Dialogue Management

Natalie Parde
UIC CS 421

Dialogue Management

- Core component of task-based dialogue systems
  - Decides what step to take next to bring the conversation closer to its goal
- Can range from simple (minimal history and/or state tracking) to complex (advanced state tracking and dialogue policy modules)
- Simplest dialogue management architecture:
  - **Finite state dialogue manager**

Finite State Dialogue Manager

States (nodes)

- Questions that the dialogue manager asks the user

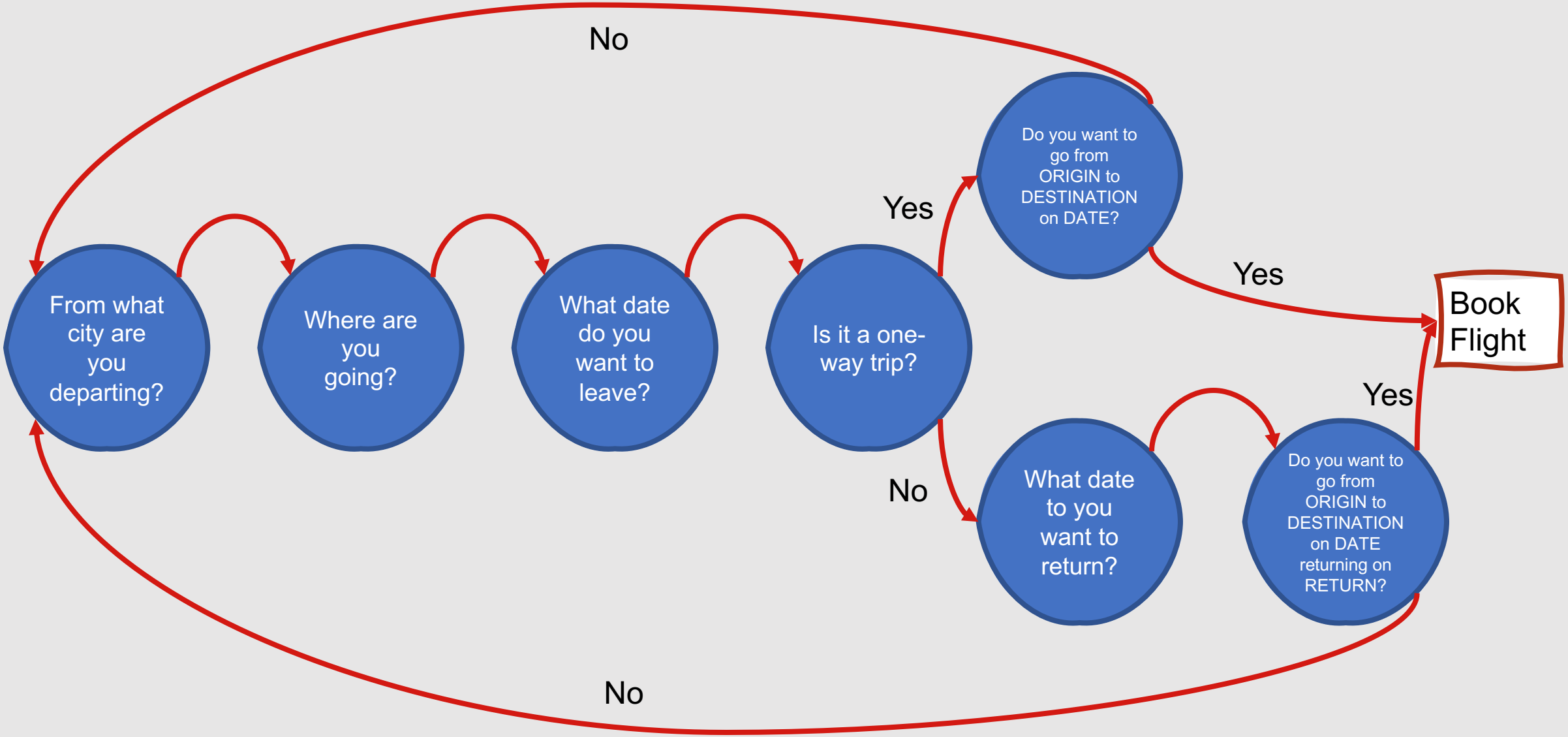
Transitions (arcs)

- Actions to take depending on how the user responds

System has full conversational initiative!

