Lecture 5

Introduction to Virtual Assistants with Mixed Initiatives

By Monica Lam
Task-Oriented Conversational Assistants

• Commercial state of the art
  • Commercial conversational chatbots: built with dialogue trees
  • Virtual assistants are not conversational

• Research state of the art
  • Dialogue representation:
    • Slot-value pairs, except for SMCalFlow* & Genie (ThingTalk)
  • Full training data:
    • Wizard-of-Oz conversations: crowdsource workers play user & agent
    • Hand annotation of full training data
    • MultiWOZ: Multi-domain training data have many errors; poor results
    • RiSAWOZ: More accurate annotations; 66% joint accuracy

• Few-shot + synthesis (Genie with ThingTalk):
  • 79% joint accuracy on MultiWOZ

Goals of this Lecture

• **Introduction to task-oriented conversational assistants**
  • What do task-oriented conversations look like?
  • **Dialogue State:**
    What do the users mean? And how do we represent the meaning?
  • **Contextual semantic parsing:**
    What is the dialogue state tracking problem?

• **Introduction to GenieScript**
  • The only high-level dialogue scripting language to improve productivity (with automatic training data synthesis)
  • From a developer’s perspective: *What can GenieScript do for developers?*
  • Why is it in the intro?
    • Language design: A high-level abstraction that hides the gory details from developers
    • Some of you may want to use it in your project
Subsequent Lectures: How and Why?

Discuss different approaches on these topics
• How to represent conversations?
• How to acquire training data?
• How to train neural models?
• What are the experimental results?
Two Kinds of Task-Oriented Assistants

<table>
<thead>
<tr>
<th>Initiatives</th>
<th>Conversational Virtual Assistants</th>
<th>Conversational Chatbots</th>
</tr>
</thead>
<tbody>
<tr>
<td>User: Must understand the users’ sentences</td>
<td></td>
<td>Agent: control the conversation Typically asks users for values</td>
</tr>
<tr>
<td>Domain</td>
<td>Multi domains</td>
<td>Single domains</td>
</tr>
</tbody>
</table>

Overall goal: multi-domain, mixed initiatives
Outline

• Conversational virtual assistants: User initiatives
  • **What do task-oriented conversations look like?**
  • Definition: Dialogue State Tracking (DST) Problem
  • Optimization: Use formal state in DST
  • Optimization: Simplify annotations of conversations
  • Virtual assistants in GenieScript
• Adding mixed initiatives
Anatomy of an Agent

- User input
- Contextual Semantic Parser
- Context: Dialogue History
- Execution
- Dialogue State
- Result
- Agent Policy
- Agent Response
Complexity in Understanding User Initiatives

Co-references
A: "I recommend Terun"
U: “Please book it.”

Ellipses
A: “Would you like an Italian restaurant?”
U: “How about Chinese?”

Ambiguity
U: “I’d like a French restaurant in Menlo Park”
A: “How about Left Bank? It is moderately priced”
U: “I want a cheap restaurant in Palo Alto.”
Example: Restaurant Reservation in \texttt{GenieScript}

Com.yelp Restaurant Skills

Knowledge Base (KB)

<table>
<thead>
<tr>
<th>Restaurant</th>
<th>“restaurant”, “food place”</th>
</tr>
</thead>
<tbody>
<tr>
<td>id:</td>
<td>Entity(Restaurant)</td>
</tr>
<tr>
<td>geo:</td>
<td>Location</td>
</tr>
<tr>
<td></td>
<td>[ “address”, “in #”, “near #”, “around #” ]</td>
</tr>
<tr>
<td>price:</td>
<td>Enum(cheap, moderate, expensive)</td>
</tr>
<tr>
<td></td>
<td>[ “#-ly priced”, “#” ]</td>
</tr>
<tr>
<td>cuisines:</td>
<td>Array(Entity(Cuisine))</td>
</tr>
<tr>
<td></td>
<td>[ “# food”, “serves # food” ]</td>
</tr>
</tbody>
</table>

