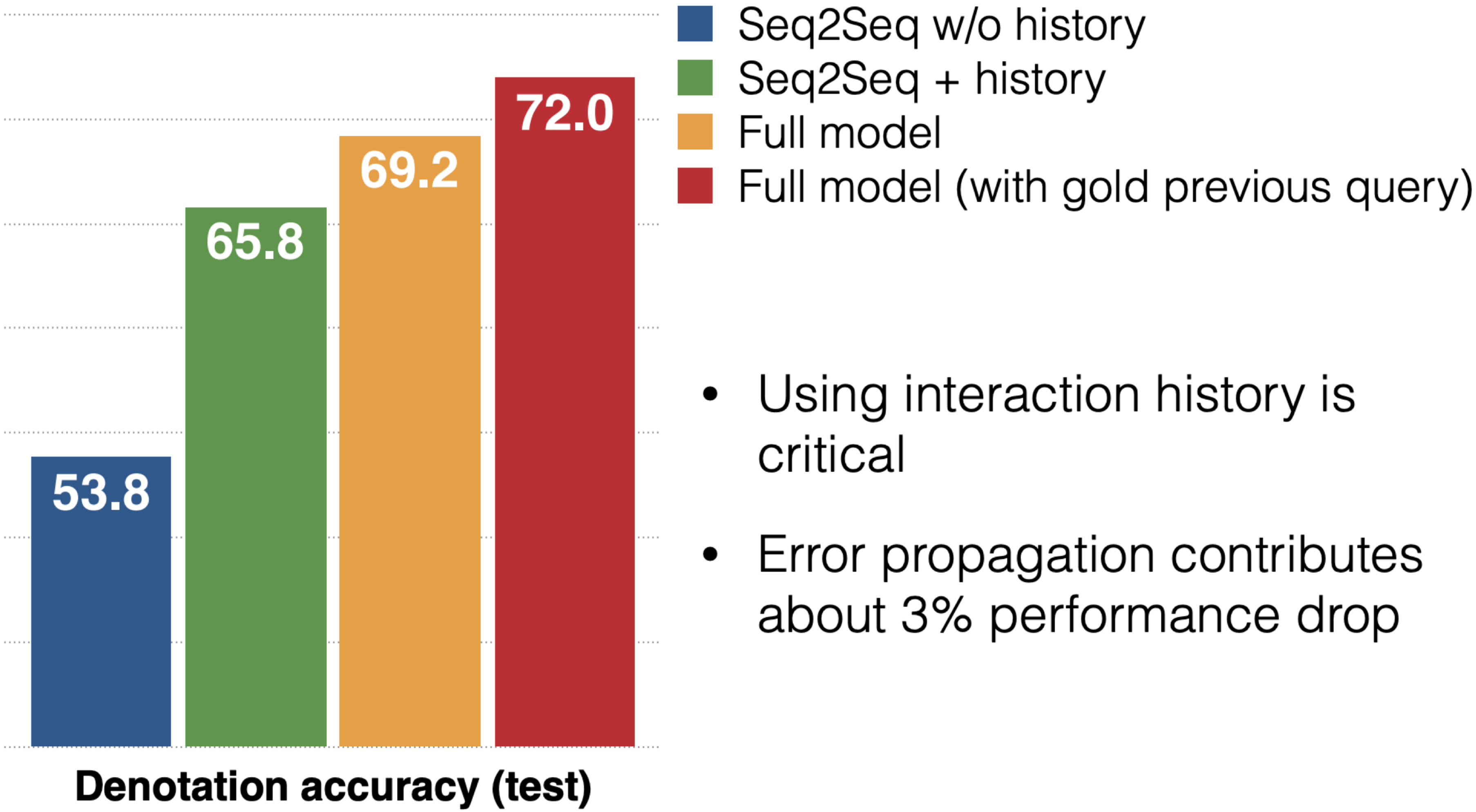
Other Evaluation measures..

