Lecture 6: End-to-end dialog approaches. Reinforcement learning in dialog

Original slides by Dan Jurafsky
Outline

• System architecture: Pipelined vs end-to-end (E2E)
• Deep learning in dialog components
• Reinforcement learning (RL) in end-to-end dialog systems
  • Quick introduction to RL
  • Recent research in end-to-end neural RL
Learning to do dialog management

- Dialog state management via frames and rules
- As dialog complexity grows:
  - More possible actions/responses after each user utterance
  - Many choices about how dialog management policy completes tasks
- Hand engineering complex dialog policies is hard
- Can we learn dialog management? By optimizing:
  - Correct response given an input
  - Task completion and user satisfaction of the full dialog
Dialog Act Markup in Several Layers (DAMSL): forward looking function

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATEMENT</td>
<td>a claim made by the speaker</td>
</tr>
<tr>
<td>INFO-REQUEST</td>
<td>a question by the speaker</td>
</tr>
<tr>
<td>CHECK</td>
<td>a question for confirming information</td>
</tr>
<tr>
<td>INFLUENCE-ON-ADDRESSEE (=Searle's directives)</td>
<td></td>
</tr>
<tr>
<td>OPEN-OPTION</td>
<td>a weak suggestion or listing of options</td>
</tr>
<tr>
<td>ACTION-DIRECTIVE</td>
<td>an actual command</td>
</tr>
<tr>
<td>INFLUENCE-ON-SPEAKER (=Austin's commissives)</td>
<td></td>
</tr>
<tr>
<td>OFFER</td>
<td>speaker offers to do something</td>
</tr>
<tr>
<td>COMMIT</td>
<td>speaker is committed to doing something</td>
</tr>
<tr>
<td>CONVENTIONAL</td>
<td>other</td>
</tr>
<tr>
<td>OPENING</td>
<td>greetings</td>
</tr>
<tr>
<td>CLOSING</td>
<td>farewells</td>
</tr>
<tr>
<td>THANKING</td>
<td>thanking and responding to thanks</td>
</tr>
</tbody>
</table>
MultiWOZ Dataset and Dialog State Tracking

Table 2: Full ontology for all domains in our data-set. The upper script indicates which domains it belongs to. *: universal, 1: restaurant, 2: hotel, 3: attraction, 4: taxi, 5: train, 6: hospital, 7: police.

<table>
<thead>
<tr>
<th>act type</th>
<th>inform* / request* / select/123 / recommend/123 / not found/123 / request booking info/123 / offer booking/1235 / inform booked/1235 / decline booking/1235 / welcome*/greet*/bye*/regmore*</th>
</tr>
</thead>
<tbody>
<tr>
<td>slots</td>
<td>address* / postcode* / phone* / name/1234 / no of choices/1235 / area/123 / pricerange/123 / type/123 / internet* / parking* / stars* / open hours* / departure* / destination* / leave after* / arrive by* / no of people/1235 / reference no./1235 / trainID* / ticket price* / travel time* / department* / day/1235 / no of days/123</td>
</tr>
</tbody>
</table>

Figure 1: A sample task template spanning over three domains - hotels, restaurants and booking.

MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling

Paweł Budzianowski\textsuperscript{1}, Tsung-Hsien Wen\textsuperscript{2}, Bo-Hsiang Tseng\textsuperscript{1}, Iñigo Casanueva\textsuperscript{2}, Stefan Ultes\textsuperscript{3}, Osman Ramadan\textsuperscript{1} and Milica Gašić\textsuperscript{1}

\textsuperscript{1}Department of Engineering, University of Cambridge, UK, \textsuperscript{2}PolyAI, London, UK
\{pfb30,mg436\}@cam.ac.uk
## MultiWOZ Example

Table 2: An example of partial dialog for MultiWOZ 2.1 dataset.