Action

<table>
<thead>
<tr>
<th>MakeReservation</th>
<th>“reserve #”, “book #”</th>
</tr>
</thead>
<tbody>
<tr>
<td>restaurant:</td>
<td>Entity(Restaurant)</td>
</tr>
<tr>
<td>book_people:</td>
<td>Number [min=1]</td>
</tr>
<tr>
<td>book_day:</td>
<td>Date [ “for #” ]</td>
</tr>
<tr>
<td>book_time:</td>
<td>Time [ “at #”, “for #” ]</td>
</tr>
</tbody>
</table>

main () {
    import com.yelp;
    dlg.say
        (“Hello, I am Genie, your restaurant assistant.”);
}

GenieScript: JavaScript + dlg package

GenieScript automatically trains contextual semantic parser on possible queries and API invocations
Hello, I am Genie, your restaurant assistant.
I’d recommend Chez TJ in Mountain View. It has one Michelin star!

The distance to Chez TJ is 7.5 miles.

The distance to Left Bank is 2.8 miles.

The price is moderate.

I booked Left Bank for 2 on November 2 at 7pm.

10382345.

Imagine writing this in a dialogue tree for each domain!
Hello, I am Genie, your restaurant assistant.

I’d recommend Chez TJ in Mountain View. It has one Michelin star!

The distance to Chez TJ is 7.5 miles.

The distance to Left Bank is 2.8 miles.

Perfect. Please book the restaurant

The price is moderate.

OK

Which day would you like to book Left Bank?

November 2, 7pm, for 2 people.

Could you confirm that you want a table for 2 people at Left Bank on November 2 at 7pm?

Yes.

I booked Left Bank for 2 on November 2 at 7pm.

What is the reservation number?

10382345.

Quiz: Why do we need to confirm?
What Is the User Saying? (In ThingTalk)

@com.yelp.restaurant() filter cuisine == ‘French’
[distance(geo, $location.current_location) of
@com.yelp.restaurant() filter cuisine == ‘French’
[distance(geo, $location.current_location) of
@com.yelp.restaurant() filter cuisine == ‘French’
&& geo == $location.current_location
@com.yelp.book_restaurant(id = ‘Left Bank’)

[price] of @com.yelp.restaurant() filter id == ‘Left Bank’

$yes
@com.yelp.book_restaurant(id = ‘Left Bank’, date = Date(11,2), time = 19:00, people = 2 )

@com.yelp.book_restaurant(id = ‘Left Bank’, date = Date(11,2), time = 19:00, people = 2 )
#[confirm = confirmed]

[reservation_number] of @com.yelp.book_restaurant(id = ‘Left Bank’, date = Date(11,2), time = 19:00, people = 2 )

Stanford University

**Quiz:** Why do we need to confirm?
User Dialogue State

- User dialogue state
  - Mostly commands!
    Execution: Query, ActionQuery, or InvokeAction (Unconfirmed/Confirmed)
  - Issue 2 meta statements
    - **Done:** “ok”, “yes”, “sounds good”: positive response
    - **Cancel:** “I don’t like that”, “Never mind”: negative response
      -- can be issued any time
What is the Agent saying?

<table>
<thead>
<tr>
<th>USER</th>
<th>GENIE</th>
<th>ActionQueryResponse</th>
</tr>
</thead>
<tbody>
<tr>
<td>I’d like a French Restaurant.</td>
<td>Hello, I am Genie, your restaurant assistant.</td>
<td>Greeting in GenieScript</td>
</tr>
<tr>
<td>I’d recommend Chez TJ in Mountain View. It has one Michelin star!</td>
<td>The distance to Chez TJ is 7.5 miles.</td>
<td>QueryResponse</td>
</tr>
<tr>
<td>How far is it from here?</td>
<td>The distance to Left Bank is 2.8 miles.</td>
<td>QueryResponse</td>
</tr>
<tr>
<td>Can you recommend something nearby?</td>
<td>Perfect. Please book the restaurant</td>
<td>QueryResponse</td>
</tr>
<tr>
<td>Which day would you like to book Left Bank?</td>
<td>The price is moderate.</td>
<td>Slot filling</td>
</tr>
<tr>
<td>Is it expensive?</td>
<td>Book Left Bank</td>
<td>Book Left Bank</td>
</tr>
<tr>
<td>OK</td>
<td>Which day would you like to book Left Bank?</td>
<td>QueryResponse</td>
</tr>
<tr>
<td>November 2, 7pm, for 2 people.</td>
<td>Could you confirm that you want a table</td>
<td>Slot filling</td>
</tr>
<tr>
<td></td>
<td>for 2 people at Left Bank on November 2 at 7pm?</td>
<td>Book Left Bank</td>
</tr>
<tr>
<td>Yes.</td>
<td>I booked Left Bank for 2 on November 2 at 7pm.</td>
<td>ActionResponse</td>
</tr>
<tr>
<td>What is the reservation number?</td>
<td>10382345.</td>
<td>ActionQueryResponse</td>
</tr>
</tbody>
</table>
User & Agent Dialogue State

