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

⇒

Show:	(Arrival-time)
Origin:	(City “Newark”)
Destination:	(City “Atlanta”)
Date:	(November 27th, 2003)

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 = a context-dependent semantic representation for W

H =

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

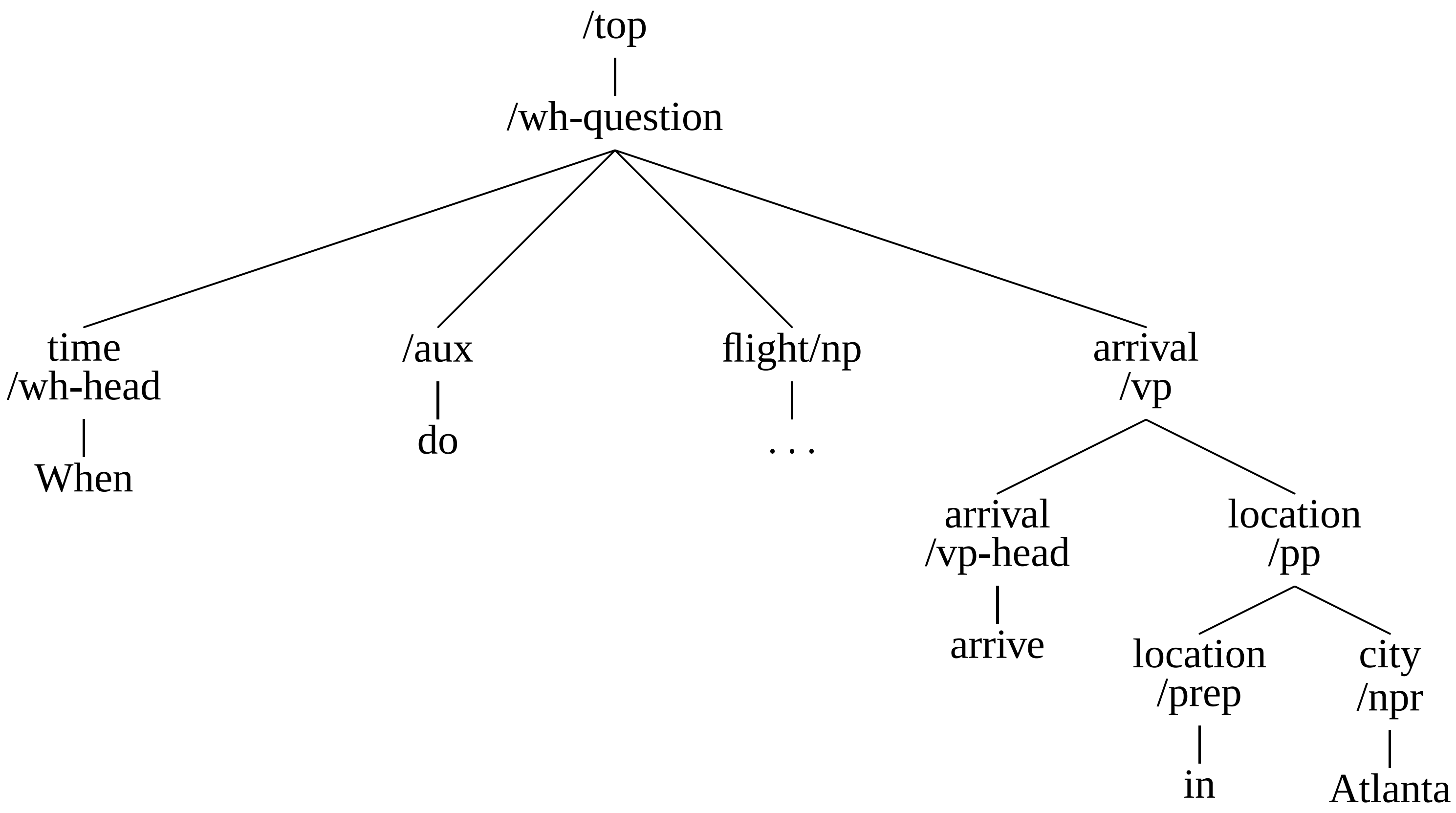
F, S =

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

M =

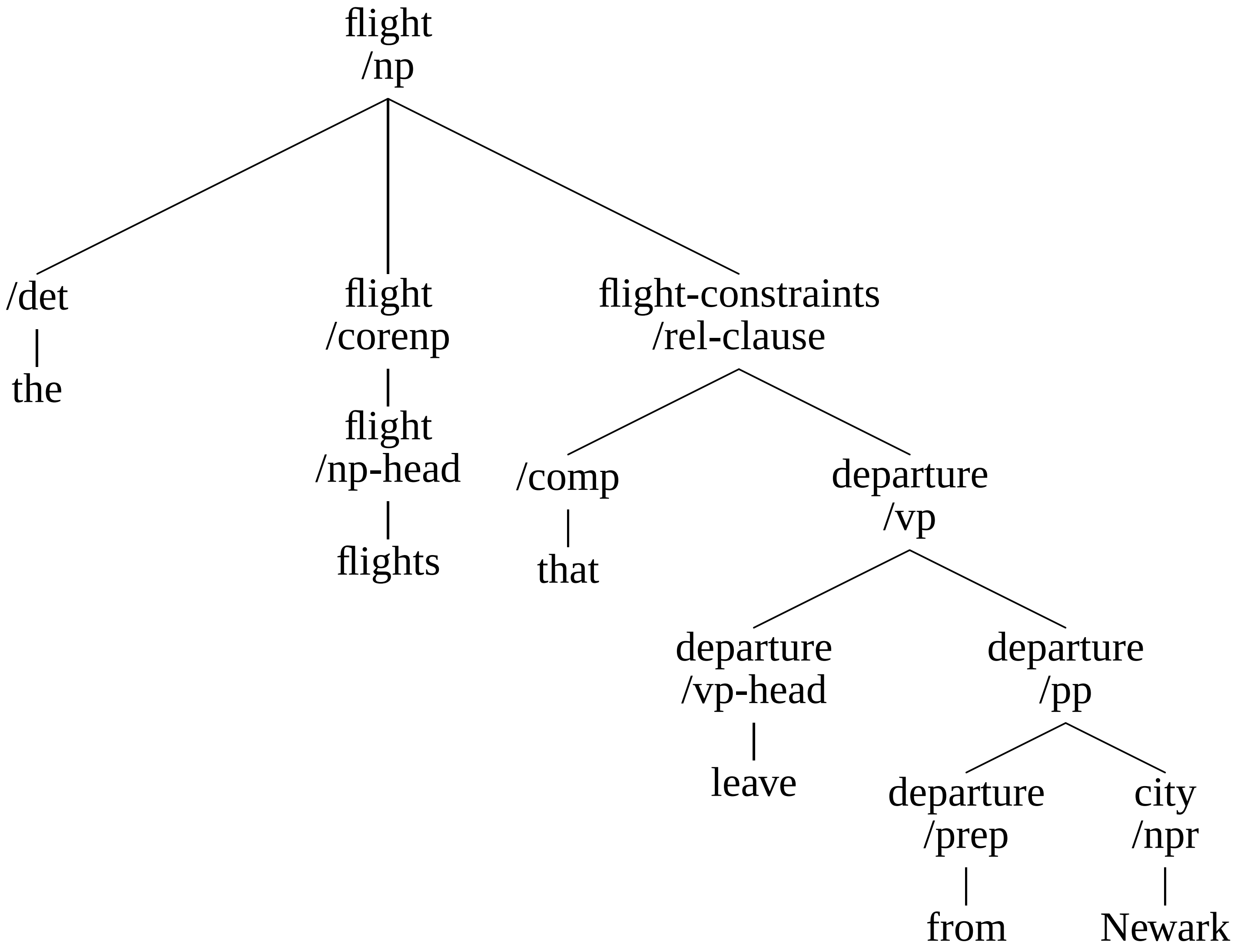
Show: (Arrival-time)
Origin: (City “Newark”)
Destination: (City “Atlanta”)
Date: (November 27th, 2003)

A Parse Tree


/top
/wh-question
time /wh-head
When
/aux
do
flight/np
. . .
arrival /vp
arrival /vp-head
arrive
location /pp
location /prep
in
city /npr
Atlanta

- Each non-terminal has a syntactic and semantic tag, e.g., `flight/np`

A Parse Tree (Continued)



flight
/np

/det
the

flight
/corenp

flight
/np-head

flights

flight-constraints
/rel-clause

/comp
that

departure
/vp

departure
/vp-head

leave

departure
/pp

departure
/prep

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

  $$\begin{aligned} \mathrm{argmax}_M P(M \mid W, H) &= \mathrm{argmax}_M P(M, W \mid H) \\ &= \mathrm{argmax}_M \left(\sum_{F,T,S} P(M, W, F, T, S \mid H) \right) \end{aligned}$$