- Query accuracy
  - % predicted **query** match with gold query

- Strict denotation accuracy
  - % **table** executed by query match with that of gold

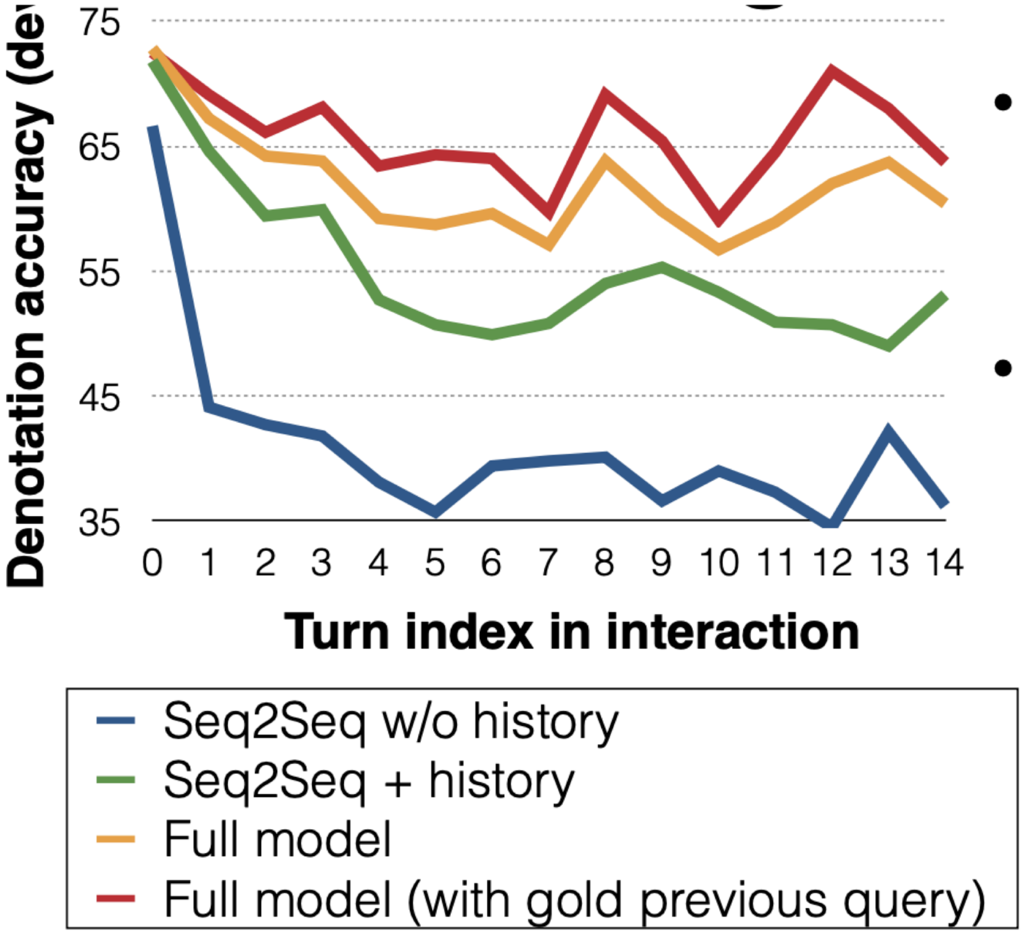
- Relaxed denotation accuracy
  - Give credit to b) if the gold query produce empty table