<table>
<thead>
<tr>
<th>Speaker</th>
<th>Utterance</th>
<th>Dialog States</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>I am leaving Cambridge. I need a train that leaves after 13:45.</td>
<td>train: {leaveAt: 13:45, departure: cambridge }</td>
</tr>
<tr>
<td></td>
<td>我要离开剑桥。我需要一列在13:45之后离开的火车。</td>
<td>火车: {出发时间: 13:45, 出发地: 剑桥}</td>
</tr>
<tr>
<td>System</td>
<td>On what day do you wish to travel?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>您想在哪一天旅行？</td>
<td></td>
</tr>
<tr>
<td>User</td>
<td>I am traveling on Wednesday and need to go to Birmingham New Street please.</td>
<td>train: {destination: birmingham new street, leaveAt: 13:45, departure: cambridge, day: wednesday }</td>
</tr>
<tr>
<td></td>
<td>我打算星期三旅行，我需要去伯明翰新街。</td>
<td>火车: {目的地: 伯明翰新街, 出发时间: 13:45, 出发地: 剑桥, 天: 星期三}</td>
</tr>
<tr>
<td>System</td>
<td>TR1766. It is going to take 163 minutes. can i book it for you?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>您可以乘坐TR1766。这将需要163分钟。需要我为您预订吗？</td>
<td></td>
</tr>
<tr>
<td>User</td>
<td>Yes, can you book 4 tickets for me?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>好的，你能帮我预订四张票吗？</td>
<td></td>
</tr>
<tr>
<td>User</td>
<td>Also looking for a place that has British food and moderately priced.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>我也正在寻找一个有英国美食且价格适中的地方。</td>
<td></td>
</tr>
<tr>
<td>System</td>
<td>I have many options for you. They are all in the city center. Do you have any other preferences?</td>
<td>train: {destination: birmingham new street, leaveAt: 13:45, departure: cambridge, day: wednesday, book: {people: 4, booked: {trainID: TR1766, reference: 6L0MANZV}}}</td>
</tr>
</tbody>
</table>

See: Dialogue System Technology Challenge
Dialog management in complex systems

- Representing state/actions for dialog management gets complex fast for systems beyond just form-filling

- Need to:
  - Decide when the user has asked a question, made a proposal, rejected a suggestion
  - Ground a user’s utterance, ask clarification questions, suggest plans

- Need to design representation and transitions
  - Model user mental state? More than just list of slots
  - Many possible responses at any turn (clarification? Move to new subtopic? Confirm/ground?)
Pipelined Conceptual Architecture

so far

- Speech Recognition
- Natural Language Understanding
- Dialogue Manager
- Task Manager
- Text-to-Speech Synthesis
- Natural Language Generation
Pipelined Conceptual Architecture
pros/cons

+ Clear component boundaries.
+ Train/eval each in isolation
  - Different datasets can help (e.g. large corpus for ASR)
  - Reuse high quality components when possible (ASR/TTS)
  - Per-component metrics and labels
+ Direct control over each component/interface
- Hand engineering dialog representations is hard!
- Interfaces force decisions about what information to pass between components. Often sub-optimal
- True metric is task completion / satisfaction. Unclear how optimizing component metrics affect overall performance
End-to-end Architectures

• What if we could learn an *end-to-end* system that directly generates actions and responses given user input?
  • + Eliminate hand engineered interfaces between components that limit information
  • + Optimize entire system for end goal vs per-component metrics/labels
  • - Potentially hard to debug, and end-to-end training might limit component/data transfer
• Especially important for multi-domain or personal assistant systems where many states/actions possible
Information-state

- discourse context
- beliefs
- goals
- user model
- task context

Natural Language Understanding

Dialogue Act Interpreter

Information State

Behavioral Agent
- update rules
- control

Natural Language Generation

Dialogue Act Generator

Speech
Dialog state tracking as a task

- Predict state (slot values + dialog act) from

<table>
<thead>
<tr>
<th>Flight Service A</th>
<th>Flight Service B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intents:</strong></td>
<td><strong>Intents:</strong></td>
</tr>
<tr>
<td>SearchFlight,</td>
<td>FindFlight,</td>
</tr>
<tr>
<td>ReserveFlight</td>
<td>ReserveFlight</td>
</tr>
<tr>
<td><strong>Slots:</strong></td>
<td><strong>Slots:</strong></td>
</tr>
<tr>
<td>origin,</td>
<td>origin,</td>
</tr>
<tr>
<td>destination,</td>
<td>destination,</td>
</tr>
<tr>
<td>num_stops,</td>
<td>num_stops,</td>
</tr>
<tr>
<td>depart,</td>
<td>depart,</td>
</tr>
<tr>
<td>return,</td>
<td>return,</td>
</tr>
</tbody>
</table>

**User**

Find **direct** round trip flights from **Baltimore** to **Seattle**.

