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

Example Task

Database Query

What states border Texas?

Oklahoma, New Mexico, Arkansas, Louisiana
Example Task

Instructing a Robot

at the chair,
turn right
Language to Meaning

Semantic Parsing

Recover complete meaning representation

More informative

Complete 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.

<table>
<thead>
<tr>
<th>Training Dataset</th>
<th>Test Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>600 queries</td>
<td>280 queries</td>
</tr>
</tbody>
</table>

**Question:**

*which state has the most rivers running through it?*

**Logical form:**

\[
\text{argmax } \theta_0 \\
\text{  (state: } t \theta_0) \\
\text{  (count } 1 \text{ (and)} \\
\text{  (river: } t \theta_1) \\
\text{  (loc: } t \theta_1 \theta_0)))
\]

**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.

<table>
<thead>
<tr>
<th>Training Dataset</th>
<th>Test Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>500 queries</td>
<td>140 queries</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Training Dataset</th>
<th>Development Dataset</th>
<th>Test Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>4480 instances</td>
<td>480 instances</td>
<td>450 instances</td>
</tr>
</tbody>
</table>

**Request:**

*Show me flights from Pittsburgh to Seattle*

**Logical form:**

\[
\text{lambda } \$0 \ e \ (\text{and} \ (\text{flight } \$0) \\
(\text{from } \$0 \ \text{pittsburgh:ci}) \\
(\text{to } \$0 \ \text{seattle:ci}))
\]

**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 website¹.

---

1. https://ifttt.com

[Quirk et al., 2015]
Training Data
(sentence, implementation) pairs

What is the largest city in Hawaii?  \rightarrow \text{answer}(A, \text{largest}(A, \text{city}(A), \text{loc}(A, B), \text{const}(B, \text{stateid}(hawaii))))

What is the capital of California?  \rightarrow \text{answer}(A, \text{capital}(A), \text{loc}(A, B), \text{const}(B, \text{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

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

\[ h_t = f_{LSTM}(h_t^{l-1}, h_t^{l-1}) \]
\[ p_t = f \odot p_t^{l-1} + i \odot g \]
\[ h_t = o \odot \tanh(p_t) \]

\[ g_t^0 = W_q e(q_t) \]
\[ h_t^0 = W_a e(a_{t-1}) \]
\[ W_q \in \mathbb{R}^{n \times |V_q|} \quad W_a \in \mathbb{R}^{n \times |V_a|} \]
\[ p(a_t | a_{<t}, q) = \text{softmax}_{a_t}(W_o h_t) \]
\[ W_o \in \mathbb{R}^{|V_a| \times n} \]
Attention Mechanism for Neural Semantic Parsing

\[ c_t = \sum_{k=1}^{\lvert q \rvert} r_{t,k} h_k^L \]

\[ r_{t,k} \propto \exp\{h_t^L, h_k^L\} \]

\[ \sum_{j=1}^{\lvert q \rvert} r_{i,j} = 1 \]

\( h_1^L, \ldots, h_{\lvert q \rvert}^L \) are the top layer hidden vectors of the encoder

\[ h_t^{att} = \tanh(W_1 h_t^L + W_2 c_t) \]

\[ p(a_t|a_{<t}, q) = \text{softmax}_{a_t}(W_o h_t^{att}) \]
Drawbacks of Seq2Seq Model

Ignore the hierarchical structure of logical forms

Use `(` `)` to linearize logical form

$$\lambda x \ v \ <x> \ \land \ <x> \ <x> \ > \ <x> \ 1600:ti \ \land \ \text{from} \ <x> \ \text{dallas:ci}$$

$$\lambda x \ v \ (\land \ (> \ (\text{departure\_time} \ <x>) \ 1600:ti) \ (\text{from} \ <x> \ \text{dallas:ci}))$$
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

---

**Task-Specific Meaning Representations**

*Show me flights from Pittsburgh to Seattle*

\[
\text{lambda } \emptyset \text{ e (and (flight } \emptyset ) \\
\text{ (from } \emptyset \text{ san_Francisco:ci) \\
\text{ (to } \emptyset \text{ seattle:ci))}
\]