Performance



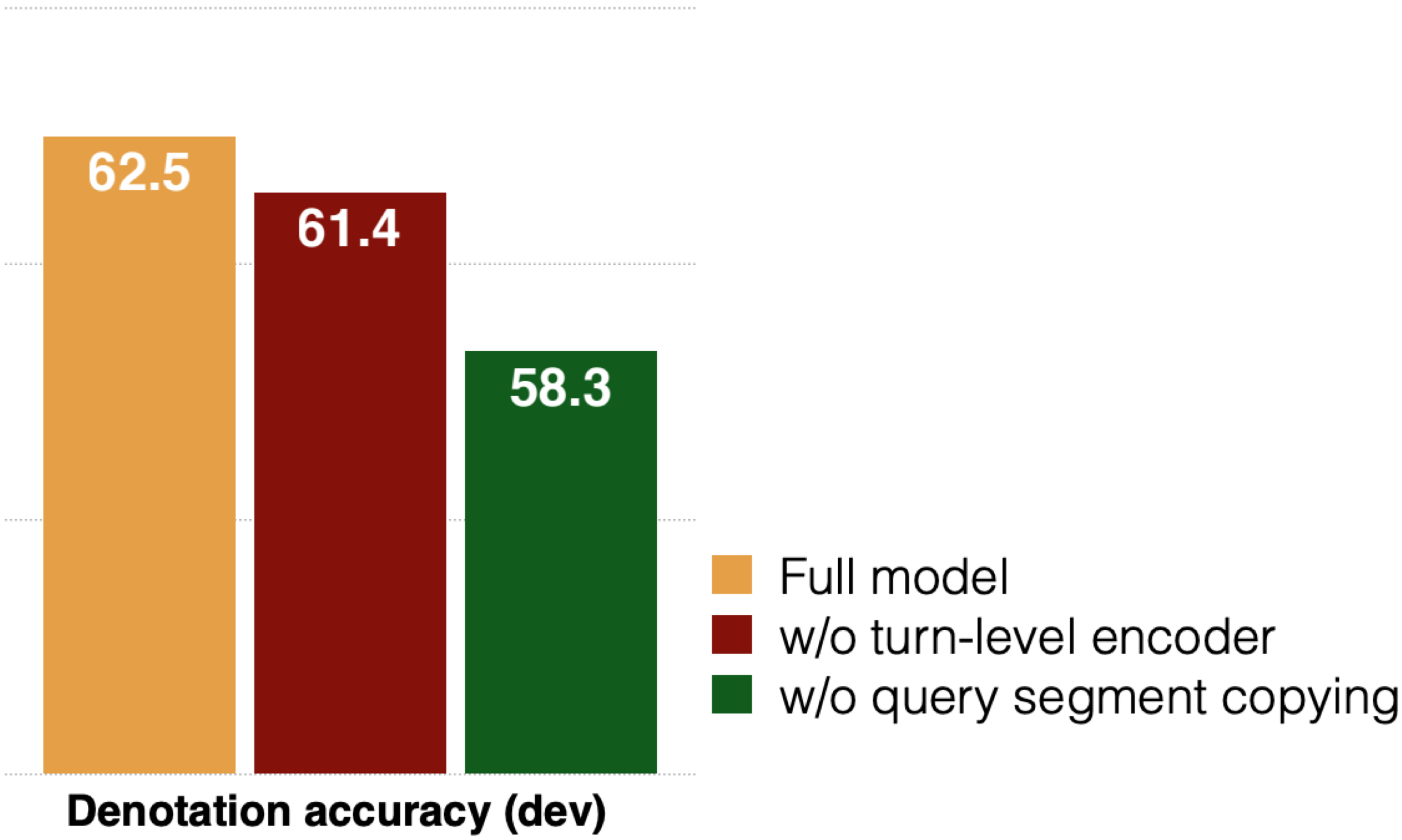
- Using interaction history is critical
- Error propagation contributes about 3% performance drop

Performance



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

Ablation Study



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 ) ) ) ) ) ;
```

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
```

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' ) )
```

⋮

Summary

- Language understanding in long and complex interactions
- Turn-level encoder: implicit mechanism for reasoning about previous requests
- Query segment copying: explicitly derive meaning of request (SQL query) from interaction history

Question:

The paperuses several different evaluation metrics. Describe their differences. Which one do you think is more reasonable (combining with the training objective)?

Question:

How to mitigate error propagation?