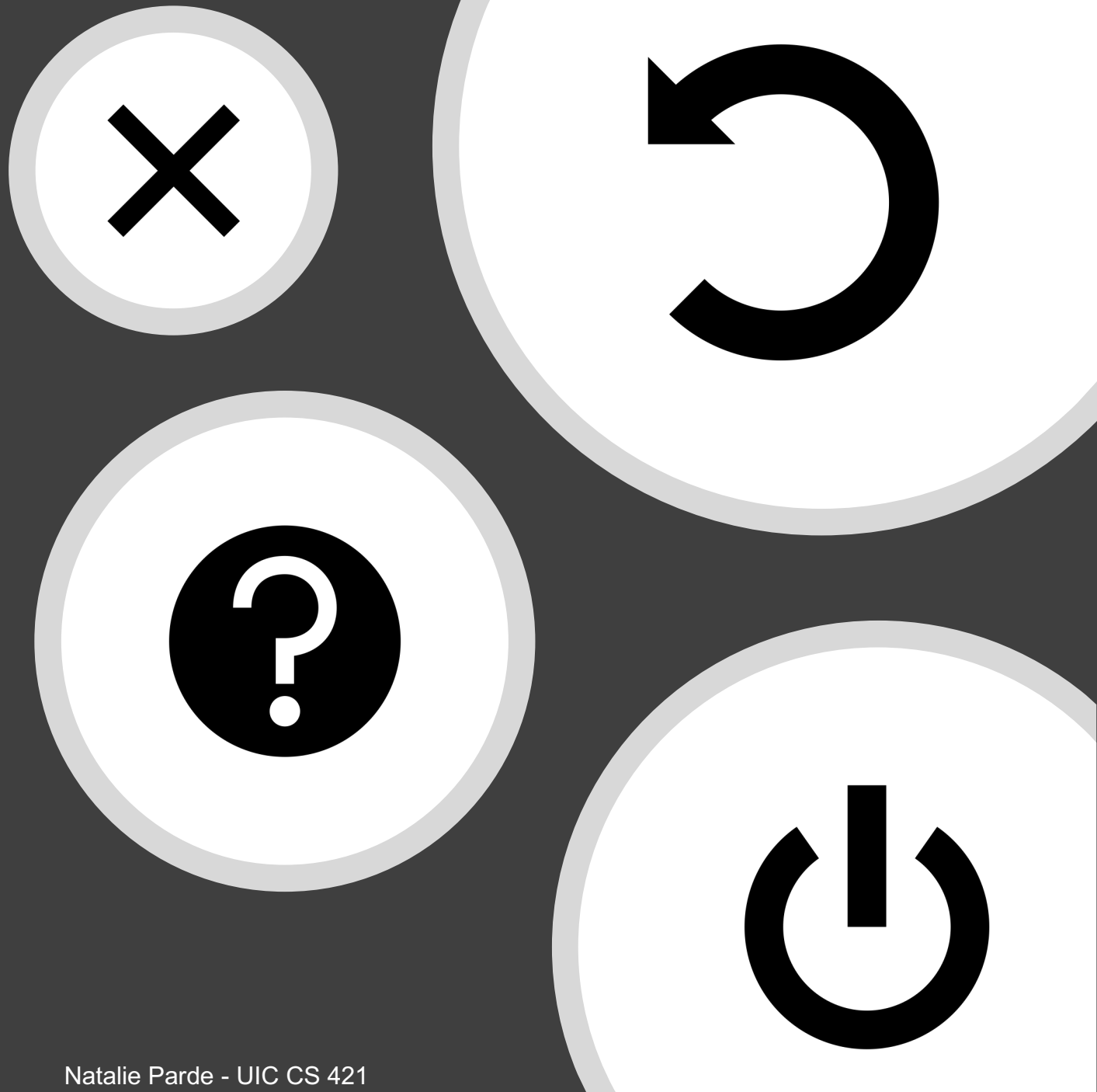
- Asks a series of questions
- Ignores or misinterprets inputs that are not direct answers to questions

