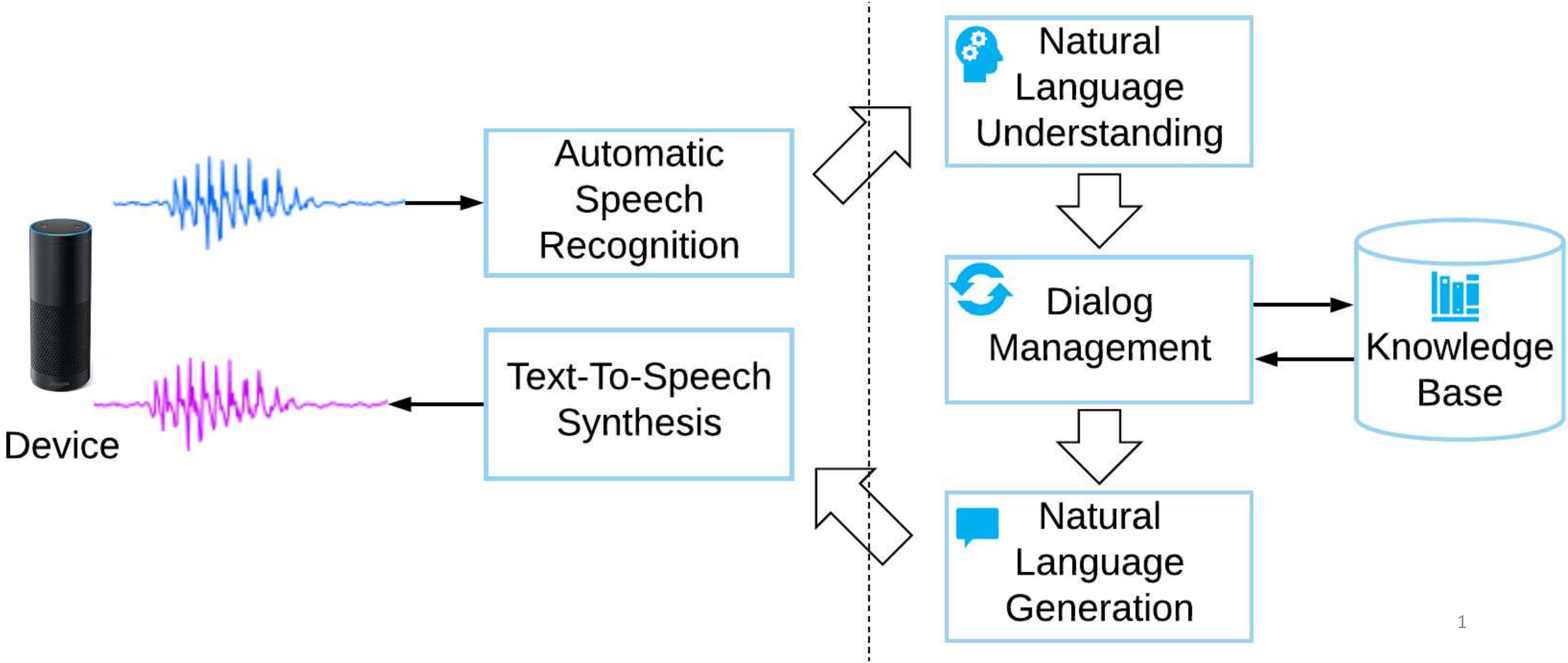
Dialog Management

EE596/LING580 -- Conversational Artificial Intelligence

Hao Cheng

University of Washington

Dialog Management in Dialog Systems



What is Dialog Management?

- Controls the interaction with the user
  - Takes input from ASR/NLU components
  - Determines what system does next
  - Passes output to NLG/TTS modules
- Communicates with external knowledge sources
- Often viewed in terms of two subcomponents
  - Dialog context modeling tracks contextual information used by the dialog manager to interpret user's input and inform the decisions of the dialog control component
  - Dialog control deals with the flow of control in the dialog

Dialog Context Modeling

Dialog Context Modeling

Conversations are highly contextualized

- Anaphoric reference
  - Bot: "*Do you want to talk about technology or science?*"
  - User: "*The first topic sounds good*"
- Ellipsis
  - Bot: "*When do you want to leave from Seattle?*"
  - User: "*[I want to leave from Seattle] Tomorrow at 2pm*"
- Non-linguistic context
  - Location: "*turn on the light*" (living room vs. bedroom)
  - User preference: "*play my favorite music*"

Knowledge Sources for Dialog Context Modeling

- Dialog history
  - a record of the dialog so far, e.g., questions that have been asked, entities that have been mentioned, topics that have been suggested
- Task record
  - a representation of the information to be gathered in the dialog, often referred to as a form, frame, template, or status graph
  - used to determine what information has been acquired by the system and what information still has to be acquired

Knowledge Sources for Dialog Context Modeling

- Domain model
  - specific information about the domain in question, e.g., flight information
  - often encoded in a database from which relevant information is retrieved by the dialog system

