Lecture 9

The Genie Dialogue Agent

1. Machine-to-machine simulation
2. Dialogue synthesis with a simulator
3. Simulation vs WOZ
4. Contextual semantic parsing
Back to Data Acquisition

• From hardest to easiest
  • Real dialogues: human-to-human
    • Often not possible, hard to annotate correctly)
  • Wizard-of-Oz: worker-to-worker
    • Expensive and hard to annotate correctly)
  • Machine-to-machine
    • Synthesize (Easier, no need to annotate)
    • Paraphrase: manual or automatic
Synthesis ➔ Paraphrase

• Schema-Guided Dialogue (SGD) Dataset [Rastogi et al 2019]

Data Acquisition

a. Generated scenario
b. Vary the expression of the value
c. Generation with 1 template
d. Crowdsourced paraphrase

Relies heavily on manual paraphrases!
### SGD Dataset

<table>
<thead>
<tr>
<th>Metric</th>
<th>DSTC2</th>
<th>WOZ2.0</th>
<th>FRAMES</th>
<th>M2M</th>
<th>MultiWOZ</th>
<th>SGD</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of domains</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>No. of dialogues</td>
<td>1,612</td>
<td>600</td>
<td>1,369</td>
<td>1,500</td>
<td>8,438</td>
<td>16,142</td>
</tr>
<tr>
<td>Total no. of turns</td>
<td>23,354</td>
<td>4,472</td>
<td>19,986</td>
<td>14,796</td>
<td>113,556</td>
<td>329,964</td>
</tr>
<tr>
<td>Avg. turns per dialogue</td>
<td>14.49</td>
<td>7.45</td>
<td>14.60</td>
<td>9.86</td>
<td>13.46</td>
<td>20.44</td>
</tr>
<tr>
<td>Avg. tokens per turn</td>
<td>8.54</td>
<td>11.24</td>
<td>12.60</td>
<td>8.24</td>
<td>13.13</td>
<td>9.75</td>
</tr>
<tr>
<td>Total unique tokens</td>
<td>986</td>
<td>2,142</td>
<td>12,043</td>
<td>1,008</td>
<td>23,689</td>
<td>30,352</td>
</tr>
<tr>
<td>No. of slots</td>
<td>8</td>
<td>4</td>
<td>61</td>
<td>13</td>
<td>24</td>
<td>214</td>
</tr>
<tr>
<td>No. of slot values</td>
<td>212</td>
<td>99</td>
<td>3,871</td>
<td>138</td>
<td>4,510</td>
<td>14,139</td>
</tr>
</tbody>
</table>

**Synthesis/Paraphrase:** many more dialogues than MultiWOZ

**Cost of paraphrases (including verification) is still a limiting factor**
Evaluation

- Evaluation set: synthesized then paraphrased
  - $\geq 95\%$ accuracy intent classification
  - $\geq 95\%$ F1 score slot tagging
F1 Score (Classification problem)

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Negative</td>
</tr>
<tr>
<td>Negative</td>
<td>True Negative</td>
</tr>
<tr>
<td>Positive</td>
<td>False Negative (fn)</td>
</tr>
</tbody>
</table>

Precision = \frac{True Positive}{True Positive + False Positive}

Is it true when the guess is true?

Recall = \frac{True Positive}{True Positive + False Negative}

How many "trues" can we get?

F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{tp}{tp + \frac{1}{2} (fp + fn)}
Evaluation

• Evaluation set: synthesized then paraphrased
  
  >= 95% accuracy intent classification
  
  >= 95% F1 score slot tagging

Quiz: Great results! Are we done?
### Two Worlds

<table>
<thead>
<tr>
<th></th>
<th>Machine-2-Machine</th>
<th>MultiWOZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Data Cost</td>
<td><strong>Moderate cost</strong></td>
<td><strong>High cost</strong> Error-prone</td>
</tr>
<tr>
<td></td>
<td>Synthesis + manual paraphrases</td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>Paraphrased</td>
<td>WOZ</td>
</tr>
<tr>
<td>Results</td>
<td>95% intent 95% F1</td>
<td>73% slot accuracy</td>
</tr>
</tbody>
</table>

We want to get an even lower annotation cost than M2M that can handle the realistic data in WOZ.
# Best of Both Worlds: Genie

<table>
<thead>
<tr>
<th></th>
<th>Machine-2-Machine</th>
<th>MultiWOZ</th>
<th>Genie (on MultiWOZ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Data</td>
<td>Moderate cost</td>
<td>High cost</td>
<td>Low cost</td>
</tr>
<tr>
<td>Cost</td>
<td>Synthesis + manual paraphrases</td>
<td>Error-prone</td>
<td>synthesis + few shot + self-training</td>
</tr>
<tr>
<td>Test</td>
<td>Paraphrased</td>
<td>WOZ</td>
<td>WOZ</td>
</tr>
<tr>
<td>Results</td>
<td>95% intent 95% F1</td>
<td>73% slot accuracy</td>
<td>87.5% slot accuracy 79.2% exact match</td>
</tr>
</tbody>
</table>
Outline

• Machine-to-machine simulation
• **Dialogue synthesis with a simulator**
• Simulation vs WOZ
• Contextual semantic parsing
Let’s See What We Know Now

• **ThingTalk** can be used to represent 98% of the turns in WOZ
  • Can we generate a good distribution with all the turns?

