6.891: Lecture 22 (December 1st, 2003)
Dialogue Systems
Overview

- A probabilistic approach to dialogue systems
- Reinforcement learning for dialogue strategies
A Probabilistic Dialogue System

- A fully statistical approach to natural language interfaces. [Miller, Stallard, Bobrow and Schwartz, 1996]
- Domain = ATIS (air travel reservations)
- An example dialogue:

User: Show me flights from Newark or New York to Atlanta, leaving tomorrow
System: <returns a list of flights>
User: When do the flights that leave from Newark arrive in Atlanta
System: <returns a list of times for the flights>
User: I’ll take the flight which arrives at 5.40 pm
System: <selects the requested flight>
The Task

- Map a sentence + context to a database query

User: Show me flights from Newark or New York to Atlanta, leaving tomorrow
System: <returns a list of flights>
User: When do the flights that leave from Newark arrive in Atlanta

⇒

<table>
<thead>
<tr>
<th>Show:</th>
<th>(Arrival-time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin:</td>
<td>(City “Newark”)</td>
</tr>
<tr>
<td>Destination:</td>
<td>(City “Atlanta”)</td>
</tr>
<tr>
<td>Date:</td>
<td>(November 27th, 2003)</td>
</tr>
</tbody>
</table>
Levels of Representation

- $W =$ input sentence
- $H =$ history (some representation of previous sentences)
- $T =$ a parse tree for $W$
- $F, S =$ a context-independent semantic representation for $W$ (i.e., $F, S$ is independent of the history $H$)
- $M =$ a context-dependent semantic representation for $W$ (i.e., $M$ depends on both $F, S$ and $H$)
Levels of Representation

- \( W \) = input sentence; \( H \) = history; \( T \) = a parse tree for \( W \); \( F, S \) = a context-independent semantic representation for \( W \); \( M \) = a context-dependent semantic representation for \( W \)

User: Show me flights from Newark or New York to Atlanta, leaving tomorrow
System: <returns a list of flights>
User: When do the flights that leave from Newark arrive in Atlanta

\( W \) = When do the flights that leave from Newark arrive in Atlanta
\( H \) =

Show: (flights)
Origin: (City “Newark”) or (City “New York”)
Destination: (City “Atlanta”)
Date: (November 27th, 2003)
Levels of Representation

- \( W = \) input sentence; \( H = \) history; \( T = \) a parse tree for \( W \); \( F, S = \) a context-independent semantic representation for \( W \); \( M = \) a context-dependent semantic representation for \( W \)

User: Show me flights from Newark or New York to Atlanta, leaving tomorrow
System: <returns a list of flights>
User: When do the flights that leave from Newark arrive in Atlanta

\( W = \) When do the flights that leave from Newark arrive in Atlanta
\( F, S = \)

```
Show: (Arrival-time)
Origin: (City "Newark")
Destination: (City "Atlanta")
```
\[ M = \text{a context-dependent semantic representation for } W \]

- **H** =
  - Show: (flights)
  - Origin: (City “Newark”) or (City “New York”)
  - Destination: (City “Atlanta”)
  - Date: (November 27th, 2003)

- **F, S** =
  - Show: (Arrival-time)
  - Origin: (City “Newark”)
  - Destination: (City “Atlanta”)

- **M** =
  - Show: (Arrival-time)
  - Origin: (City “Newark”)
  - Destination: (City “Atlanta”)
  - Date: (November 27th, 2003)
A Parse Tree

- Each non-terminal has a syntactic and semantic tag, e.g., flight/np
A Parse Tree (Continued)

flight /np
   /det
   the
   /corenp
   flight
   /np-head
   flights

flight-constraints /rel-clause
   /comp
   that
   departure /vp

   departure /vp-head
   leave

   departure /pp
   from
   city /npr
   Newark
Building a Probabilistic Model

- Basic goal: build a model of

\[ P(M \mid W, H) \]

i.e., probability of a context-dependent interpretation, given a sentence and a history

- We’ll do this by building a model of

\[ P(M, W, F, T, S \mid H) \]

giving

\[ P(M, W \mid H) = \sum_{F, T, S} P(M, W, F, T, S \mid H) \]

and

\[ \text{argmax}_M P(M \mid W, H) = \text{argmax}_M P(M, W \mid H) \]

\[ = \text{argmax}_M \left( \sum_{F, T, S} P(M, W, F, T, S \mid H) \right) \]
Building a Probabilistic Model

- Our aim is to build a model of
  \[ P(M, W, F, T, S \mid H) \]