• User dialogue state
  • **Execution**: Query, ActionQuery, or InvokeAction (Unconfirmed/Confirmed)
  • Issue 2 meta statements
    • **Done**: “ok”, “yes”, “sounds good”: positive response
    • **Cancel**: “I don’t like that”, “Never mind”: negative response
      -- can be issued any time

• Agent dialogue state
  • **Response**: to Queries, ActionQueries, InvokeActions (confirmed)
  • Ask 2 kinds of questions
    • **Slot filling**: missing required parameters
    • **AskToConfirm**: to ensure the parse is correct (actions have side effects)
Abstract Dialogue States (Virtual Assistants)

Any Time

KB Navigation

Query (TT)

SlotFillQuestion  QueryResponse  Done

Action

requesteds

InvokeAction (TT)

SlotFillQuestion

AsktoConfirm

confirmed

InvokeAction (TT)

ActionQuery (TT)

ActionResponse

ActionQueryResponse

Done

Agent implementation:
executes these domain-independent state machines
Hello, I am Genie, your restaurant assistant.
I’d recommend Chez TJ in Mountain View. It has one Michelin star!
The distance to Chez TJ is 7.5 miles.
The distance to Left Bank is 2.8 miles.
Can you recommend something nearby?
The price is moderate.
Perfect. Please book the restaurant
Which day would you like to book Left Bank?
The price is moderate.
November 2, 7pm, for 2 people.
Could you confirm that you want a table
for 2 people at Left Bank on November 2 at 7pm?
Yes.
I booked Left Bank for 2 on November 2 at 7pm.
What is the reservation number?
10382345.
How Should We Handle Interrupts?

• Save outstanding queries/actions on a stack
  • Process latest queries first
  • When the user says “done” or if the user pauses
    • Repeat the agent’s question (automatically)
Outline

• Conversational virtual assistants: User initiatives
  • What do task-oriented conversations look like?
  • **Definition: Dialogue State Tracking (DST) Problem**
  • Optimization: Use formal state in DST
  • Optimization: Simplify annotations of conversations
  • Synthesis overview
• Adding mixed initiatives
<table>
<thead>
<tr>
<th>USER</th>
<th>GENIE</th>
</tr>
</thead>
</table>
| I’d like a French Restaurant.  
I’d recommend Chez TJ in Mountain View. It has one Michelin star!  
How far is it from here? | Hello, I am Genie.  
The distance to Chez TJ is 7.5 miles.  
The distance to Left Bank is 2.8 miles.  
Perfect. Please book the restaurant  
Which day would you like to book Left Bank?  
Is it expensive? | The price is moderate.  
OK  
Which day would you like to book Left Bank?  
November 2, 7pm, for 2 people.  
Could you confirm that you want a table  
for 2 people at Left Bank on November 2 at 7pm? | Yes.  
I booked Left Bank for 2 on November 2 at 7pm.  
What is the reservation number? | 10382345. |
I’d like a French Restaurant.
I’d recommend Chez TJ in Mountain View. It has one Michelin star!
How far is it from here?
The distance to Chez TJ is 7.5 miles.
Can you recommend something near by?
The distance to Left Bank is 2.8 miles.
Perfect. Please book the restaurant
Which day would you like to book Left Bank?
Is it expensive?
The price is moderate.
OK
Which day would you like to book Left Bank?
November 2, 7pm, for 2 people.
Could you confirm that you want a table
for 2 people at Left Bank on November 2 at 7pm?
Yes.
I booked Left Bank for 2 on November 2 at 7pm.
What is the reservation number?
10382345.
Dialogue State Tracking (DST) Problem Definition

- **Dialogue State (DS):** Meaning of the user utterance
  - ThingTalk, Slot-value pairs (traditional)

- Problem: Given a dialogue with alternating user-agent turns in NL, \( u_1, a_1, u_2, a_2, \ldots \)

  For each turn \( i \)
  - Predict the dialogue state: \( DS \ (u_1, a_1, \ldots, u_i) \)
  - Training data:
    \[ ((u_1, a_1, \ldots, a_{i-1}, u_i), DS \ (u_1, a_1, \ldots, u_i)) \]

  A dialogue with \( n \) turns \( \rightarrow \) \( n \) training samples
**Quiz:**

How hard is this problem?
- 1st sentence
- last sentence

Can we make parsing conversation easier?