**System**

Sure, what dates are you looking for?

**User**

Flying out **May 16** and returning **May 20**.

**System**

OK, I found a Delta flight for 302 dollars.

**User**

FindFlight: **depart** = **Baltimore**

**System**

arrive = **Seattle**

**direct_only** = **True**

FindFlight: **depart** = **Baltimore**

arrive = **Seattle**

**direct_only** = **True**

depart_date = **May 16**

return_date = **May 20**

Figure 2: Dialogue state tracking labels after each user utterance in a dialogue in the context of two different flight services. Under the schema-guided approach, the annotations are conditioned on the schema (extreme left/right) of the underlying service.

DSTC8 overview
Dialog state tracking

Figure 1: Dialogue System Architecture. Arrows illustrate retrieving/updating information from/to the blackboard. Boxes labeled GPT-2 (DST, POL, NLG) represent the same neural module which is invoked multiple times using different aggregated inputs. Dotted boxes contain symbols and double-line boxes contain natural language.

(Romero et al., 2021)
MultiWOZ Single domain example

<table>
<thead>
<tr>
<th>Dialogue id: “SNG0689”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goal</strong></td>
</tr>
<tr>
<td><strong>Domain:</strong> “restaurant”</td>
</tr>
<tr>
<td>Informable</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Requestable</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Book</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Database (restaurant)</th>
</tr>
</thead>
<tbody>
<tr>
<td>“ref”: “00000081”</td>
</tr>
<tr>
<td>“name”: “graffiti”</td>
</tr>
<tr>
<td>“phone”: “0123277977”</td>
</tr>
<tr>
<td>“postcode”: “cb30lx”</td>
</tr>
<tr>
<td>“address”: “Hotel Felix Whitehouse Lane Hunthidndon Road”</td>
</tr>
<tr>
<td>“food”: “british”</td>
</tr>
<tr>
<td>“area”: “west”</td>
</tr>
</tbody>
</table>

| “ref”: “00000084”     |
| “name”: “graffon hotel restaurant” |
| “phone”: “0123241387” |
| “postcode”: “cb580a”  |
| “address”: “Grafton Hotel 619 Newmarket Road Fen Ditton” |
| “food”: “british”     |
| “area”: “east”        |

**Dialogue turns**

[usr] Are there any restaurants that serve proper British food in town?  
[sys] Oh yes quite a few. Which part of town will you be dining in?  
[usr] **West**, if possible.  
[sys] We have three: Graffiti, Saint John’s Chop House, and Traveller.  
[usr] Will you give me the **phone number, address, and postcode** for Graffiti?  
[sys] The phone number is **0123277977**. The address is **Hotel Felix Whitehouse Lane Hunthidndon Road** and post code is **cb30lx**. Want to book?  
[usr] 6 people, please.  
[sys] Booking is successful. Reference number is **00000081**.

---

**Blue**: Informable slot  
**Yellow-Green**: Requestable slot name  
**Orange**: Requestable slot value

(Ham et al. ACL 2020)
End-to-end neural pipeline for goal-oriented dialogue systems using GPT2

Figure 2: The overview of our end-to-end neural dialogue model. For the transformer, we use fine-tuned GPT-2. The dashed line represents the information to and from the DB query, which is invoked when the system action needs to fetch an actual value from the database.

(Ham et al. ACL 2020)
End-to-end neural pipeline for goal-oriented dialogue systems using GPT2

- Key insights for other E2E neural approaches:
  - Output each module-specific inference using shared encoder
  - Fine-tune encoder from GPT-2 → Transfer learning

(Ham et al. ACL 2020)
End-to-end neural pipeline for goal-oriented dialogue systems using GPT2

<usr> I am looking for a place to stay that has cheap price range it should be in a type of hotel
<sys> Okay, do you have a specific area you want to stay in?