Finite State Dialogue Manager



Finite State Dialogue Manager

- Many finite state systems also allow **universal commands**
  - Commands that can be stated anywhere in the dialogue and still be recognized
    - Help
    - Start over
    - Correction
    - Quit



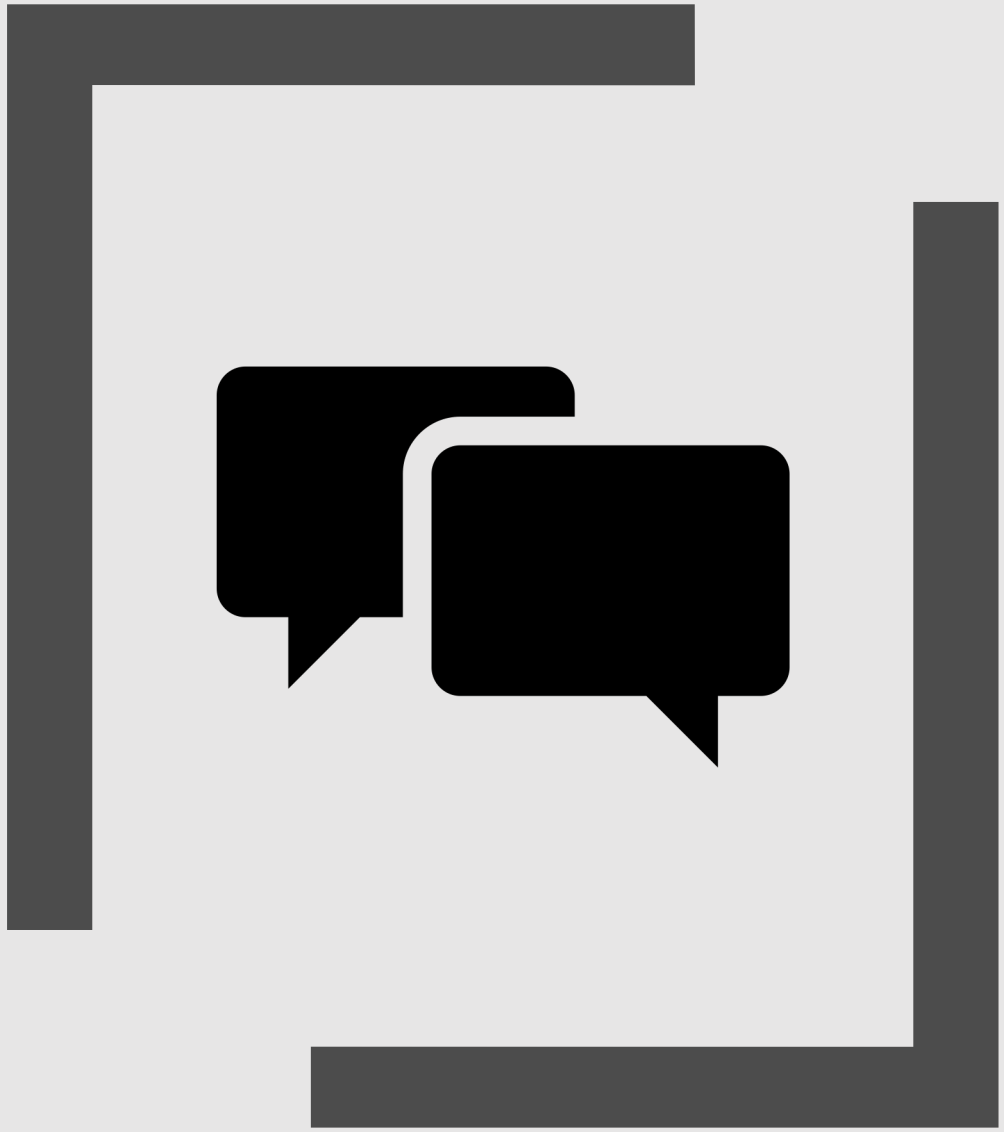
Advantages and Disadvantages of Finite State Dialogue Managers

Advantages:

- Easy to implement
- Sufficient for simple tasks

Disadvantages:

- Can be awkward and annoying
- Cannot easily handle complex sentences



Dialogue Management

- More common in modern dialogue state architectures:
  - **Dialogue state tracker**
  - **Dialogue policy**

What does a dialogue state tracker do?