<table>
<thead>
<tr>
<th>USER</th>
<th>GENIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I’d like a French Restaurant.</td>
<td>Hello, I am Genie.</td>
</tr>
<tr>
<td>I’d recommend Chez TJ in Mountain View. It has one Michelin star!</td>
<td></td>
</tr>
<tr>
<td>How far is it from here?</td>
<td>The distance to Chez TJ is 7.5 miles.</td>
</tr>
<tr>
<td>Can you recommend something near by?</td>
<td>The distance to Left Bank is 2.8 miles.</td>
</tr>
<tr>
<td>Perfect. Please book the restaurant</td>
<td>Which day would you like to book Left Bank?</td>
</tr>
<tr>
<td>Is it expensive?</td>
<td>The price is moderate.</td>
</tr>
<tr>
<td>OK</td>
<td>Which day would you like to book Left Bank?</td>
</tr>
<tr>
<td>November 2, 7pm, for 2 people.</td>
<td>Could you confirm that you want a table for 2 people at Left Bank on November 2 at 7pm?</td>
</tr>
<tr>
<td>Yes.</td>
<td>I booked Left Bank for 2 on November 2 at 7pm.</td>
</tr>
<tr>
<td>What is the reservation number?</td>
<td>10382345.</td>
</tr>
</tbody>
</table>
I’d like a French Restaurant.
I’d recommend Chez TJ in Mountain View. It has one Michelin star!

The distance to Chez TJ is 7.5 miles.

The distance to Left Bank is 2.8 miles.

Perfect. Please book the restaurant

Which day would you like to book Left Bank?

The price is moderate.

Which day would you like to book Left Bank?

November 2, 7pm, for 2 people.

Could you confirm that you want a table for 2 people at Left Bank on November 2 at 7pm?

Yes.

I booked Left Bank for 2 on November 2 at 7pm.

What is the reservation number?

10382345.
Outline

• Conversational virtual assistants: User initiatives
  • What do task-oriented conversations look like?
  • Definition: Dialogue State Tracking (DST) Problem
  • **Optimization: Use formal state in DST**
  • Optimization: Simplify annotations of conversations
  • Synthesis overview
• Adding mixed initiatives
Use Formal State in DST

- Do not parse agent statements
  - We generate them! Represent agent state formally
- Results
- Slot-fills
- Confirmations
- Do not reparse user statements
  - We already parsed it!
  - Replace conversation history with a formal representation
Agent Anatomy

1. **Contextual Semantic Parser:**
   (Dialogue State Tracking DST)
   User utterance + context $\rightarrow$ user state

2. **ThingTalk Execution:**
   ThingTalk $\rightarrow$ result state

3. **Agent policy:** code to decide the response
   User context + result $\rightarrow$ agent state

   **Natural Language Generator (NLG):**
   agent state $\rightarrow$ agent utterance

4. **Output Context:**
   Summarizes the dialogue to date
   Becomes input context for next turn
Dialogue Context for Multi-Domains

- **Full dialogue history**: For each KB and action in each domain
  - One outstanding request
  - History of completed requests (with answers)
  - Note: each request may represent a long discussion
- **Context used** for semantic parsing
  - Limit to the last 5 ongoing/completed statements

Quiz: An outstanding request is forgotten if not on the last 5. Is it OK?
Quiz

• Is it more accurate to use a formal state representation?