“metadata”: {“hotel”: {“semi”: “not mentioned”,
    “area”: “not mentioned”,
    “parking”: “not mentioned”,
    “pricerange”: “cheap”,
    “stars”: “not mentioned”,
    “internet”: “not mentioned”,
    “type”: “hotel”}}

⇒ Dialogue State

“dialog_act”: {“Hotel-Request”: {“Area”, “?”}}

⇒ System Action

<usr> no, I just need to make sure it’s cheap, oh, and I need parking

Figure 3: In the MultiWOZ dataset, the ‘metadata’ is treated as the dialogue state and the ‘dialogue act’ is treated as the system action.

Figure 4: Input representation for fine-tuning GPT-2.

(Ham et al. ACL 2020)
End-to-end neural pipeline for goal-oriented dialogue systems using GPT2

<table>
<thead>
<tr>
<th>Rank</th>
<th>Team ID</th>
<th>Success Rate ↑</th>
<th>Language Understanding ↑</th>
<th>Response Appropriateness ↑</th>
<th>Turns ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OURS(504430)</td>
<td>68.32%</td>
<td>4.149</td>
<td>4.287</td>
<td>19.507</td>
</tr>
<tr>
<td>2</td>
<td>504429</td>
<td>65.81%</td>
<td>3.538</td>
<td>3.632</td>
<td>15.481</td>
</tr>
<tr>
<td>3</td>
<td>504563</td>
<td>65.09%</td>
<td>3.538</td>
<td>3.840</td>
<td>13.884</td>
</tr>
<tr>
<td>4</td>
<td>504651</td>
<td>64.10%</td>
<td>3.547</td>
<td>3.829</td>
<td>16.906</td>
</tr>
<tr>
<td>5</td>
<td>504641</td>
<td>62.91%</td>
<td>3.742</td>
<td>3.815</td>
<td>14.968</td>
</tr>
<tr>
<td>N/A</td>
<td>Baseline</td>
<td>56.45%</td>
<td>3.097</td>
<td>3.556</td>
<td>17.543</td>
</tr>
</tbody>
</table>

Table 2: Overall results of the human evaluation carried out by DSTC8 organizers. Only the top five teams and the baseline results are compared.

<table>
<thead>
<tr>
<th>Model</th>
<th>Joint Acc.</th>
<th>Slot Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLAD</td>
<td>35.57</td>
<td>95.44</td>
</tr>
<tr>
<td>GCE</td>
<td>36.27</td>
<td>98.42</td>
</tr>
<tr>
<td>SUMBT</td>
<td>46.64</td>
<td>96.44</td>
</tr>
<tr>
<td>TRADE</td>
<td>48.62</td>
<td>96.92</td>
</tr>
<tr>
<td>OURS + greedy</td>
<td>44.03</td>
<td>96.07</td>
</tr>
</tbody>
</table>

Table 3: Performance comparison with other state-of-the-art models in Dialogue State Tracking benchmark of MultiWOZ dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Inform</th>
<th>Success</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASELINE (Budzianowski et al., 2018)</td>
<td>71.29</td>
<td>60.96</td>
<td>18.80</td>
</tr>
<tr>
<td>TOKENMOE (Pei et al., 2019)</td>
<td>75.30</td>
<td>59.70</td>
<td>16.81</td>
</tr>
<tr>
<td>HDSA (Chen et al., 2019)</td>
<td>82.9</td>
<td>68.90</td>
<td>23.60</td>
</tr>
<tr>
<td>STRUCTURED FUSION (Mehri et al., 2019)</td>
<td>82.70</td>
<td>72.10</td>
<td>16.34</td>
</tr>
<tr>
<td>LARL (Zhao et al., 2019)</td>
<td>82.78</td>
<td>79.20</td>
<td>12.80</td>
</tr>
<tr>
<td>OURS + greedy</td>
<td>77.00</td>
<td>69.20</td>
<td>6.01</td>
</tr>
</tbody>
</table>

Table 4: Performance comparison with other state-of-the-art models in Dialogue-Context-to-Text Generation benchmark of MultiWOZ dataset.

(Ham et al. ACL 2020)
Neural end-to-end trainable task-oriented dialogue systems

Figure 1: The proposed end-to-end trainable dialogue system framework

(Wen et al. EACL 2017)
Modeling a dialogue system as a probabilistic agent

- The current knowledge of the system
  - Set of states $S$ the agent can be in
- Set of actions $A$ the agent can take
- A success metric that tells us how well the agent achieved its goal
- A way of using this metric to learn a policy $\pi$ for what action to take in any state.
  (Reinforcement Learning)
What do we mean by actions $A$ and policies $\pi$?