Determine both:

- The current state of the frame
  - What slots have been filled, and how?
- The user's most recent dialogue act

Current state of the frame: More than just the slot fillers expressed in the current sentence!

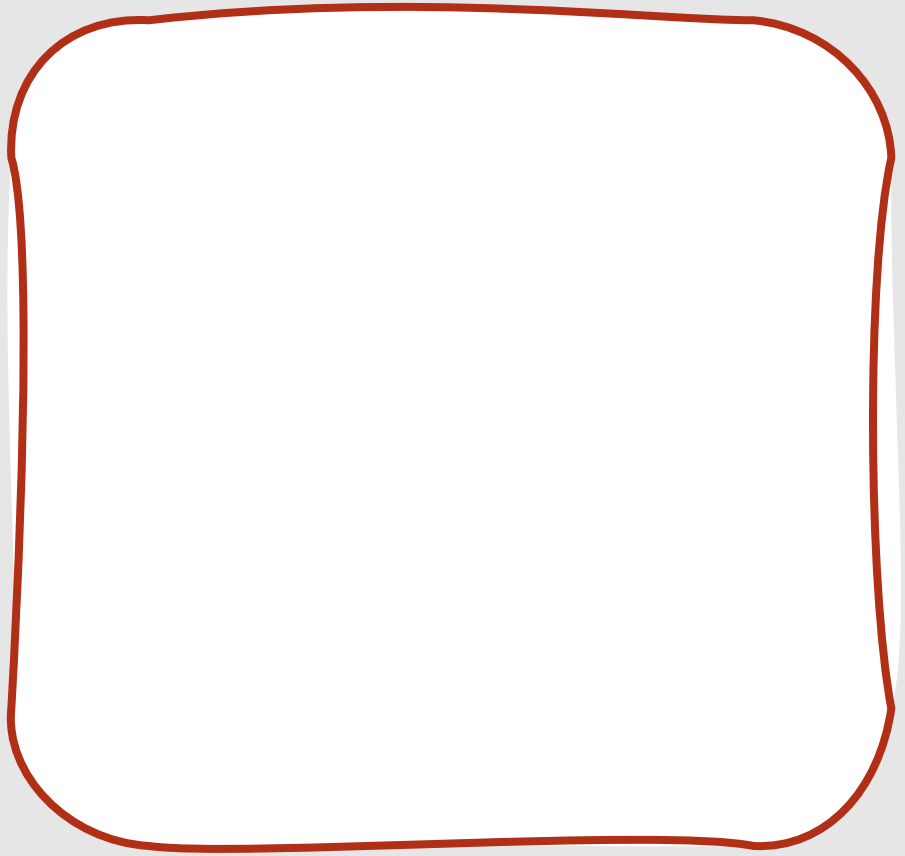
- Entire state of the frame up to and including this point

Example: Dialogue State Tracker

I'm looking for an upscale restaurant.

Tag
Hello(a=x, b=y, …)
Inform(a=x, b=y, …)
Request(a, b=x, …)
Reqalts(a=x, …)
Confirm(a=x, b=y, …)
Confreq(a=x, …, d)
Select(a=x, a=y)
Affirm(a=x, b=y, …)
Negate(a=x)
Deny(a=x)
Bye()

Dialogue State Tracker



Example: Dialogue State Tracker

I'm looking for an upscale restaurant.

Tag
Hello(a=x, b=y, …)
Inform(a=x, b=y, …)
Request(a, b=x, …)
Reqalts(a=x, …)
Confirm(a=x, b=y, …)
Confreq(a=x, …, d)
Select(a=x, a=y)
Affirm(a=x, b=y, …)
Negate(a=x)
Deny(a=x)
Bye()

Dialogue State Tracker

inform(price=expensive)

Example: Dialogue State Tracker

I'm looking for an upscale restaurant.