**Tree-structured Representation**
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

Show me flight from Dallas departing after 16:00
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

(l 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: \( A \ B \ (C) \)

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

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

- **Goal:**
  Maximize the likelihood of the generated logical forms given natural language utterances as input

- **Objective function:**
  
  \[
  \text{minimize} \sum_{(q,a) \in \mathcal{D}} \log p(a | q)
  \]
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 num_0”
     and its logical form: “job(ANS), salary greater than(ANS, 40000, year)”
     → “job(ANS), salary greater than(ANS, 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 \( \text{type}_j \) 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

Accuracy

<table>
<thead>
<tr>
<th>Jobs</th>
<th>ATIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZC07</td>
<td>84.6</td>
</tr>
<tr>
<td>UBL</td>
<td>71.4</td>
</tr>
<tr>
<td>FUBL</td>
<td>82.8</td>
</tr>
<tr>
<td>GUSP-FULL</td>
<td>74.8</td>
</tr>
<tr>
<td>GUSP++</td>
<td>83.5</td>
</tr>
<tr>
<td>TISP</td>
<td>84.2</td>
</tr>
<tr>
<td>Seq2SeQ</td>
<td>84.2</td>
</tr>
<tr>
<td>Seq2Tree</td>
<td>84.6</td>
</tr>
</tbody>
</table>

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
Results

Geo

Accuracy

Zettlemoyer and Collins,... 79.3
Kwiatkowski et al., 2010 86.1
Kwiatkowski et al., 2011 87.9
Liang et al., 2013 88.6
Zhao and Huang, 2015 89
vanilla Seq2Seq 87.9
w/argument 88.9
w/attention 84.6
Seq2Tree 87.1

IFTTT

F1

retrieval 56.2
phrase 45.5
sync 42.8
classifier 65
posclass 66.5
vanilla Seq2Seq 70.8
w/argument 72.9
w/attention 73.7
Seq2Tree 74.2

(Quirk et al., 2015): IFTTT baselines
Ablation Study

**JOBS**

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq2Seq w/o attention</td>
<td>87.1</td>
</tr>
<tr>
<td>Seq2Seq w/o argument</td>
<td>77.9</td>
</tr>
<tr>
<td>Seq2Tree w/o attention</td>
<td>90</td>
</tr>
<tr>
<td>Seq2Tree w/o attention</td>
<td>83.6</td>
</tr>
</tbody>
</table>

**GEO**

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq2Seq w/o attention</td>
<td>84.6</td>
</tr>
<tr>
<td>Seq2Seq w/o argument</td>
<td>72.9</td>
</tr>
<tr>
<td>Seq2Tree w/o attention</td>
<td>68.6</td>
</tr>
<tr>
<td>Seq2Tree w/o attention</td>
<td>87.1</td>
</tr>
<tr>
<td>Seq2Tree w/o attention</td>
<td>76.8</td>
</tr>
</tbody>
</table>

**ATIS**

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq2Seq w/o attention</td>
<td>84.2</td>
</tr>
<tr>
<td>Seq2Seq w/o argument</td>
<td>75.7</td>
</tr>
<tr>
<td>Seq2Tree w/o attention</td>
<td>72.3</td>
</tr>
<tr>
<td>Seq2Tree w/o attention</td>
<td>77.5</td>
</tr>
</tbody>
</table>

**IFTTT (≥ 3 turkers agree with gold)**

<table>
<thead>
<tr>
<th>Model Type</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq2Seq</td>
<td>73.7</td>
</tr>
<tr>
<td>Seq2Seq w/o attention</td>
<td>72.9</td>
</tr>
<tr>
<td>Seq2Tree w/o argument</td>
<td>70.8</td>
</tr>
<tr>
<td>Seq2Tree</td>
<td>74.2</td>
</tr>
<tr>
<td>Seq2Tree w/o attention</td>
<td>73.5</td>
</tr>
</tbody>
</table>
Attention helps!