Knowledge Sources for Dialog Context Modeling

- Model of conversational competence
  - generic knowledge of principles of conversational turn-taking and discourse obligations, e.g., an appropriate response to a request for information is to supply the information or provide a reason for not supplying it
  - often encoded in a data structure known as the “agenda”
- User preference model
  - stable information about the user, e.g., age, gender, preferences
  - dynamic information that changes over the course of the dialog, e.g., goals, beliefs, intentions

Dialog Control

Dialog Control

- Dialog control involves deciding what to do next once the user's input has been received and interpreted.
- Examples of decisions include:
  - Prompting the user for more input
  - Clarifying or grounding the user's previous input
  - Outputting some information to the user
- Many design considerations:
  - Dialog initiative: determines who has control of conversation
  - Conversational grounding: acknowledges the user & explicitly/implicitly explains the system's action

Dialog Initiative

System-Initiative

- System completely controls the dialog
- System "knows" what user can say
- System ignores/misinterprets anything the user says that is not expected by the system
- Common in simple and well-defined tasks

User-Initiative

- User completely controls the dialog
- User knows what system can do
- System doesn't extend the dialog
- Common in short-term conversations, e.g., question answering and voice-based web search

More natural but brings challenges for dialog control

Mixed-Initiative

- Initiative shifts back and forth between the system and the user
- Involves both system-initiative and user-initiative

Conversational Grounding

- Presumed a joint & collaborative communication
  - speaker & hearer mutually believe the same thing
  - Speaker tries to establish and add to common ground and mutual belief
  - Hearer must ground speaker's utterances to indicate heard and understood
- Principle of Closure (Clark 1996) (Norman 1988)
  - agents performing an action require evidence that they have succeeded in performing it
  - non-speech example: push elevator button -> light turns on

A Human-Human Conversation

C_1:	. . . I need to travel in May.
A_1:	And, what day in May did you want to travel?
C_2:	OK uh I need to be there for a meeting that's from the 12th to the 15th.
A_2:	And you're flying into what city?
C_3:	Seattle.
A_3:	And what time would you like to leave Pittsburgh?
C_4:	Uh hmm I don't think there's many options for non-stop.
A_4:	Right. There's three non-stops today.
C_5:	What are they?
A_5:	The first one departs PGH at 10:00am arrives Seattle at 12:05 their time. The second flight departs PGH at 5:55pm, arrives Seattle at 8pm. And the last flight departs PGH at 8:15pm arrives Seattle at 10:28pm.
C_6:	OK I'll take the 5ish flight on the night before on the 11th.
A_6:	On the 11th? OK. Departing at 5:55pm arrives Seattle at 8pm, U.S. Air flight 115.
C_7:	OK.

