

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]

Show:	(Arrival-time)
Origin	(City "NY")
Destination:	(City "Boston")
Date:	(November 27th, 2003)

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**; **E**, **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 "NY") or (City "NY")
Destination:	(City "Atlanta")
Date:	(November 27th, 2003)

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

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

Example

H=

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

F,S=

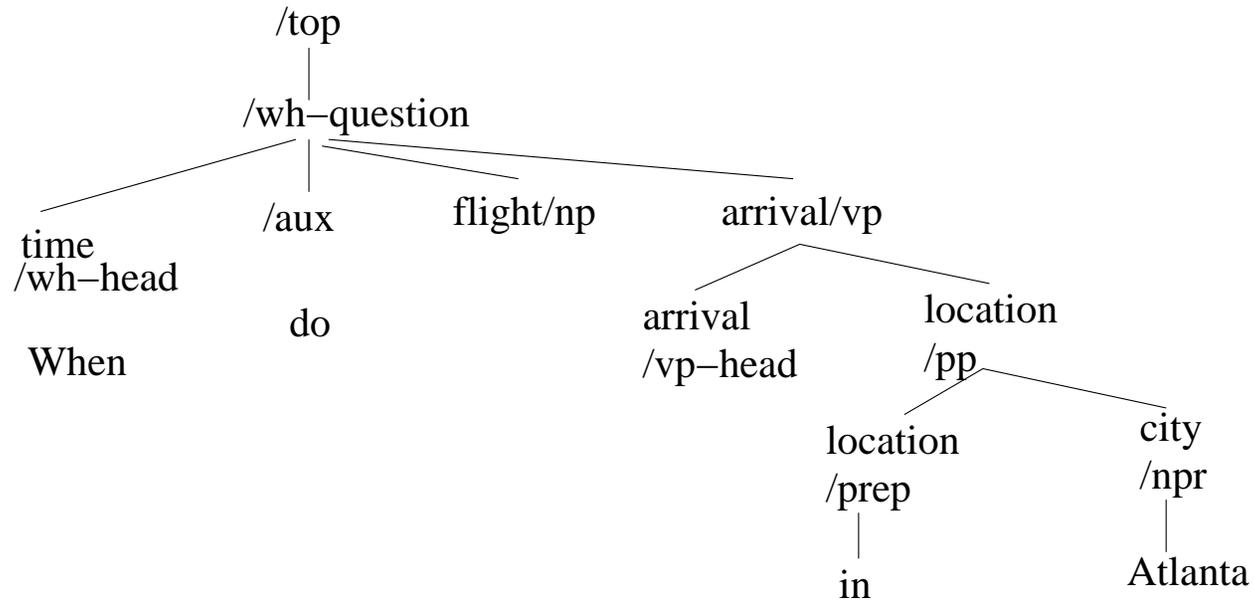
Show:	(Arrival-time)
Origin	(City "Newark")
Destination:	(City "Atlanta")

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., 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

$$\begin{aligned} \operatorname{argmax}_M P(M|W, H) &= \operatorname{argmax}_M P(M, W|H) \\ &= \operatorname{argmax}_M \sum_{F, T, S} P(M, W, F, T, S|H) \end{aligned}$$

Building a Probabilistic Model

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

- Apply Chain rule:

$$P(M, W, F, T, S|H) = P(F|H)P(T, W|F, H)P(S|T, W, F, H)P(M|S, T, W, F, H)$$

- Independence assumption:

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

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

H=

Show:	(flights)
Origin	(City "NY") or (City "NY")
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)

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

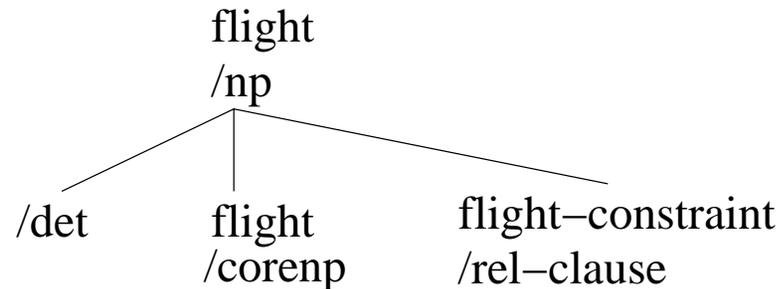
Destination:

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



$$\begin{aligned} & P(/det \text{ flight}/corenp \text{ flight-constraints}/rel-clause|flight/np) \\ &= P(/det|NULL, \text{ flight}/np) * P(\text{flight}/corenp|/det, \text{ flight}/np) \\ &* P(\text{flight-constraints}/rel-clause|flight/corenp, \text{ flight}/np) \\ &* P(STOP|\text{flight-constraints}/rel-clause, \text{ flight}/np) \end{aligned}$$

- Use maximum likelihood estimation

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

- 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

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

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

H=

Show:	(flights)
Origin	(City "NY") or (City "NY")
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 Contextual Model

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

M=	Show:	(Arrival-time)
	Origin	(City "Newark")
	Destination:	(City "Atlanta")
	Date:	(November 27th, 2003)

- 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

- *Automatic Optimization of Dialogue Management*
(Litman&Kearns&Singh&, 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, 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, \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 $P(s'|s, a)$
- 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, $(s_1, a_1), (s_2, a_2) \dots (s_n, a_n)$

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** $\pi : \mathcal{S} \rightarrow \mathcal{A}$ 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 $\pi(s) = \operatorname{argmax}_a Q(s, a)$

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
($(s_1, a_1), (s_2, a_2) \dots (s_n, a_n)$ 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

- **greet**=0 if user has to be 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/miiddle/high confidence in the speech recognizer;
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

States in the Dialogue System

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

- 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

Action	Description
GreetS	attribute=greeting, system initiative
GreetU	attribute=greeting, user initiative
REAsk1S	attribute=activity, system initiative, restrictive gram.
Ask2U	attribute=location, system initiative, unrestrictive gram.

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

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

(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

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

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

greet attr conf val times gram hist							Action	Turn	Reward
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)

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

- $B(S,P)$ = “speaker S believes proposition P”
- $KNOW(S,P)$ = P and $B(S,P)$
- $KNOWIF(S,P)$ = “S knows whether P” = $KNOW(S,P)$ or $KNOW(S,\text{not}P)$
- $W(SiP)$ “S wants P to be true”, where P is a state or the execution of some action
- $W(S,ACT(H))$ = 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, dep-time(F))& Know(A, dep-time(F)) &Know(A,origin(F))&Has-Seats(F)& W(C, 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