- Kinds of decisions a conversational agent needs to make:
  - When should I ground/confirm/reject/ask for clarification on what the user just said?
  - When should I ask a directive prompt, when an open prompt?
  - When should I use user, system, or mixed initiative?
Markov Decision Processes

• Or MDP

• Characterized by:
  • a set of states $S$ an agent can be in
  • a set of actions $A$ the agent can take
  • A reward $r(a,s)$ that the agent receives for taking an action in a state

• Learn from human-human examples, or learn online by interactions

• More thorough RL intros: CS234 videos online. Berkeley CS285
What is a state?

- In principle, MDP state could include any possible information about dialogue
- Usually use a much more limited set due to compute limits
  - Values of slots in current frame
  - Most recent question asked to user
  - User’s most recent answer
  - ASR confidence, etc.
- In deep learning approaches, state might be implicit. NN predicts next action from NLU input + history/memory neural representations
MDP

- We can think of a dialogue as a trajectory in state space

$S_1 \rightarrow a_1, r_1 \quad \rightarrow a_2, r_2 \quad S_3 \rightarrow a_3, r_3 \ldots$

- The best policy $\pi^*$ is the one with the greatest expected reward over all trajectories
- How to compute a reward for a state sequence?
Reward for a state sequence

- Central RL theme: discounted rewards
- Cumulative reward $Q$ of a sequence is discounted sum of utilities of individual states

$$Q([s_0, a_0, s_1, a_1, s_2, a_2 \cdots]) = R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \cdots,$$

- Discount factor $\gamma$ between 0 and 1
- Makes agent care more about current than future rewards; the more future a reward, the more discounted its value
The Markov assumption

- MDP assumes that state transitions are Markovian

\[ P(s_{t+1} \mid s_t, s_{t-1}, \ldots, s_o, a_t, a_{t-1}, \ldots, a_o) = P_T(s_{t+1} \mid s_t, a_t) \]
Expected reward for an action

- Expected cumulative reward $Q(s,a)$ for taking a particular action from a particular state can be computed by Bellman equation:

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a')$$

- Expected cumulative reward for a given state/action pair is:
  - immediate reward for current state
  - $+$ expected discounted utility of all possible next states $s'$
  - Weighted by probability of moving to that state $s'$
  - And assuming once there we take optimal action $a'$
What we need for Bellman equation

- A model of $p(s' | s, a)$
- Estimate of $R(s, a)$

How to get these?

- If we had labeled training data
  - $P(s' | s, a) = C(s, s', a) / C(s, a)$
- If we knew the final reward for whole dialogue
  - $R(s_1, a_1, s_2, a_2, ..., s_n)$
- Given these parameters, can use value iteration algorithm to learn $Q$ values (pushing back reward values over state sequences) and hence best policy
How to estimate $p(s'|s,a)$ without labeled data

Have random conversations with real people:

- Carefully hand-tune small number of states and policies
- Then can build a dialogue system which explores state space by generating a few hundred random conversations with real humans
- Set probabilities from this corpus

Have random conversations with simulated people:

- Now you can have millions of conversations with simulated people
- Allows for larger state space
Neural dialog learning with human teaching and RL

Figure 1: Proposed end-to-end task-oriented dialogue system architecture.

(Liu et al. ACL 2020)
Neural dialog learning with human teaching and RL

Figure 2: Dialogue state and policy network.

\[ P(a_k | U_{\leq k}, A_{< k}, E_{\leq k}) = \text{PolicyNet}(s_k, v_k, E_k) \]  

\[(Liu\ et\ al.\ ACL\ 2020)\]
Neural dialog learning with human teaching and RL

Figure 3: Interactive learning curves on task success rate.

Figure 4: Interactive learning curves on average dialogue turn size.

Figure 5: Interactive learning curves on dialogue state tracking accuracy.

Figure 6: Interactive learning curves on task success rate with different RL training settings.