Sure. What cuisine?

Turkish food would be great.

Tag
Hello(a=x, b=y, …)
Inform(a=x, b=y, …)
Request(a, b=x, …)
Reqalts(a=x, …)
Confirm(a=x, b=y, …)
Confreq(a=x, …, d)
Select(a=x, a=y)
Affirm(a=x, b=y, …)
Negate(a=x)
Deny(a=x)
Bye()

Dialogue State Tracker

inform(price=expensive)

Example: Dialogue State Tracker

I'm looking for an upscale restaurant.

Sure. What cuisine?

Turkish food would be great.

Tag
Hello(a=x, b=y, …)
Inform(a=x, b=y, …)
Request(a, b=x, …)
Reqalts(a=x, …)
Confirm(a=x, b=y, …)
Confreq(a=x, …, d)
Select(a=x, a=y)
Affirm(a=x, b=y, …)
Negate(a=x)
Deny(a=x)
Bye()

Dialogue State Tracker

inform(price=expensive,
cuisine=Turkish)

Example: Dialogue State Tracker

I'm looking for an upscale restaurant.

Sure. What cuisine?

Turkish food would be great.

Okay. Where should the restaurant be?

Near the Chicago Theatre.

Tag
Hello(a=x, b=y, …)
Inform(a=x, b=y, …)
Request(a, b=x, …)
Reqalts(a=x, …)
Confirm(a=x, b=y, …)
Confreq(a=x, …, d)
Select(a=x, a=y)
Affirm(a=x, b=y, …)
Negate(a=x)
Deny(a=x)
Bye()

Dialogue State Tracker

inform(price=expensive,
cuisine=Turkish,
area=ChicagoTheatre)

Example: Dialogue State Tracker

I'm looking for an upscale restaurant.

Sure. What cuisine?

Turkish food would be great.

Okay. Where should the restaurant be?

Near the Chicago Theatre.

So an upscale Turkish restaurant near the Chicago Theatre?

Yes, please.

Tag
Hello(a=x, b=y, …)
Inform(a=x, b=y, …)
Request(a, b=x, …)
Reqalts(a=x, …)
Confirm(a=x, b=y, …)
Confreq(a=x, …, d)
Select(a=x, a=y)
Affirm(a=x, b=y, …)
Negate(a=x)
Deny(a=x)
Bye()

Dialogue State Tracker

inform(price=expensive,
cuisine=Turkish,
area=ChicagoTheatre);

affirm(price=expensive,
cuisine=Turkish,
area=ChicagoTheatre)

How can we detect corrections?

- Users generally correct errors (either theirs or the system's) by **repeating** or **reformulating** their utterance
- Harder to do than detecting regular utterances!
  - Speakers often **hyperarticulate** corrections
- Common characteristics of corrections:
  - Exact or close-to-exact repetitions
  - Paraphrases
  - Contain "no" or swear words
  - Low ASR confidence

Dialogue Policy

- Goal: Determine **what action the system should take next**
  - What dialogue act should be generated?
- More formally, at turn *i* in the conversation, we want to predict which action A_i to take, **based on the entire dialogue state** (the sequence of dialogue acts from the system *A* and the user *U*):
  - $A'_i = \underset{A_i \in A}{\operatorname{argmax}} P(A_i | A_1, U_1, \ldots, A_{i-1}, U_{i-1})$
- To simplify this, we can maintain the dialogue state as the set of slot-fillers the user has expressed (thereby allowing us to condition on **the current state of *Frame*$_i$ and the last turn**):
  - $A'_i = \underset{A_i \in A}{\operatorname{argmax}} P(A_i | Frame_{i-1}, A_{i-1}, U_{i-1})$

How can we estimate these probabilities?

- **Neural classifier** trained on **vector representations of the slot fillers and utterances**
- More sophisticated models may also use **reinforcement learning**
  - Reinforcement learning system gets a reward at the end of a dialogue
  - Uses that reward to train a policy to take an optimal sequence of actions
    - Large positive reward if the dialogue system terminates with the correct slot representation for a training instance
    - Large negative reward if all slots are wrong

How can dialogue managers handle mistakes?