- Step 1: chain rule (an exact step)
  \[ P(M, W, F, T, S \mid H) = P(F \mid H)P(T, W \mid F, H)P(S \mid T, W, F, H)P(M \mid S, T, W, F, H) \]

- Step 2: independence assumptions (an assumption/approximation)
  \[ P(M, W, F, T, S \mid H) = \underbrace{P(F)P(T, W \mid F)P(S \mid T, W, F)} \times \underbrace{P(M \mid S, F, H)} \]
  Sentence processing model  Contextual processing model
More About the Model

- The model:

\[ P(M, W, F, T, S \mid H) = P(F) P(T, W \mid F) P(S \mid T, W, F) \times P(M \mid S, F, H) \]

Sentence processing model \hspace{1cm} Contextual processing model

- The **sentence processing model** is a model of

\[ P(T, W, F, S) \]

i.e., a joint model of a sentence, its parse tree, and its “semantics”. Maps \( W \) to a \((F, S, T)\) triple (a context-independent interpretation)

- The **contextual processing model** goes from a \((F, S, H)\) triple to a final interpretation, \( M \)
$M = a \text{ context-dependent semantic representation for } W$

\[
H = \begin{aligned}
\text{Show: (flights)} \\
\text{Origin: (City “Newark”) or (City “New York”)} \\
\text{Destination: (City “Atlanta”)} \\
\text{Date: (November 27th, 2003)}
\end{aligned}
\]

\[
F, S = \begin{aligned}
\text{Show: (Arrival-time)} \\
\text{Origin: (City “Newark”)} \\
\text{Destination: (City “Atlanta”)}
\end{aligned}
\]

\[
M = \begin{aligned}
\text{Show: (Arrival-time)} \\
\text{Origin: (City “Newark”)} \\
\text{Destination: (City “Atlanta”)} \\
\text{Date: (November 27th, 2003)}
\end{aligned}
\]
The Sentence Processing Model

\[ P(T, W, F, S) = P(F)P(T, W | F)P(S | T, W, F) \]

- First step: choose the frame \( F \) with probability 
  \[ P(F) \]

- e.g., choose the frame

  \[
  \begin{array}{l}
  \text{Show: (Arrival-time)} \\
  \text{Origin:} \\
  \text{Destination:}
  \end{array}
  \]

- Note: there are a relatively small number of possible frames
The Sentence Processing Model

\[ P(T, W, F, S) = P(F) P(T, W \mid F) P(S \mid T, W, F) \]

- Next step: generate the parse tree and sentence
- Method uses a probabilistic context-free grammar, where Markov processes are used to generate rules. Different rule parameters are used for each value of F
\begin{equation}
P(\text{/det} \text{ flight/corenp} \text{ flight-constraints/rel-clause} | \text{flight/np}) \\
= P(\text{/det} | \text{NULL, flight/np}) \\
\times P(\text{flight/corenp} | \text{/det, flight/np}) \\
\times P(\text{flight-constraints/relclause} | \text{flight/corenp, flight/np}) \\
\times P(\text{STOP} | \text{flight-constraints/relclause, flight/np})
\end{equation}
Smoothing

\[ P(\text{flight/corenp} | /\text{det, flight/np}) \]
\[ = \lambda \times P_{ML}(\text{flight/corenp} | /\text{det, flight/np}) \]
\[ + (1 - \lambda) \times P_{ML}(\text{flight} | \text{flight}) P_{ML}(\text{corenp} | \text{np}) \]

- \(P_{ML}\) are maximum-likelihood estimates, e.g.,

\[ P_{ML}(\text{corenp} | \text{np}) = \frac{\text{Count}(\text{corenp, np})}{\text{Count}(\text{np})} \]

- \(\lambda\) is a smoothing coefficient between 0 and 1

- Backed-off estimates generate semantic, syntactic parts of each label separately
The Sentence Processing Model

\[ P(T, W, F, S) = P(F) P(T, W | F) P(S | T, W, F) \]

- Next step: given a frame \( F \), and a tree \( T \), fill in the semantic slots \( S \)

| Show: (Arrival-time) | \( \Rightarrow \) | Show: (Arrival-time) |
| Origin:              | \( \Rightarrow \) | Origin: Newark       |
| Destination:         |                | Destination: Atlanta |

- Method works by considering each node of the parse tree \( T \), and applying probabilities

\[ P(\text{slot-fill-action} | S, \text{node}) \]
The Sentence Processing Model

\[ P(T, W, F, S) = P(F)P(T, W | F)P(S | T, W, F) \]

• Next problem: search

• Goal: produce \( n \) high probability \((F, S, T, W)\) tuples

• Method:
  
  – In first pass, produce \( n \)-best parses under a parsing model that is independent of \( F \)
  
  – For each tree \( T \), for each possible frame \( F \), create a \((W, T, F)\) triple with probability \( P(T, W | F) \). Keep the top \( n \) most probable triples.
  
