Dialogue and Conversational Agents

Regina Barzilay
MIT

December, 2005
Outline

- Statistical NLU component
- Reinforcement learning for dialogue management
- Planning-based agent system
Statistical NLU component

- A fully statistical approach to natural language interfaces
- Task: map a sentence + context to a database query

User: Show me flights from NY to Boston, leaving tomorrow
System: [returns a list of flights]

<table>
<thead>
<tr>
<th>Show:</th>
<th>(Arrival-time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin</td>
<td>(City &quot;NY&quot;)</td>
</tr>
<tr>
<td>Destination</td>
<td>(City &quot;Boston&quot;)</td>
</tr>
<tr>
<td>Date:</td>
<td>(November 27th, 2003)</td>
</tr>
</tbody>
</table>
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 $W$
- $M =$ a context-dependent representation for $W$ ($M$ depends on both $F, S$ and $H$)
Example

\( W = \) 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

<table>
<thead>
<tr>
<th>Show:</th>
<th>(flights)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin</td>
<td>(City ”NY“) or (City ”NY“)</td>
</tr>
<tr>
<td>Destination:</td>
<td>(City ”Atlanta”)</td>
</tr>
<tr>
<td>Date:</td>
<td>(November 27th, 2003)</td>
</tr>
</tbody>
</table>
Example

\[ W = \text{input sentence}; \ H = \text{history}; \ T = \text{a parse tree for } W; \ F, S = \text{a context independent semantic representation for } W; \ M = \text{a context-dependent semantic representation for } W \]

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

\[ W = \text{When do the flights that leave from Newark arrive in Atlanta} \]

\[
\begin{array}{|c|c|}
\hline
\text{Show:} & (\text{Arrival-time}) \\
\hline
\text{Origin} & (\text{City “Newark”}) \\
\hline
\text{Destination:} & (\text{City ”Atlanta”}) \\
\hline
\end{array}
\]
### Example

<table>
<thead>
<tr>
<th>H=</th>
<th>Show: (flights)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Origin: (City &quot;NY&quot;) or (City &quot;NY&quot;)</td>
</tr>
<tr>
<td></td>
<td>Destination: (City &quot;Atlanta&quot;)</td>
</tr>
<tr>
<td>Date: (November 27th, 2003)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>E,S=</th>
<th>Show: (Arrival-time)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Origin: (City &quot;Newark&quot;)</td>
</tr>
<tr>
<td></td>
<td>Destination: (City &quot;Atlanta&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M=</th>
<th>Show: (Arrival-time)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Origin: (City &quot;Newark&quot;)</td>
</tr>
<tr>
<td></td>
<td>Destination: (City &quot;Atlanta&quot;)</td>
</tr>
<tr>
<td>Date: (November 27th, 2003)</td>
<td></td>
</tr>
</tbody>
</table>
A Parse Tree

Each non-terminal has a syntactic and semantic tag, e.g., city/npr
Building a Probabilistic Model

- Basic goal: build a model of $P(M|W, H)$ – probability of a context-dependent interpretation, given a sentence and a history
- We’ll do this by building a model of $P(M, W, F, T, S|H)$, giving

$$P(M, W|H) = \sum_{F, T, S} P(M, W, F, T, S|H)$$

and

$$argmax_M P(M|W, H) = argmax_M P(M, W|H) = argmax_M \sum_{F, T, S} P(M, W, F, T, S|H)$$
Building a Probabilistic Model

Our aim is to estimate \( P(M, W, F, T, S|H) \)

- Apply Chain rule:
  \[
  \]

- Independence assumption:
  \[
  \]
Building a Probabilistic Model

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

- The sentence processing model is a model of 
  \( P(T, W, F, S) \). Maps \( W \) to \( (F, S, T) \) triple (a context-independent interpretation)
- The contextual processing model goes from a \( (F, S, H) \) triple to a final interpretation, \( M \)
Example

Show: (flights)
Origin (City "NY") or (City "NY")
Destination: (City "Atlanta")
Date: (November 27th, 2003)