• Assumption: Formal state is better than NL history
## Dialogue State Tracking

<table>
<thead>
<tr>
<th>USER</th>
<th>GENIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I’d like a French Restaurant. I’d recommend Chez TJ in Mountain View. It has one Michelin star! How far is it from here?</td>
<td>Hello, I am Genie. The distance to Chez TJ is 7.5 miles.</td>
</tr>
<tr>
<td>Can you recommend something near by?</td>
<td>The distance to Left Bank is 2.8 miles.</td>
</tr>
<tr>
<td>Perfect. Please book the restaurant</td>
<td>Which day would you like to book Left Bank?</td>
</tr>
<tr>
<td>Is it expensive?</td>
<td>The price is moderate.</td>
</tr>
<tr>
<td>OK</td>
<td>Which day would you like to book Left Bank?</td>
</tr>
<tr>
<td>November 2, 7pm, for 2 people.</td>
<td>Could you confirm that you want a table for 2 people at Left Bank on November 2 at 7pm?</td>
</tr>
<tr>
<td>Yes.</td>
<td>I booked Left Bank for 2 on November 2 at 7pm.</td>
</tr>
<tr>
<td>What is the reservation number?</td>
<td>10382345.</td>
</tr>
</tbody>
</table>

### Quiz:

In dialogue state tracking, if turn $i-1$ is wrong, can turn $i$ be correct?

Does getting the turn $i$ right but not $(i-1)$ matter in real life?
Outline

• Conversational virtual assistants: User initiatives
  • What do task-oriented conversations look like?
  • Definition: Dialogue State Tracking (DST) Problem
  • Optimization: Use formal state in DST
  • Optimization: Simplify annotations of conversations
  • Synthesis overview
• Adding mixed initiatives
Annotating User Utterances

<table>
<thead>
<tr>
<th>Context (c)</th>
<th>English</th>
<th>ThingTalk</th>
</tr>
</thead>
<tbody>
<tr>
<td>How far is the best</td>
<td>[distance(geo, $location.current_location)]</td>
<td>(sort ratings desc of</td>
</tr>
<tr>
<td>Mexican restaurant?</td>
<td>of (sort ratings desc of</td>
<td></td>
</tr>
<tr>
<td></td>
<td>@com.yelp.restaurant filter cuisine == 'Mexican')[1]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Next Sentence</th>
<th>How about Italian?</th>
<th>(I want Italian restaurants)</th>
</tr>
</thead>
</table>

| Dialogue State      | How far is the best Italian restaurant?      | [distance(geo, $location.current_location)] of (sort ratings desc of       |
|---------------------|----------------------------------------------| (sort ratings desc of                                                       |
|                     |                                              | @com.yelp.restaurant filter cuisine == ‘Italian’)[1]|                                                                          |

- The dialogue state of a conversation can be large!
  - Highly dependent on the dialogue context
  - Hard to annotate, hard to predict, hard to synthesize (many possible contexts)
  - It can be ambiguous
  - Does not know which in the prediction is the “new info” – useful for agent responses

- Introduce the concept of “Core” sentence
  - A minimal, complete TT statement that covers and only covers what the user says, by referring to the context.
Concept of a “Core” Sentence

<table>
<thead>
<tr>
<th></th>
<th>English</th>
<th>ThingTalk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context (c)</td>
<td>How far is the best Mexican restaurant?</td>
<td>[distance(geo, $location.current_location)] of (sort ratings desc of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>@com.yelp.restaurant filter cuisine == 'Mexican')[1]</td>
</tr>
<tr>
<td>User core</td>
<td>How about Italian?</td>
<td>$continue @com.yelp.restaurant filter cuisine == 'Italian'</td>
</tr>
<tr>
<td>sentence (u)</td>
<td>(I want Italian restaurants)</td>
<td></td>
</tr>
<tr>
<td>DS = Apply (u, c)</td>
<td>How far is the best Italian restaurant?</td>
<td>[distance(geo, $location.current_location)] of (sort ratings desc of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>@com.yelp.restaurant filter cuisine == ‘Italian’)[1]</td>
</tr>
</tbody>
</table>

- core representation
  - Standard ThingTalk syntax + keyword $continue
  - Resolve co-references + minimal ellipses
  - To be predicted by the semantic parser

- Apply (u, c): a deterministic procedure
  - Apply context c to user statement u to include non-conflicting constraints in c.
Another Example