(Liu et al. ACL 2020)
E2E LSTM-based dialog control with RL

> Call Jason Williams
Office or cellphone?
> -

Figure 1: Operational loop. Green trapezoids refer to programmatic code provided by the software developer. The blue boxes indicate the recurrent neural network, with trainable parameters. The orange box performs entity extraction. The vertical bars in steps 4 and 8 are a feature vector and a distribution over template actions, respectively. See text for a complete description.
E2E LSTM-based dialog control with RL

Figure 5: Task completion rate (TCR) mean and standard deviation for a policy initially trained with \( N = (0, 1, 2, 5, 10) \) dialogs using supervised learning (SL), and then optimized with 0 to 10,000 dialogs using reinforcement learning (RL). Training and evaluation were done with the same stochastic simulated user. Each line shows the average of 10 runs, where the dialogs used in the SL training in each run were randomly sampled from the 21 example dialogs.
State of the art

1. Use deep learning approaches for ASR/TTS. Keep these modules separate
2. Build shared encoder with task outputs for intermediate state (NLU slots, actions, belief state, DB query info)
3. Training options (depends on availability):
   1. Initialize encoder with pre-trained weights (e.g. GPT2)
   2. Use supervised learning to optimize per-task outputs
   3. Use supervised/imitation learning to optimize end-to-end (match desired output/action and backprop all the way back)
   4. Interact with live/simulated users for reinforcement learning
Appendix
An example of dialogue act detection: Correction Detection

- If system misrecognizes an utterance, and either
  - Rejects
  - Via confirmation, displays its misunderstanding
- Then user has a chance to make a correction
  - Repeat themselves
  - Rephrasing
  - Saying “no” to the confirmation question.
Corrections

• Unfortunately, corrections are harder to recognize than normal sentences!
  • Swerts et al (2000): corrections misrecognized twice as often (in terms of WER) as non-corrections!!!
  • Why?
    • Prosody seems to be largest factor: hyperarticulation
    • Liz Shriberg example:
      • “NO, I am DE-PAR-TING from Jacksonville”
DAMSL: backward looking function

AGREEMENT  speaker's response to previous proposal
ACCEPT    accepting the proposal
ACCEPT-PART accepting some part of the proposal
MAYBE     neither accepting nor rejecting the proposal
REJECT-PART rejecting some part of the proposal
REJECT     rejecting the proposal
HOLD       putting off response, usually via subdialogue
ANSWER     answering a question
UNDERSTANDING  whether speaker understood previous
SIGNAL-NON-UNDER.  speaker didn't understand
SIGNAL-UNDER.  speaker did understand
ACK        demonstrated via continuer or assessment
REPEAT-REPHRASE  demonstrated via repetition or reformulation
COMPLETION demonstrated via collaborative completion
A DAMSL Labeling

[info-req,ack] A1: And, what day in May did you want to travel?
[assert, answer] C2: OK uh I need to be there for a meeting that’s from the 12th to the 15th.
[info-req,ack] A2: And you’re flying into what city?
[assert, answer] C3: Seattle.
[info-req,ack] A3: And what time would you like to leave Pittsburgh?
[check, hold] C4: Uh hmm I don’t think there’s many options for non-stop.
[assert] C5: There’s three non-stops today.
[info-req] A5: What are they?
[assert, open-option] C5: The first one departs PGH at 10:00am arrives Seattle at 12:05 their time. The second flight departs PGH at 5:55pm, arrives Seattle at 8pm. And the last flight departs PGH at 8:15pm arrives Seattle at 10:28pm.
[accept, ack] C6: OK I’ll take the 5ish flight on the night before on the 11th.
[check, ack] A6: On the 11th?
Older example


- NJFun system, people asked questions about recreational activities in New Jersey
- Idea of paper: use reinforcement learning to make a small set of optimal policy decisions
Very small # of states and acts

- States: specified by values of 8 features
  - Which slot in frame is being worked on (1-4)
  - ASR confidence value (0-5)
  - How many times a current slot question had been asked
  - Restrictive vs. non-restrictive grammar
- Result: 62 states

- Actions: each state only 2 possible actions
  - Asking questions: System versus user initiative
  - Receiving answers: explicit versus no confirmation.
Ran system with real users

- 311 conversations
- Simple binary reward function
  - 1 if competed task (finding museums, theater, winetasting in NJ area)
  - 0 if not
- System learned good dialogue strategy: Roughly
  - Start with user initiative
  - Backoff to mixed or system initiative when re-asking for an attribute
  - Confirm only a lower confidence values