• We know how to generate user single commands (execute)

• How do we generate dialogues?
  1. Augment the agent and turn it into a simulator
  2. Simulate an **abstract state machine** (domain independent)
  3. Generate many dialogues using **templates** by combining **abstract states** with **domain schemas**, **results** so far
Simulator: Agent + User Loop

1. Contextual Semantic Parser (NLU): 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

5. User Policy: Generates user utterances

For Training
Can We Write One Program?

<table>
<thead>
<tr>
<th></th>
<th>Real Agent</th>
<th>Simulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Input</td>
<td>Real user input</td>
<td>Generated by user policy</td>
</tr>
<tr>
<td>Semantic Parser</td>
<td>Inferencing</td>
<td>Training</td>
</tr>
<tr>
<td>User State</td>
<td>Inference</td>
<td>From training sample</td>
</tr>
<tr>
<td>Execution</td>
<td>ThingTalk</td>
<td>ThingTalk</td>
</tr>
<tr>
<td>Agent Policy</td>
<td>1 outcome for each state</td>
<td>Multiple outcomes to simulate WOZ agent</td>
</tr>
<tr>
<td>NLG</td>
<td>Generate NL with a <strong>template</strong></td>
<td>-- NA --</td>
</tr>
<tr>
<td>User Policy</td>
<td>-- NA --</td>
<td>Many outcomes to simulate the user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Generate NL with a <strong>template</strong></td>
</tr>
</tbody>
</table>

A **template** is used to generate instances of natural language utterance & formal state for a given state.
function agent() {
    let state = initial();
    for (;;) {
        let cmd = get(); // call CSP
        execute(cmd); // call real APIs
        context = update(state, cmd); // concat context
        state = policy(context); // agent policy
        // use say to reply
    }
}

say: primitive for generating agent
NL and formal state with templates

The Agent+User Loop at Simulation Time

function agent() {
  let state = initial();
  for (;;) {
    let cmd = get();       // process one user input
    execute(cmd);          // call simulated APIs
    context = update(state, cmd); // concat context
    state = policy(context); // agent & user policy
    // call say to generate agent data
    // call expect to generate user data
  }
}

expect: generates user NL and formal state with a template

Outline

• Machine-to-machine simulation
• Dialogue synthesis with a simulator
  1. Augment the agent and turn it into a simulator
  2. *Simulate an abstract state machine (domain independent)*
  3. Generate many dialogues using templates
• Simulation vs WOZ
• Contextual semantic parsing
Simulator: Agent + User Loop

1. Contextual Semantic Parser (NLU): User utterance + context → user state
2. ThingTalk Execution: ThingTalk → result state
3. Agent policy: code to decide the response User context + result → agent state Natural Language Generator (NLG): agent state → agent utterance
4. Output Context: Summarizes the dialogue to date Becomes input context for next turn
5. User Policy: Generates user utterances
Abstract State Machine

Agent Transition

**Agent Policy**
- User & Result State → Agent State
- Agent State → Agent Utterance

User Transition

**User Policy**
- Output Context → User State
- User State → User Utterance
- Input Context = Output Context

Need 1 in a real agent; Need multiple to simulate WOZ

Need many to simulate users
Abstract States and Transitions

Result states are not shown for brevity

Campagna, Foryciarz, Moradshahi, Lam, ACL 2020
Abstract Agent Policy

• Code to decide the agent transition in a given agent context
• `say`: what the agent says abstractly using templates
• Examples:

  • Deterministically:
    ```
    if (state.results.length == 0)
      say(Templates.empty_search)
    else
      say(Templates.recommend_one)
    ```

  • Non-deterministically to simulate WOZ:
    ```
    either([
      say(Templates.search_question),
      say(Templates.recommend_one)
    ])
User Policy

• Embed user policy after each agent transition to decide on user transitions
  • Executed during simulation
  • Ignored during runtime
• expect: what we expect the user to say after the agent's response
• Example
  
say(Templates.recommend);
  expect([  
    Templates.change_search,
    Templates.ask_action,
    Templates.thanks
  ]);
Outline

• Machine-to-machine simulation
• Dialogue synthesis with a simulator
  1. Augment the agent and turn it into a simulator
  2. Simulate an abstract state machine (domain independent)
  3. Generate dialogues using templates
• Simulation vs WOZ
• Contextual semantic parsing
Training Data Generation