<table>
<thead>
<tr>
<th>Context (c)</th>
<th>English</th>
<th>ThingTalk</th>
</tr>
</thead>
<tbody>
<tr>
<td>I’d like a French restaurant in Menlo Park</td>
<td>@com.yelp.restaurant() filter cuisine == ‘French’ &amp;&amp; location == Location(‘Menlo Park’)</td>
<td></td>
</tr>
<tr>
<td>[Result: Left Bank]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User core sentence (u)</td>
<td>What is its address?</td>
<td>[address] of @com.yelp.restaurant()</td>
</tr>
<tr>
<td>[What is Left Bank’s address]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DS = Apply (u, c)</td>
<td>What is the address of Left Bank?</td>
<td>[address] of @com.yelp.restaurant() filter cuisine == ‘French’ &amp;&amp; location == Location(‘Menlo Park’)</td>
</tr>
<tr>
<td></td>
<td>English</td>
<td>ThingTalk</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Context (c)</td>
<td>I’d like a French restaurant in Menlo Park [Result: Left Bank]</td>
<td>@com.yelp.restaurant() filter cuisine == ‘French’ &amp;&amp; geo == Location(‘Menlo Park’)</td>
</tr>
<tr>
<td>User core</td>
<td>I want a cheap restaurant in Palo Alto</td>
<td>@com.yelp.restaurant() filter price == enum cheap &amp;&amp; geo == Location(‘Palo Alto’)</td>
</tr>
<tr>
<td>sentence (u)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- What does the user mean?
  - “a cheap restaurant in Palo Alto”
  - “a cheap French restaurant in Palo Alto”?
- What if there are no cheap French restaurants in Palo Alto?
## Best Answers for Ambiguous Queries: Use KB

<table>
<thead>
<tr>
<th>Context (c)</th>
<th>English</th>
<th>ThingTalk</th>
</tr>
</thead>
<tbody>
<tr>
<td>I’d like a French restaurant in Menlo Park [Result: Left Bank]</td>
<td>@com.yelp.restaurant() filter cuisine == ‘French’ &amp;&amp; geo == Location(‘Menlo Park’)</td>
<td></td>
</tr>
<tr>
<td>User core sentence (u)</td>
<td>I want a cheap restaurant in Palo Alto</td>
<td>@com.yelp.restaurant() filter price == enum cheap &amp;&amp; geo == Location(‘Palo Alto’)</td>
</tr>
<tr>
<td>DS = Apply (u, c)</td>
<td>I want a cheap French restaurant in Palo Alto</td>
<td>@com.yelp.restaurant() filter cuisine == ‘French’ &amp;&amp; price == enum cheap &amp;&amp; geo == Location(‘Palo Alto’)</td>
</tr>
<tr>
<td>DS* = Apply* (u, c)</td>
<td>[no cheap French rest.]</td>
<td>@com.yelp.restaurant() filter price == enum cheap &amp;&amp; geo == Location(‘Palo Alto’)</td>
</tr>
</tbody>
</table>

- **Apply* (u, c)** returns a non-null query, if one exists, that satisfies u and as many constraints from c as possible
# Generalizing Ambiguity

<table>
<thead>
<tr>
<th>Context (c)</th>
<th>English</th>
<th>ThingTalk</th>
</tr>
</thead>
<tbody>
<tr>
<td>How many Chinese restaurants are on the top 10 list in Palo Alto?</td>
<td>count((sort ratings desc of @com.yelp.restaurant() filter geo == Location('Palo Alto'))[1:10] filter cuisine == 'Chinese')</td>
<td></td>
</tr>
</tbody>
</table>

| User core sentence (u) | What is the average rating of the Italian ones? [the average rating of Italian restaurants] | avg ratings of (@com.yelp.restaurant() filter cuisine == 'Italian') |

| DS = Apply (u, c) | What is the average rating of the Italian restaurants on the top 10 list in Palo Alto? | avg ratings of ((sort ratings desc of @com.yelp.restaurant() filter geo == Location('Palo Alto'))[1:10] filter cuisine == 'Italian') |

| DS* = Apply* (u, c) | [No Italian in top 10 in PA, but there are Italian in PA] What is the average rating of the Italian restaurants in Palo Alto? | avg ratings of (@com.yelp.restaurant() filter geo == Location('Palo Alto')) && cuisine == 'Italian') |

| DS* = Apply* (u, c) | [No Italian in PA] What is the average rating of Italian Restaurants? | avg ratings of (@com.yelp.restaurant() filter cuisine == 'Italian') |
Hello, I am Genie, your restaurant assistant.