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)

$$P(M, W, F, T, S \mid H) = \underbrace{P(F)P(T, W \mid F)P(S \mid T, W, F)}_{\text{Sentence processing model}} \quad \times \quad \underbrace{P(M \mid S, F, H)}_{\text{Contextual processing model}}$$

More About the Model

- The model:

$$P(M,W,F,T,S \mid H) = \underbrace{P(F)P(T,W \mid F)P(S \mid T,W,F)}_{\text{Sentence processing model}} \quad \times \quad \underbrace{P(M \mid S,F,H)}_{\text{Contextual processing model}}$$

- 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 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)

The Sentence Processing Model

$$P(T, W, F, S) = P(F) P(T, W \mid F) P(S \mid T, W, F)$$

- First step: choose the frame **F** with probability

$$P(F)$$

- e.g., choose the frame

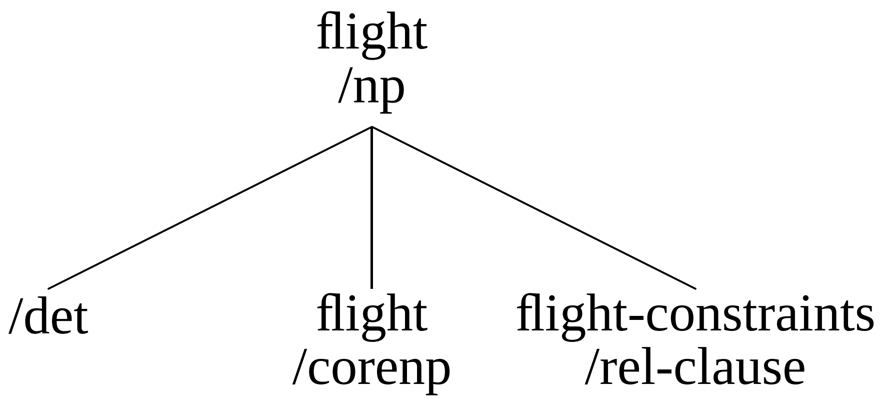
Show: (Arrival-time)
Origin:
Destination:

- 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**



$$
\begin{aligned}
& P(\text{/det} \quad \text{flight/corenp} \quad \text{flight-constraints/rel-clause} \mid {\color{red}\text{flight/np}}) \\
= \quad & P(\text{/det} \mid \text{NULL}, {\color{red}\text{flight/np}}) \\
& \times P(\text{flight/corenp} \mid \text{/det}, {\color{red}\text{flight/np}}) \\
& \times P(\text{flight-constraints/relclause} \mid \text{flight/corenp}, {\color{red}\text{flight/np}}) \\
& \times P(\text{STOP} \mid \text{flight-constraints/relclause}, {\color{red}\text{flight/np}})
\end{aligned}
$$

Smoothing

$$\begin{aligned} & P(\text{flight/corenp} \mid \text{/det, flight/np}) \\ = \quad & \lambda \times P_{ML}(\text{flight/corenp} \mid \text{/det, flight/np}) \\ & +(1-\lambda) \times P_{ML}(\text{flight} \mid \text{flight}) P_{ML}(\text{corenp} \mid \text{np}) \end{aligned}$$

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

$$P_{ML}(\text{corenp} \mid \text{np}) = \frac{Count(\text{corenp, np})}{Count(\text{np})}$$

- λ is a smoothing coefficient between 0 and 1
- Backed-off estimates generate semantic, syntactic parts of each label seperately

The Sentence Processing Model

$$P(T, W, F, S) = P(F)P(T, W \mid F)P(S \mid T, W, F)$$

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

Show: (Arrival-time)
Origin:
Destination:

⇒

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

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

$$P(\text{slot-fill-action} \mid \text{S, node})$$

The Sentence Processing Model

$$P(T, W, F, S) = P(F)P(T, W \mid F)P(S \mid T, W, F)$$

- 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 $P(T, W \mid F)$. 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

$$\underbrace{P(M \mid S, F, H)}_{\text{Contextual processing model}}$$

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”)

Only issue is whether values in **H**, but not in **(F, S)**, should be carried over to **M**.
⇒ **Two possible values for M**:

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

Show: (Arrival-time)
Origin: (City “Newark”)
Destination: (City “Atlanta”)
Date: (November 27th, 2003)

The Contexual Model

- Method uses a decision-tree model to estimate probability of "carrying over" each slot in **H** which is not in **F, 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

$$\text{argmax}_M \left(\max_i P(M \mid F_i, S_i, H) P(F_i, S_i, T_i, W) \right)$$

- Note that this is an approximation to finding

$$\text{argmax}_M P(M, W \mid H) = \text{argmax}_M \sum_{F,S,T} P(M, W, F, S, T \mid H)$$

- Results: training on 4000 sentences, achieved an error rate of 21.6%

Reinforcement Learning for Dialogue Management

- *Automatic Optimization of Dialogue Management.* [Litman, Kearns, Singh, and Walker 2000]

- 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, $\mathcal{S}$

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

- 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) $\Rightarrow$ we need to infer the utility of each action in each state from this *indirect* or *delayed* form of supervision

Policies

- A **policy** $\pi : \mathcal{S} \rightarrow \mathcal{A}$ is a function that maps states to actions
- Define

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

- $Q(s,a)$ is the expected reward when action a is taken in state s
- If $P(s' \mid s,a)$ is known, $Q(s,a)$ can be calculated, and optimal policy is

$$\pi(s) = \mathrm{argmax}_a Q(s,a)$$

Main point: if $P(s' \mid s,a)$ 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 $P(s' \mid s, a)$, 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

feature	greet	attr	conf	val	times	gram	hist
possible values	0,1	1,2,3,4	0,1,2,3,4	0,1	0,1,2	0,1	0,1

- **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.

Action	Description
GreetS	attribute=greeting, system initiative
GreetU	attribute=greeting, user initiative
ReAsk1S	attribute=activity, system initiative, restrictive grammar
Ask2U	attribute=location, user initiative, unrestrictive grammar

Action	Prompt
GreetS	Welcome to NJFun. Please say an activity name or say 'list activities' for a list of activities I know about.
GreetU	Welcome to NJFun. How may I help you?
ReAsk1S	I know about amusement parks, aquariums, cruises, historic sites, museums, parks, theaters, wineries, and zoos. Please say an activity name from this list.
ReAsk1M	Please tell me the activity type.You can also tell me the location and time.
Ask2S	Please say the name of the town or city that you are interested in.
Ask2U	Please give me more information.
ReAsk2S	Please tell me the name of the town or city that you are interested in.
ReAsk2M	Please tell me the location that you are interested in. You can also tell me the time.

Figure 2: Sample initiative strategy choices.

understanding, text-to-speech (TTS), database access, and dialogue management. NJFun uses a speech recognizer with stochastic language and understanding models trained from example user utterances, and a TTS system based on concatenative diphone synthesis. Its database is populated from the `nj.online` webpage to contain information about activities. NJFun indexes this database using three attributes: activity type, location, and time of day (which can assume values morning, afternoon, or evening).

Informally, the NJFun dialogue manager sequentially queries the user regarding the activity, loca-

The examples in Figure 2 show that NJFun can ask the user about the first 2 attributes[1] using three types of initiative, based on the combination of the wording of the system prompt (*open* versus *directive*), and the type of grammar NJFun uses during ASR (*restrictive* versus *non-restrictive*). If NJFun uses an open question with an unrestricted grammar, it is using *user initiative* (e.g., GreetU). If NJFun instead uses a directive prompt with a restricted grammar, the system is using *system initiative* (e.g., GreetS). If NJFun uses a directive question with a non-restrictive grammar, it is using *mixed initiative*, because it allows the user to take the initiative by

An Example

- Initial state is always

greet	attr	conf	val	times	gram	hist
0	1	0	0	0	0	0

- 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

greet	attr	conf	val	times	gram	hist
1	1	2	1	0	0	0

(user has been greeted, current atribute 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

greet	attr	conf	val	times	gram	hist
1	2	2	1	0	0	1

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.

State							Action	Turn	Reward
greet	attr	conf	val	times	gram	hist			
0	1	0	0	0	0	0	GreetU	S1	0
1	1	2	1	0	0	0	NoConf	-	0
1	2	2	1	0	0	1	ExpConf2	S2	0
1	3	2	1	0	0	1	ExpConf2	S3	0
1	4	0	0	0	0	0	Tell	S4	1

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)