  – For each triple, use beam search to generate several high probability \((W, T, F, S)\) tuples. Keep the top \( n \) most probable.
The Contextual Model

\[ P(M \mid S, F, H) \]

Contextual processing model

\[
\begin{array}{l}
H = \\
\quad \text{Show: (flights)} \\
\quad \text{Origin: (City “Newark”) or (City “New York”)} \\
\quad \text{Destination: (City “Atlanta”)} \\
\quad \text{Date: (November 27th, 2003)} \\
\end{array}
\]

\[
\begin{array}{l}
(F, S) = \\
\quad \text{Show: (Arrival-time)} \\
\quad \text{Origin: (City “Newark”)} \\
\quad \text{Destination: (City “Atlanta”)} \\
\end{array}
\]

Only issue is whether values in \(H\), but not in \((F, S)\), should be carried over to \(M\).

⇒ Two possible values for \(M\):

\[
\begin{array}{l}
\quad \text{Show: (Arrival-time)} \\
\quad \text{Origin: (City “Newark”)} \\
\quad \text{Destination: (City “Atlanta”)} \\
\end{array}
\]

\[
\begin{array}{l}
\quad \text{Show: (Arrival-time)} \\
\quad \text{Origin: (City “Newark”)} \\
\quad \text{Destination: (City “Atlanta”)} \\
\quad \text{Date: (November 27th, 2003)} \\
\end{array}
\]
The Contextual Model

- Method uses a decision-tree model to estimate probability of “carrying over” each slot in $\mathbf{H}$ which is not in $\mathbf{F}, \mathbf{S}$. 
The Final Model

- Final search method:
  - Given an input sentence $W$, use the sentence processing model to produce $n$ high probability ($F$, $S$, $T$, $W$) tuples. Call these $(F_i, S_i, T_i, W)$ for $i = 1 \ldots n$
  - Choose the final interpretation as
    \[
    \arg\max_M \left( \max_i P(M \mid F_i, S_i, H) P(F_i, S_i, T_i, W) \right)
    \]
- Note that this is an approximation to finding
  \[
  \arg\max_M P(M, W \mid H) = \arg\max_M \sum_{F, S, T} P(M, W, F, S, T \mid H)
  \]
- Results: training on 4000 sentences, achieved an error rate of 21.6%
Reinforcement Learning for Dialogue Management


- Uses reinforcement learning to learn a high-level dialogue strategy
System: Welcome to NJFun. How may I help you?
User: I’d like to find um winetasting in Lambertville in the morning
System: Did you say you are interested in Lambertville?
User: Yes
System: Did you say you want to go in the morning?
User: Yes
System: I found a winery near Lambertville that is open in the morning. It is [. . .]. Please give me feedback by saying ’good’, ’so-so’ or ’bad’
User: Good.
The NJFun System

- Three attributes needed: activity, location, time
e.g., wine-tasting, Lambertville, Sunday

- Basic strategy: first get activity attribute, then location, finally time, then make a database query.
Dialogue Strategies

At any point in the dialogue, the following choices can be made:

- **System initiative vs. user initiative**
  - **System initiative:** Welcome to NJFun. Please say an activity name or say ‘list activities’ for a list of activities I know about.
  - **User initiative:** Welcome to NJFun. How may I help you?

- **Confirmation/no confirmation of attribute values**
  - **Confirmation:** Did you say you are interested in Lambertville?
The Abstract Model

- We have a set of possible states, $S$

- For each state $s \in S$, there is a set of possible actions, $A(s)$.

- Given an action $a$ in state $s$, the probability of transitioning to state $s'$ is $P(s' \mid s, a)$

- For a state-action pair $(s, a)$, the *reward* received is $R(s, a)$.
  (e.g., in Litman et. al, $R(s, a) = 1$ if the action leads to the dialogue being successfully completed, $R(s, a) = 0$ otherwise)

- A dialogue is a sequence of $n$ state/action pairs, 
  $(s_1, a_1), (s_2, a_2) \ldots (s_n, a_n)$
Why Reinforcement Learning?