Abstract Dialogue State Machine

Restaurant
- id: Entity(Restaurant)
- geo: Location
  - [“address”, “in #”, “near #”, “around #”]
- price: Enum(cheap, moderate, expensive)
  - [# -ly priced “, “#”]
- cuisines: Array(Entity(Cuisine))
  - [# food”, “serves # food“]

MakeReservation
- “reserve #”, “book #”
- restaurant: Entity(Restaurant)
- book_people: Number [min=1]
  - [“for #”, “for # people”]
- book_day: Date [“for #”]
- book_time: Time [“at #”, “for #”]

Dialoge Training Data

StateResult
- Restaurant, price == moderate && geo == “Palo Alto”
  - { id = “Terun”, price = moderate, cuisines = [“pizza”], … }  
  - { id = “Coconuts”, price = moderate, cuisines = [“caribbean”], … }  

ProposeOne
- I have Terun. It’s a moderately priced restaurant that serves pizza.

ProposeN
- I found Terun and Coconuts. Both are moderately priced.

AskAction
- I like that. Can you help me book it? I need it for 3 people.

SearchRefine
- I don’t like pizza. Do you have something Caribbean?

InfoQuestion
- Can you tell me the address of Terun?

Formal representation not shown
Templates (part 1)

- Generate NL + formal state from a given user/agent state
- *Grammar* for generating the sentence
Templates: NL Generation

Grammar:

```
recommend_one =
    "I found $name. It is a $result" | "Would you like to $action $name?"
result = "$base_table with $property" | "$adjective $base_table"
```

```
base_table = "restaurant"
property = "$value food"
action = "book" | "reserve"
adjective = "cheap" | "moderate" | "expensive"
```

```
name = <run-time result> | <simulation-token>
value = <run-time result> | <simulation-token>
```

Results: dynamic terminal grammar rules
Simulation tokens are substituted in param augmentation
Templates (part 2)

- Generate NL + formal state from a given user/agent state
- **Grammar** for generating the sentence
- **Semantic Function**
  - Associated with each grammar rule to generate the formal state recursively
  - \((\text{state}, \text{param}, \text{phrase}) \rightarrow \text{Formal representation}\)
  - Includes sanity check to decide if transition is valid (i.e. meaningful/likely to occur)
Semantic Functions

```javascript
thanks = "\{" thank you \| thanks \} \{ this is enough \| that’s all \}"
   : (state) => makeSimpleState(state, ‘cancel’);

slot_fill = "\"what $param \{ would you like \| are you interested in \} ?\""
   : (state, param) => makeSimpleState(state, ‘sys_slot_fill’, param);

slot_fill_answer = "\"I would like $value \{ \| please \\} \""
   : (state, value) =>
      { if (type !== state.questionType) return null;
        return makeTargetState(state,
          addParam(state.next, state.dialogueActParam, value)); } 
```

Call a helper function to construct formal state

Sanity check

outstanding query
Complex Semantic Functions

`anything_phrase` = "{something | anything} $adjective" | "{something | anything} with $property"

`change_search` = "I don’t like that. do you have $anything_phrase?"

: (state, phrase) => {
  // get the filter in the current state (what the user asked before)
  // and the filter in the new phrase (what the user is asking now)
  const currentFilter = getFilter(state.current);
  const newFilter = getFilter(phrase);

  // the user must change something in the filter, or this template is not valid
  if (isSubset(newFilter, currentFilter)) return null;

  // merge what the user is asking in this turn, and what they asked before
  const merged = new BooleanExpression.And(currentFilter, newFilter)
    .optimize(); // remove redundant clauses

  // return the formal state
  return makeTargetState(replaceFilter(state.current, merged));
}
Synthesis: Output Training Data

WS = {"Init"}    // Working set containing partial dialogues

For each partial dialogue $d$ in WS:

- Simulate execution of ThingTalk code with random data
- Run policy
  - Generate instances of agent states (say)
  - Generate instances of user NLs and user states (expect)
- Candidate-set = concatenate each agent-user states to $d$

If the maximum turn is reached

{ Output candidate-set; } else

{ Add a sample of the candidate-set to WS;
  Output the rest; }
Outline

• Machine-to-machine simulation
• Dialogue synthesis with a simulator
  1. Turn the agent into a simulator
  2. Simulate an abstract state machine
  3. Generate many dialogues using templates
• **Simulation vs WOZ**
• Contextual semantic parsing
An Example from MultiWOZ

• U: Please book a table for 5 at 14:30 on Wednesday at Royal Spice. I also need to find a place to stay

• A: I was able to book your table successfully. Your reference number is kqmxil0z. Now, what type of accommodations are you looking for today?

Quiz: Can we represent this in ThingTalk?