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

<table>
<thead>
<tr>
<th>User</th>
<th>Show me flights from Seattle to Boston next Monday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>Found 31 Flights: ![Flights Image]</td>
</tr>
</tbody>
</table>
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
Show me flights from Seattle to Boston next Monday
Idea

Show me flights from Seattle to Boston next Monday

On American Airlines
Idea

Show me flights from Seattle to Boston next Monday

On American Airlines

Mechanism 1

Previous Requests: Turn-level Encoder
Idea

Show me flights from Seattle to Boston next Monday

On American Airlines

Previous Requests: Turn-level Encoder
Previous Queries: Query Segment Copying
Mechanism #1:
Incorporating Previous Request
Incorporating Previous Requests

Show me flights from Seattle to Boston next Monday

On American Airlines

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

Show me flights from Seattle to Boston next Monday

1. State Update

Encoded request

RNN Update

Discourse-level vector state

New discourse-level vector state

Encoder

Turn-Level Encoder
Incorporating Previous Requests

Show me flights from Seattle to Boston next Monday

On American Airlines

2. Using State
   On
   American Airlines
Incorporating Previous Requests

Show me flights from Seattle to Boston next Monday

1. State Update

Encoded request

RNN Update

Discourse-level vector state

New discourse-level vector state
Incorporating Previous Requests

- Show me flights from Seattle to Boston next Monday
- On American Airlines
  - Persistent vector state, updated throughout interaction
  - Encode information from beginning to end of interaction
  - Completely learned
Incorporating Previous Requests

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

- \( x_i \) is the current utterance
- \( \phi \) is the embedding
- \( h_{i-1}^l \) is the discourse state following utterance \( x_{i-1} \)
- \( h_i^l = LSTM^I \left( h_i^E |_{x_i}; h_{i-1}^l \right) \)
Mechanism #2: Incorporating Previous Query
Incorporating Previous Query

Show me flights from Seattle to Boston next Monday

On American Airlines

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

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 =
:))

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

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

... WHERE city.city_name = 'SEATTLE' ) ...

... city.city_name = 'SEATTLE'
Incorporating Previous Query

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 =
:)))

3. Generating Query Segments

On American Airlines

Query Segment Copying

SQL query

Probability of query segment computed using its vector state
Incorporating Previous **Query**

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

- \( h^S = [h^Q_i; h^Q_r; \phi^g(y_b)] \) represents the hidden state of segment encoding
- \( \phi^g \) is the embedding of the SQL
- \( < h^Q_1, h^Q_2, \ldots, h^Q_n > \) are the query level hidden state
- \( y_b \) is the generated SQL
System Diagram

\[ x_1: \text{show me flights from seattle to boston} \]
\[ y_1: \text{(SELECT DISTINCT flight.flight_id ...)} \]
\[ x_2: \text{on american airlines} \]
\[ y_2: \text{(SELECT DISTINCT flight.flight_id ...)} \]
\[ x_3: \text{which ones arrive at 7pm} \]

\[ \phi^x(x_3) \]

Encoder State \( h_2 \)

Turn-Level Encoder

Segments from Previous Queries

\[ s_1: \text{DISTINCT flight.flight_id FROM flight} \]
\[ s_2: \text{flight.airline_code = 'AA'} \]
\[ s_3: \text{flight.from_airport IN (SELECT airport_service.airport_code ... city.city_code FROM city WHERE city.city_name = 'SEATTLE')} \]

Segment Encoder

Attention Scores

SoftMax + Weighted Sum

Output Distribution

\[ \text{(SELECT DISTINCT flight.flight_id FROM flight ...)} \]
Idea

Show me flights from Seattle to Boston next Monday

On American Airlines

Previous Requests: Turn-level Encoder
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

Denotation accuracy (dev)

- Full model: 62.5
- w/o turn-level encoder: 61.4
- w/o query segment copying: 58.3
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?

Error: looking for flights around 5pm

```
flight.arrival_time >= 1630 AND flight.arrival_time <= 1730
```
<table>
<thead>
<tr>
<th>User</th>
<th>Which kind of airplane is that?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Query</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( 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 &gt;= 1630 AND flight.arrival_time &lt;= 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' ) ) ) )</td>
</tr>
<tr>
<td></td>
<td>)</td>
</tr>
</tbody>
</table>
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
The paper uses 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?