- First, check to make sure the user's input has been interpreted correctly:
  - **Confirm understandings** with the user
  - **Reject utterances** that the system is likely to have **misunderstood**
- These checks can be performed **explicitly** or **implicitly**

Explicit Confirmation

- System asks the user a direct question to confirm its understanding

S: From which city do you want to leave?

U: Chicago.

S: You want to leave from Chicago?

U: Yes.

U: I'd like to fly from Chicago to Dallas on November twenty-seventh.

S: Okay, I have you going from Chicago to Dallas on November twenty-seventh. Is that correct?

U: Yes.

Implicit Confirmation

- System demonstrates its understanding as a **grounding** strategy
- Usually done by repeating back its understanding as part of the next question

U: I want to travel to Chicago.

S: When do you want to travel to Chicago?

U: Hi, I'd like to fly to Chicago tomorrow afternoon.

S: Traveling to Chicago on November fifteenth in the afternoon. What is your full name?

When to use explicit vs. implicit confirmation?

Explicit Confirmation

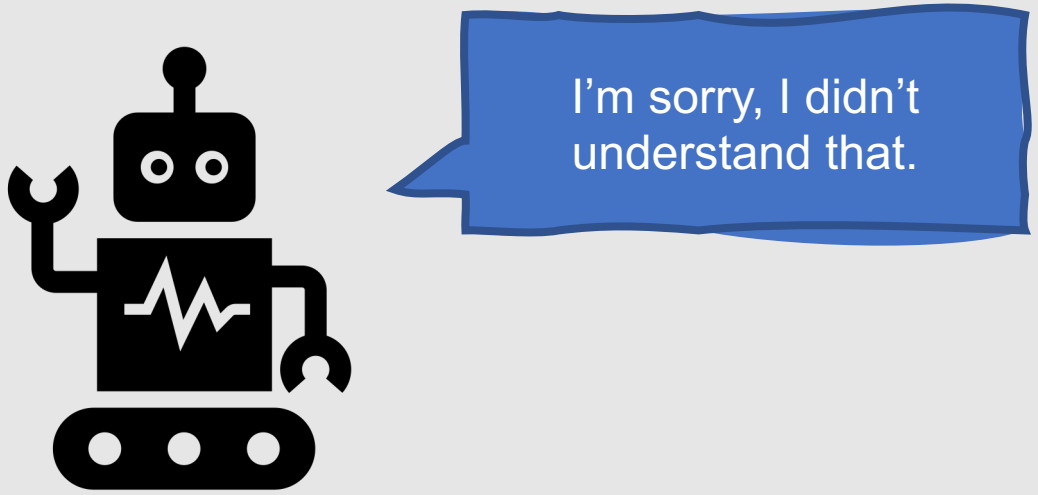
- Easier for users to correct the system's misrecognitions
  - User can just say "no" when the system tries to confirm
- But, can be awkward and unnatural
- Also extends the length of the conversation

Implicit Confirmation

- Sounds more conversationally natural
- More efficient
- But, more difficult for users to correct the system's misrecognitions

What if the dialogue manager has no idea how to handle the user's input?

- Systems can outright **reject** the user's input
- Often, when utterances are rejected, systems will follow a strategy of **progressive prompting** or **escalating detail**



Progressive Prompting

- System gradually gives the user more guidance about how to formulate an utterance that will be accepted

S: When would you like to leave?

U: Well, I need to be in Dallas for my post-vaccination party.

S: Sorry, I didn't get that. Please say the month and day you'd like to leave.

U: I would like to leave on January 14th.

Other Strategies for Error Handling

- **Rapid Reprompting:** For the first rejection, the system just says “I’m sorry?” or “What was that?”
  - From the second rejection onward, **progressive prompting** can be applied
  - **Users tend to prefer rapid reprompting** as a first-level error prompt (Cohen et al., 2004)
- Explicitly confirm **low-confidence ASR outputs**
- **Design thresholds based on the cost** of making an error
  - Low confidence → Reject
  - Confidence just above minimum threshold → Confirm explicitly
  - Confidence comfortably above threshold → Confirm implicitly
  - Very high confidence → Don’t confirm at all