Quiz: Can we synthesize this example?
Synthesis vs WOZ

- Multi-domain turns
- Domain switches
- Abandoned transactions

Quiz: Should we add these transitions to the abstract state machine?
We can’t synthesize all possible states exhaustively!
Quantitative Differences in MultiWOZ

- Between the validation data set and the synthesized data set

![Pie chart showing distribution]

- 68% Not Trained
- 15% Not Synthesizable
- 14% Not Representable
- 2% Trained
Quiz

• The semantic parser may correctly parse something that cannot be synthesized.
• What happens to the agent?
The Agent Loop at Runtime

```javascript
function agent() {
  let state = initial();
  for (;;) {
    let cmd = get(); // call CSP
    execute(cmd); // call real APIs
    context = update(state, cmd); // concat context
    state = policy(context); // agent policy
    // use say to reply
  }
} // The policy must handle all possible ThingTalk representations!
```

Quiz

• How to improve the semantic parser for unsynthesizable ThingTalk sentences?
Training Strategy for WOZ

- Add Few-shot manually annotated WOZ data
  - As realistic as possible
  - Encodes unexpected user behavior
  - Improves variability of natural language
  - Only need a small amount
Self-Training

• **Goal:** take advantage of *unannotated* WOZ dialogue data
• **Self-training:** Use the best parser so far to annotate new data
  • Train a semantic parser with synthesis + few shot
  • Use that to generate ThingTalk for unannotated data
• **Note:** WOZ has both user and agent sentences
  • Also need to train semantic parser to parse agent utterances
• **Annotation is not perfect**
  • Cheap way to use unannotated data
  • Empirically self-training helps
Outline

• Machine-to-machine simulation
• Dialogue synthesis with a simulator
  1. Turn the agent into a simulator
  2. Simulate an abstract state machine
  3. Generate many dialogues using templates
• Simulation vs WOZ
• Contextual semantic parsing
Contextual Semantic Parsing with BART

- Concatenate context and utterance into a single input
- BART output is directly the desired user state
Fine-tuning BART

• Fine-tune BART on triples of (context, utterance, user state)
• Treat ThingTalk as text (input and output)
  • Tokens are transformed to make them look like natural language (“price_range” → “price range”)

• No architecture changes
  • No new parameters
• Fine-tune all parameters
Synthesis + Few Shot

Domain Information

Data Synthesis

Augmentation

Automatic Paraphrasing

Fine-tuned BART

2\textsuperscript{nd} Fine-tuned Model

State Machine:
Policy Function + Templates

Value Datasets

Few-Shot Annotated Data

Fine-tuned BART
Synthesis + Few Shot + Self-Training

Domain Information

Data Synthesis

State Machine: Policy Function + Templates

Value Datasets

Augmentation

Automatic Paraphrasing

Fine-tuned BART

Few-Shot Annotated Data

2nd Fine-tuned Model

Self-training: Automatically Annotated Data

3rd Fine-tuned Model
Experiment: Applying ThingTalk to MultiWOZ

- MultiWOZ 3.0: reannotation of (partial) dev, test set with ThingTalk
- Abstract state machine
  - 20 agent transitions, 43 user transitions
- Training set:
  - 831k synthesized
  - ~1k turns few-shot (2% of original training)
  - 56k self-trained
    - We use the semantic parser to annotate the training data
Performance of Genie

Test Accuracy on MultiWOZ 3.0
• Annotate test set (7k turns), partial dev set (1K turns) in ThingTalk

<table>
<thead>
<tr>
<th>Training Set</th>
<th>Exact match</th>
<th>Slots only</th>
</tr>
</thead>
<tbody>
<tr>
<td>Few-shot only</td>
<td>73.7%</td>
<td>81.6%</td>
</tr>
<tr>
<td>Genie</td>
<td>79.2%</td>
<td>87.5%</td>
</tr>
</tbody>
</table>

Remember: SOTA slot-based accuracy: 73.6%
Full reannotation of MultiWOZ 2.4
Specialized STAR model
MultiWOZ 3.0 (ThingTalk)

Overall turn-by-turn exact match accuracy = 79% with few-shot annotations (2% of original)

![Graph showing turn-by-turn exact match accuracy and occurrence of different states.](image)
Conclusion

• **Write One Program**, both agent + simulator
  • Because ThingTalk is executable
    • Completeness also makes synthesis effective
  • Agent handles all ThingTalk (98% of MultiWOZ turns)
  • Simulator covers 84% of MultiWOZ turns

• **Sample-Efficient Training**
  • Synthesis, automatic paraphrase
  • Few shot (for NL for out-of-simulation states)
  • Self-trained (if unannotated data are available)

• **Contextual Semantic Parser**
  • Formal context eliminates reanalyzing history of dialogue
  • 79% state-by-state exact match accuracy on MultiWOZ