Show: (Arrival-time)
Origin (City "Newark")
Destination: (City "Atlanta")

Show: (Arrival-time)
Origin (City "Newark")
Destination: (City "Atlanta")
Date: (November 27th, 2003)
Building a Probabilistic 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 \( P(F) \)

Show: (Arrival-time)

Origin

Destination:
The Sentence Processing Model

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

- Next step: generate the parse tree \( T \) and sentence \( W \)
- Method uses a probabilistic context-free grammar, where Markov processes are used to generate rules. Different rule parameters are used for each value of \( F \)
The Sentence Processing Model

\[
P(\text{det flight/corenp flight-constraint/rel-clause|flight/np}) = P(\text{det|NULL, flight/np}) \times P(\text{flight/corenp|det,flight/np}) \times P(\text{flight-constraint|rel-clause|flight/corenp,flight/np}) \times P(\text{STOP|flight-constraint/rel-clause,flight/np})
\]

- Use maximum likelihood estimation

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

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

• Given a frame $F$, and a tree $T$, fill in the semantic slots $S$

<table>
<thead>
<tr>
<th>Show:</th>
<th>(Arrival-time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin</td>
<td></td>
</tr>
<tr>
<td>Destination:</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Show:</th>
<th>(Arrival-time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin</td>
<td>Newark</td>
</tr>
<tr>
<td>Destination:</td>
<td>Atlanta</td>
</tr>
</tbody>
</table>

• Method works by considering each node of the parse tree $T$, and applying probabilities $P$(slot-fill-action|S,node)
The Sentence Processing Model: Search

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

- **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

<table>
<thead>
<tr>
<th>H=</th>
<th>Show: (flights)</th>
<th>Origin: (City &quot;NY&quot;) or (City &quot;NY&quot;)</th>
<th>Destination: (City &quot;Atlanta&quot;)</th>
<th>Date: (November 27th, 2003)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F,S=</td>
<td>Show: (Arrival-time)</td>
<td>Origin: (City &quot;Newark&quot;)</td>
<td>Destination: (City &quot;Atlanta&quot;)</td>
<td></td>
</tr>
<tr>
<td>M=</td>
<td>Show: (Arrival-time)</td>
<td>Origin: (City &quot;Newark&quot;)</td>
<td>Destination: (City &quot;Atlanta&quot;)</td>
<td>Date: (November 27th, 2003)</td>
</tr>
</tbody>
</table>
The Contextual Model

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

  \[
  \begin{array}{|l|c|}
  \hline
  \text{Show:} & \text{(Arrival-time)} \\
  \text{Origin} & \text{(City “Newark”)}
  \\
  \text{Destination:} & \text{(City ”Atlanta”)}
  \\
  \text{Date:} & \text{(November 27th, 2003)}
  \\
  \hline
  \end{array}
  \]

- Method uses a decision-tree model to estimate probability of "carrying over" each slot in $H$ which is not in $F, S$. 
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, Lambertsville, 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**
  
  **System initiative:** Welcome to NJFun. Please say an activity name or say “list activities” for 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 $P(s'|s, a)$
- For a state-action pair $(s, a)$, the reward received is $R(s, a)$
  (e.g., $R(s, a) = 1$ 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

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

• $Q(s, a)$ 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

- 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
States in the Dialogue System

- $\text{greet}=0$ if user has to be greeted, $1$ otherwise
- $\text{attr}$ represents attribute being queried; $1/2/3=\text{activity/location/time}$, $4=\text{done with attributes}$
- $\text{conf}$ represents confidence in the attribute value.
  - $0,1,2=\text{low/middle/high confidence in the speech recognizer}$
  - $3=\text{recognition system has received “YES” as an answer to a confirmation}$
  - $4=\text{system has received “NO”}$
- $\text{val}=1$ if attribute value has been obtained, $0$ otherwise
- $\text{times}=\text{number of times system has asked about the attribute}$
- $\text{gram}=\text{type of grammar used to obtain the attribute value}$
- $\text{hist}=0$ if system has had problems in understanding the user earlier in the conversation; $1$ otherwise
States in the Dialogue System

<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>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>

- 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
  System initiative: I know about amusement parks, aquariums, cruises, . . . . Please say a name from the list
  User initiative: Please tell me the activity type. You can also tell me the location and time.