- Problem is to learn a mapping from states to actions

- Why isn’t this a regular supervised learning problem?

- The reward is **delayed**: we might take several actions in sequence, and the only supervised information comes at the end of the dialogue (success or failure) ⇒ we need to infer the utility of each action in each state from this *indirect* or *delayed* form of supervision
Policies

- A policy $\pi : S \rightarrow A$ is a function that maps states to actions

- Define

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

- $Q(s, a)$ is the expected reward when action $a$ is taken in state $s$

- If $P(s' | s, a)$ is known, $Q(s, a)$ can be calculated, and optimal policy is

$$\pi(s) = \arg\max_a Q(s, a)$$

Main point: if $P(s' | s, a)$ can be learned from training examples, then optimal policy can be computed
Learning in this Model

• User builds the skeleton of a dialogue system:
  – A set of possible states
  – A set of possible actions in each state

• Training stage:
  – Interact with a user, with a random choice of actions in each state
  – Result: a training set of example dialogues 
    \(((s_1, a_1), (s_2, a_2), \ldots, (s_n, a_n)\) sequences)
  – From these sequences, estimate \(P(s' | s, a)\),
    and compute the optimal policy
States in the Dialogue System

14 attributes representing state of the system:

- Has the system greeted the user?

- Which attribute is the system trying to obtain? (activity, location or time)

- For each of the 3 attributes (activity, location, time):
  - Has the system obtained the attribute’s value?
  - What is the system’s confidence in the attribute’s value?
  - Number of times the system has asked about the attribute
  - Type of speech recognition grammar most recently used in the attribute query
Creating a Small Set of Possible States

<table>
<thead>
<tr>
<th>feature</th>
<th>greet</th>
<th>attr</th>
<th>conf</th>
<th>val</th>
<th>times</th>
<th>gram</th>
<th>hist</th>
</tr>
</thead>
<tbody>
<tr>
<td>possible values</td>
<td>0,1</td>
<td>1,2,3,4</td>
<td>0,1,2,3,4</td>
<td>0,1</td>
<td>0,1,2</td>
<td>0,1</td>
<td>0,1</td>
</tr>
</tbody>
</table>

- **greet** = 0 if user has been greeted, 1 otherwise
- **attr** represents attribute being queried; 1/2/3 = activity/location/time, 4 = done with attributes
- **conf** represents confidence in the attribute value. 0, 1, 2 = low/middle/high confidence in the speech recogniser; 3 = recognition system has received “YES” as an answer to a confirmation; 4 = system has received “NO”
- **val** = 1 if attribute value has been obtained, 0 otherwise
- **times** = number of times system has asked about the attribute
- **gram** = type of grammar used to obtain the attribute value
- **hist** = 0 if system has had problems in understanding the user earlier in the conversation; 1 otherwise
• An example state: 1240101

• In total, there are 62 possible states
Actions in the System

Possible choices:

- Greeting vs. asking user about activity/location/time

- Type of prompt: user initiative vs. system initiative. e.g.,

  **System:** I know about amusement parks, aquariums, cruises, ... Please say a name from the list.

  **User:** Please tell me the activity type. You can also tell me the location and time

- Type of grammar used in the speech recogniser: restrictive vs. non-restrictive
**System initiative:** I know about amusement parks, aquariums, cruises, . . . Please say a name from the list.

⇒ use a speech recogniser grammar which only allows items from the list

**User initiative:** Please tell me the activity type. You can also tell me the location and time

⇒ use a speech recogniser grammar with a much broader set of possible utterances
Actions in the System

- **Choices:** Greeting vs. asking user about activity vs. asking user about location (time is always queried using a direct question). Type of prompt: user initiative vs. system initiative. Type of grammar used: restrictive vs. non-restrictive.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GreetS</td>
<td>attribute=greeting, system initiative</td>
</tr>
<tr>
<td>GreetU</td>
<td>attribute=greeting, user initiative</td>
</tr>
<tr>
<td>ReAsk1S</td>
<td>attribute=activity, system initiative, restrictive grammar</td>
</tr>
<tr>
<td>Ask2U</td>
<td>attribute=location, user initiative, unrestrictive grammar</td>
</tr>
</tbody>
</table>
Informally, the NJFun dialogue manager selects the next query to ask the user regarding the activity, location, or event (with which can assume values morning, afternoon, or evening).