I’d like a French Restaurant.

I’d recommend Chez TJ in Mountain View. It has one Michelin star!

How far is it from here?

The distance to Chez TJ is 7.5 miles.

Can you recommend something nearby?

The distance to Left Bank is 2.8 miles.
Summary: Handling Ambiguous User Utterances

Quiz: Should we ask users what they mean?

• Assume the most restrictive version
  • It is correct regardless
  • Answer with enough context to avoid misunderstanding

• What if there are no answers to the query?
  • To be helpful, return an answer to the relaxed constraints
  • This requires looking up the database
Outline

• Conversational virtual assistants: User initiatives
  • What do task-oriented conversations look like?
  • Definition: Dialogue State Tracking (DST) Problem
  • Optimization: Use formal state in DST
  • Optimization: Simplify annotations of conversations
  • Synthesis overview
• Adding mixed initiatives
Recall: Abstract Dialogue States (Virtual Assistants)

Agent: executes these domain-independent state machines
Synthesize Conversations

- Generate many perfectly annotated conversations
  - Simulate the state machine
  - Synthesize user utterances with templates covering all constructs (Substituting parameters with actual values)
  - Execute: Generate arbitrary results
  - Synthesize agent response, also with templates (Used in agent’s policy)

- Note: after “apply”, the dialogue state has a lot variety
Future lecture

- Details on Apply
- Details on simulation
- Experimental results
Outline

- Conversational virtual assistants: User initiatives
  - What do task-oriented conversations look like?
  - Definition: Dialogue State Tracking (DST) Problem
  - Optimization: Use formal state in DST
  - Optimization: Simplify annotations of conversations
  - Synthesis overview
- Adding mixed initiatives
Mixed-Initiative Conversational Chats

• Agent Initiatives
  • When: at the beginning, when the user is done or pauses
    • [Greetings]. I can help you with flights, hotels, or car reservations. Would you like to book a flight today?
    • [Empty search] e.g. Italian restaurants. Would you like Japanese restaurants instead, there are some very good ones in the area.
    • [Follow-up Action] e.g. After a flight is booked. Would you like to get a car?
  • Relies on the developer for the policy
  • GenieScript primitives: InitiateQuery, InitiateActions
Abstract Dialogue States (Mixed Initiatives)

KB Navigation:
- Any Time
- InitiateQuery
  - Query (TT)
    - SlotFillQuestion
    - QueryResponse
      - Done
- Cancel

Action:
- InitiateAction
  - InvokeAction (TT)
    - requested
      - SlotFillQuestion
      - AsktoConfirm
    - confirmed
      - InvokeAction (TT)
        - ActionQuery (TT)
          - ActionQueryResponse
            - Done
Example: Restaurant Reservation Dialogues

Restaurant Skills

Knowledge Base (KB)

Restaurant [“restaurant”, “food place”]

id: Entity(Restaurant)
geo: Location

[“address”, “in #”, “near #”, “around #”]

price: Enum(cheap, moderate, expensive)

[“# -ly priced “, “#”]

Cuisines: Array(Entity(Cuisine))

[“# food”, “serves # food”]

Action

MakeReservation [“reserve #”, “book #”]

restaurant: Entity(Restaurant)

book_people: Number [min=1]

[“for #”, “for # people”]

book_day: Date [“for #”]

book_time: Time [“at #”, “for #”]

Code

main () {
    dlg.say (“Hello, I am Genie.”);
    // User profile: last had pizzas 20 days ago
    result = dlg.initiateQuery (@restaurant(), filter cuisine==pizza, “Are you interested in some pizzas today?”);
    if (result.status == “success”) {
        // Found a restaurant
        dlg.initiateAction (@MakeReservation() restaurant=’result1.id’, “Would you like me to book ’result1.name’?”);
        dlg.say (“Goodbye.”);
    }
}
## Example 1: Dialogue