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

⇒ use a speech recognizer 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 recognizer 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.
- User initiative vs. system initiative
- 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 gram.</td>
</tr>
<tr>
<td>Ask2U</td>
<td>attribute=location, system initiative, unrestrictive gram.</td>
</tr>
</tbody>
</table>
Actions in the System
An Example

- Initial state is always

- Possible actions in this state:
  GreetU: Welcome to NJFun. How may I help you?
  GreetS: Welcome to NJFun. Please say an active 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
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>greet</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)
Planning-based Conversational Agents

• Idea: Model Gricean inference to design intelligent conversational systems

• Methods: Use planning and reasoning methods
  – Involves planning, plus various extensions to logic to create logic for Belief, Desire, Intention
Plan-based agent interpreting user utterance

C: I need to travel in May
A: And, what day in May did you want to travel?
C: OK, uh, I need to be there for a meeting that’s from the 12th
C: to the 15th

Inference chain:

- Systems knows that one precondition for having a meeting is being at the place where the meeting is
- One way of being at a place is flying there
- Booking a flight is a precondition for flying there

System abduces that user wants to fly on a date before the 12th
Plan-based agent producing user utterance

C: I need to travel in May
A: And, what day in May did you want to travel?
C: OK, uh, I need to be there for a meeting that’s from the 12th
C: to the 15th

Inference chain:

- The system must know enough information about the flight to book it
- Knowing the month (May) is insufficient information to specify a departure or return day

System asks the client about the needed dates
BDI Logic

- $B(S,P) = \text{“speaker S believes proposition P”}$
- $\text{KNOW}(S,P) = P \text{ and } B(S,P)$
- $\text{KNOWIF} (S,P) = \text{“S knows whether } P \text{”} = \text{KNOW} (S,P) \text{ or } \text{KNOW} (S,\text{not}P)$
- $W(S\text{!}P) \text{“S wants P to be true”}, \text{ where } P \text{ is a state or the execution of some action}$
- $W(S,\text{ACT}(H)) = \text{S wants H to do ACT}$
How to represent actions

- **Preconditions:**
  - Conditions that must already be true in order to successfully perform the action

- **Effects:**
  - Conditions that become true as a result of successfully performing the action

- **Body:**
  - A set of partially ordered goal states that must be achieved in performing the action
The action of booking a flight

- **BOOK-FLIGHT \((A,C,F)\)**
- **Constraints:** Agent\((A)\) & Flight\((F)\) & Client\((C)\)
- **Precondition:** Know \((A, \text{dep-time}(F))\) & Know\((A, \text{dep-time}(F))\) & Know\((A, \text{origin}(F))\) & Has-Seats\((F)\) & W\((C, \text{BOOK}, A, C, F)\) & . . .
- **Effect:** Flight-Booked\((A,C,F)\)
- **Body:** Make-Reservation \((A,F,C)\)
Speech acts

- INFORM(S,H,P)
- Constraints: Speaker(S) & Hearer(H) & Proposition(P)
- Precondition: Know(S,P) & $W(S,INFORM(S,H,P))$
- Effect: Know(H,P)
- Body: $B(H(W(S,Know(H,P))))$
Speech acts

- REQUEST-INFORM(A,C,I)
- Constraints: Agent(A) & Client (C)
- Precondition: Know(C,I)
- Effect: Know(A,I)
- Body: B(C(W(A,Know(A,I))))
How a plan-based agent works

While conversation is not finished

- If user has completed a turn
  - Then interpret user’s utterance

- If system has obligations
  - Then address obligations

- Else if system has turn
  - Then if system has intended conversation acts, then call generator to produce utterances
  - Else if high-level goals are unsatisfied, then address goals
  - Else release turn or attempt to end conversation

- Else if no one has turn or long pause
  - Then take turn
Summary

- Statistical NLU component
- Reinforcement learning for dialogue management
- Planning-based agent system