The examples in Figure 2 show that NJFun can ask the user about the first attribute using three types of initiative, based on the combination of the wording of the system prompt (open versus direct), and the type of grammar NJFun uses during ASR (restrictive versus non-restrictive). NJFun can use an open question with an unrestricted grammar (e.g., GreetU) if NJFun instead uses a directive prompt with a restricted grammar (e.g., GreetS). If NJFun uses an open question with an unrestricted grammar, NJFun can use mixed initiative (e.g., ReAsks), and if NJFun is using user initiative (e.g., ReAsks), NJFun uses a directive question with a non-restrictive grammar.

### Figure 2: Sample initiative strategy choices.

<table>
<thead>
<tr>
<th>Prompt</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Welcome to NJFun. How may I help you?</td>
<td>Greet</td>
</tr>
<tr>
<td>Welcome to NJFun. Please say an activity name or say “I know about none of these activities.”</td>
<td>Prompt</td>
</tr>
<tr>
<td>Please tell me the town or city that you are interested in.</td>
<td>ReAsks3</td>
</tr>
<tr>
<td>Please give me more information.</td>
<td>ReAsks3</td>
</tr>
<tr>
<td>Please say the name of the town or city that you are interested in.</td>
<td>ReAsks3</td>
</tr>
<tr>
<td>Please tell me the activity type. You can also tell me the location and time.</td>
<td>ReAsks3</td>
</tr>
<tr>
<td>Welcome to NJFun. How may I help you?</td>
<td>Greet</td>
</tr>
<tr>
<td>Welcome to NJFun. Please say an activity name or say “I know about none of these activities.”</td>
<td>Prompt</td>
</tr>
<tr>
<td>Figure 2: Sample initiative strategy choices.</td>
<td></td>
</tr>
</tbody>
</table>
An Example

- Initial state is always

<table>
<thead>
<tr>
<th>greet</th>
<th>attr</th>
<th>conf</th>
<th>val</th>
<th>times</th>
<th>gram</th>
<th>hist</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Possible actions in this state:

  **GreetU:** Welcome to NJFun. How may I help you?

  **GreetS:** Welcome to NJFun. Please say an activity name or say ’list activities’ for a list of activities I know about

  **In this state, system learns that GreetU is optimal action**

- Results in the following reply from the user:

  **System:** Welcome to NJFun. How may I help you?
  **User:** I’d like to find um winetasting in Lambertville in the morning
An Example

System: Welcome to NJFun. How may I help you?
User: I’d like to find um winetasting in Lambertville in the morning

- At this point, state is

<table>
<thead>
<tr>
<th>greet</th>
<th>attr</th>
<th>conf</th>
<th>val</th>
<th>times</th>
<th>gram</th>
<th>hist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(user has been greeted, current attribute is activity, confidence in answer = 2, val = 1 (activity value has been obtained) etc.)

- Possible actions in this state:

  **ExpConf1**: Did you say you are interested in winetasting?
  
  **NoConf**: say nothing, move directly to the state

<table>
<thead>
<tr>
<th>greet</th>
<th>attr</th>
<th>conf</th>
<th>val</th>
<th>times</th>
<th>gram</th>
<th>hist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

In this state, system learns that NoConf is optimal action
The Complete Dialogue

System: Welcome to NJFun. How may I help you?
User: I’d like to find um winetasting in Lambertville in the morning
System: Did you say you are interested in Lambertville?
User: Yes
System: Did you say you want to go in the morning?
User: Yes
System: I found a winery near Lambertville that is open in the morning. It is [...].
Please give me feedback by saying ’good’, ’so-so’ or ’bad’
User: Good.

<table>
<thead>
<tr>
<th>State</th>
<th>attr</th>
<th>conf</th>
<th>val</th>
<th>times</th>
<th>gram</th>
<th>hist</th>
<th>Action</th>
<th>Turn</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>GreetU</td>
<td>S1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NoConf</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>ExpConf2</td>
<td>S2</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>ExpConf2</td>
<td>S3</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Tell</td>
<td>S4</td>
<td>1</td>
</tr>
</tbody>
</table>
Experiments

- Each user asked to solve a particular task:
  e.g., *You feel thirsty and want to do some winetasting in the morning. Are there any wineries close by your house in Lambertville?*

- Collected 311 complete dialogues
  Randomly picked between possible actions in each state

- 54/62 states had more than 10 training examples
  Used examples to compute the optimal dialogue *policy*

- Gathered 124 complete test dialogues under the optimal strategy

- Performance: 64% task completion in test (i.e., under the computed policy,
  52% task completion in training phase (i.e., under the randomized policy)