<table>
<thead>
<tr>
<th>USER</th>
<th>GENIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello, I am Genie. Are you interested in some pizzas today?</td>
<td>I’d recommend Terun in Palo Alto.</td>
</tr>
<tr>
<td>Sure.</td>
<td>Would you like me to book Terun?</td>
</tr>
<tr>
<td>Sounds good. I heard it is good. (Done)</td>
<td>Which day?</td>
</tr>
<tr>
<td>OK</td>
<td>Could you confirm that you want a table for 2 people at Terun on November 2 at 7pm?</td>
</tr>
<tr>
<td>November 2, 7pm, for 2 people.</td>
<td>I booked Terun for 2 on November 2 at 7pm.</td>
</tr>
<tr>
<td>Yes.</td>
<td></td>
</tr>
</tbody>
</table>
Example 2: Dialogue

**GENIE**
Hello, I am Genie. Are you interested in some pizzas today?
I’d recommend Chez TJ in Mountain View. It has one Michelin star!

**USER**
How far is it from here?

**GENIE**
The distance to Chez TJ is 7.5 miles.

**USER**
Can you recommend something nearby?

**GENIE**
The distance to Left Bank is 2.8 miles.

**USER**
Perfect. (Done)

**GENIE**
Would you like me to book Left Bank?

**USER**
Is it expensive?

**GENIE**
The price is moderate.

**USER**
OK

**GENIE**
Would you like me to book Left Bank?

**USER**
Sure.

**GENIE**
Which day would you like to book Left Bank?

**USER**
November 2, 7pm, for 2 people.

**GENIE**
Could you confirm that you want a table for 2 people at Left Bank on November 2 at 7pm?

**USER**
Yes.

**GENIE**
I booked Left Bank for 2 on November 2 at 7pm.
dlg.initiate[Query | Action]

- dlg.initiate[Query | Action] (TT, agent-utterance)
- Puts proposed TT into the context for the semantic parsers
- Synthesizer
  - Uses templates to generate all kinds of agent initiations

```
main () {
    dlg.say ("Hello, I am Genie.");
    // User profile: last had pizzas 20 days ago
    result = dlg.initiateQuery (@restaurant(),filter cuisine==pizza,
                              "Are you interested in some pizzas today?");

    if (result.status == "success") {
        // Found a restaurant
        dlg.initiateAction (@MakeReservation() restaurant='result1.id',
                            "Would you like me to book 'result1.name'?);
        dlg.say ("Goodbye.");
    }
}
```
Another Example: Empty Search

- If the conversation does not identify a restaurant
  - Suggest one
  - Repeat

```plaintext
main () {
    dlg.say ("Hello, I am Genie.");
    // User profile: last had pizzas 20 days ago
    result = dlg.initiateQuery (@restaurant() filter cuisine==pizza,
                              "Are you interested in some pizzas today?");
    While (result.status == "failure") {
        request: LastAttemptedRequest ();
        newProposal = FindClosestRequestWithResult (request);
        result = dlg.initiateQuery (newProposal, agentProposalUtterance(newProposal));
    }
    if (result.status == "success") { // Found a restaurant
        dlg.initiateAction (  
            "@MakeReservation restaurant=result1.id",
            "Would you like me to book 'result1.name'?'");
        dlg.say ("Goodbye.");
    }
}
```
Other dlg Functions

- Intents for simple questions:
  - Yes, no: “Would you like to take a survey after the call?”
  - Small number of classes:
    “Is this statement positive, neutral, or negative

- Current GenieScript implementation: Regular expression
More User Utterances in Conversations

When users talk to a human, they communicate more
Understanding users beyond database queries:

1. **Discovery is extra hard in voice assistants!**
   Meta information about services offered (meta-skills)
   - What can you do? Could you tell me more?
   - What do you recommend?

2. **Beyond knowledge bases!**
   Answered by consulting with free-text or FAQ
   - To be covered in a different lecture
   - (Not implemented in Geniescript yet).
Summary of Task-Oriented Assistants

- Virtual assistants: Conversations are mostly commands and responses (Simple state machines)
- DST: dialogue state tracking (meaning of user utterances)
  - User utterances have co-references, ellipses, and ambiguity
  - Context grows with the conversation, prediction gets long
- Optimizations:
  - Use formal states rather than natural language in the context
  - Predict the core sentences
    - Core sentences: Minimal and complete ThingTalk sentences
    - Use deterministic “Apply*” which looks up DB to resolve ambiguity
- Mixed initiatives
  - Use dlg.InitiateQuery and dlg.InitiateAction to initiate discussions