Dialog Control Methods

- Finite-state-base
- Frame-based
- Statistical
- Classical AI Planning

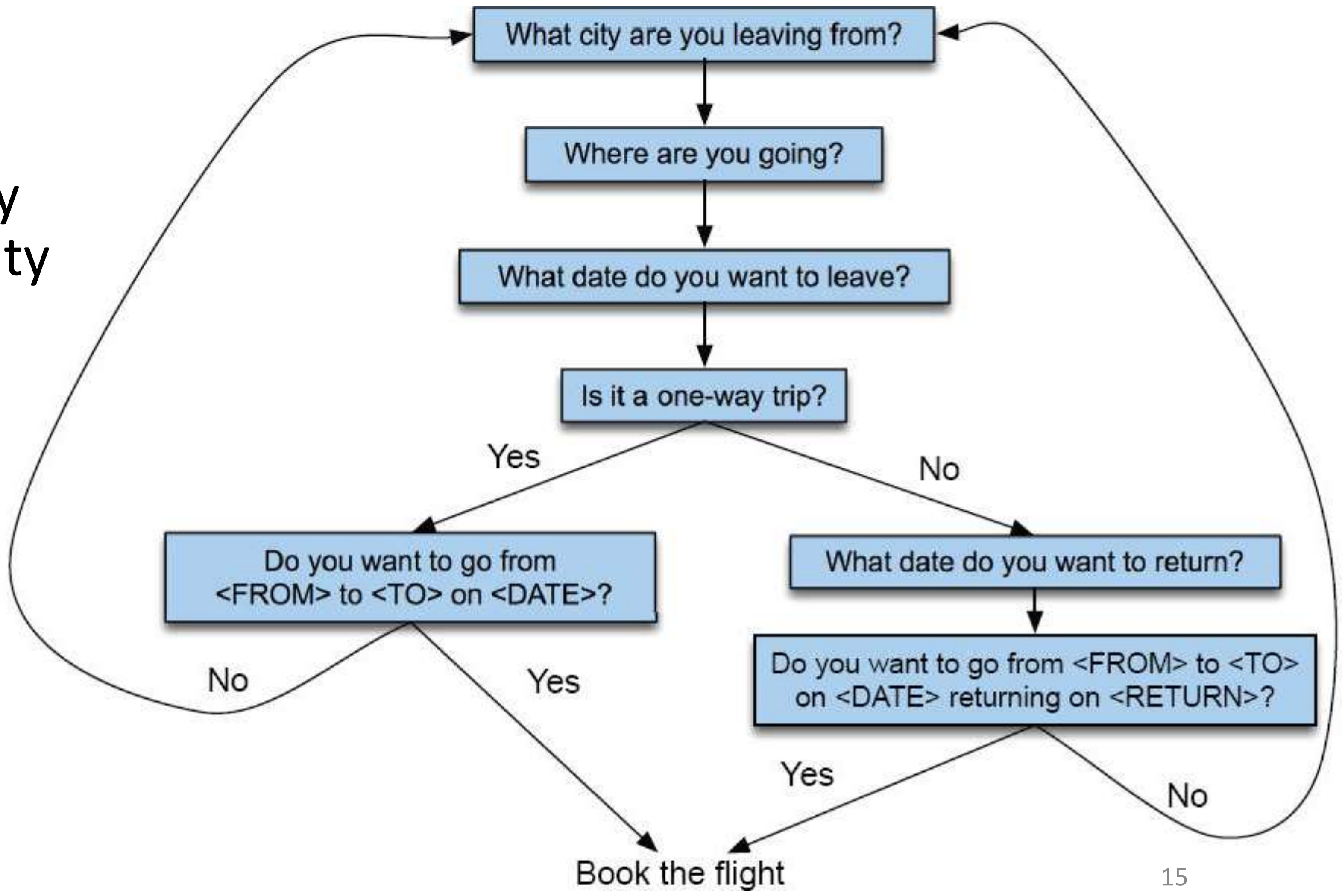
Today's lecture

Finite-State-Based Dialog Control

- Actions that can be taken at each point (or state) of the dialog are depicted in a graph.
- The states of the dialog graph can be traversed using a finite state automaton.

Example: A Trivial Airline Travel System

- Nodes represent the system's questions
  - Ask for a departure city
  - Ask for a destination city
  - Ask for a time
  - Ask whether the trip is round-trip or not
- Transitions between nodes represent answers to the questions



Example: A Trivial Airline Travel System

Advantages

- Straightforward to encode
- Clear mapping of interaction to model
- Well-suited to simple information access

Disadvantages

- Limited flexibility of interaction
- Constrained input – single item
- Only supports system initiative
- Restrictive dialog structure & order

We can add limited user-imitative capability by allowing some common commands at every state (called “universals”), e.g., Help, Repeat, Start Over, Weather, etc.

Finite-State-Based Dialog Control

- Each node can also be viewed as a state in which a collection of system actions are performed.
- Transitions can rely on complex language analysis on the user utterance and long-term conversation context.
- A possible implementation: using a collection of if-else conditions at every state

Frame-Based Dialog Control

- A frame represents the information that the system has to elicit in the course of the dialog.
- Frames consist of slots that are filled with the values elicited from the user.

FLIGHT FRAME

ORIGIN:

    CITY: Boston
    DATE: Tuesday
    TIME: morning

DEST:

    CITY: San Francisco

AIRLINE:

    …

Frame-Based Dialog Control

- Use the structure of the **frame** to guide dialogue

Slot	**Question**
ORIGIN	What city are you leaving from?
DEST	Where are you going?
DEPT DATE	What day would you like to leave?
DEPT TIME	What time would you like to leave?
AIRLINE	What is your preferred airline?

- User can answer multiple questions at once
- If user answers multiple questions at once, system fills all slots and does not ask these questions again
  - No strict constraints on order of questions

Frame-Based Dialog Control

- Require an elaborate algorithm to determine what the system's next question should be based on the information in the current frame.
- A possible implementation:

condition: unknown(origin) & unknown(destination)
question: "Which route do you want to travel?"

condition: unknown(origin)
question: "Where do you want to travel from?"

condition: unknown(destination)
question: "Where do you want to travel to?"

- Each question is listed along with its preconditions.
- The dialog control algorithm loops through all questions and selects the first question for which the condition were true.

Frame-Based Dialog Control

- Frame-based dialog control can still be viewed as a finite-state machine with a large set of dialog states
  - 5 questions, each 10 possible answers: 10,000 nodes
- Advantages of frame-based dialog control
  - Relatively flexible input & orders
  - Supports both system initiative and user initiative
    - i.e., user can provide more information than asked
  - Well-suited to complex information access

manually crafted finite-state-based dialog control is challenging for a huge state space

Issues of Manually-Crafted Dialog Control

- Dialog control in traditionally finite-state-based and frame-based dialog control are manually scripted.
  - based on experience and best practice guidelines
- Designers need to experiment with various choices
  - prompt design
  - confirmation strategy design
  - language models for ASR
  - …
- Difficult to design all the rules that would be required to cover all potential interactions of a dialog system

Statistical Dialog Control

- Statistical dialog control (data-driven)
  - a set of states S the system can be in
  - a set of actions A the system can take
  - a success metric that tells us the system performance
  - a policy π for what action to take in any particular state
- Approaches
  - supervised learning
  - reinforcement learning (RL)

Pre-defined

Pre-defined or learned from data

Learned from data

Labels for optimal (immediate) decisions

Maximize the "return", i.e., sum of rewards for the immediate gain associated with an action

Break (15min)

Dialog Policy Optimization using Reinforcement Learning

Basic Concepts in RL

- Reinforcement Learning is the framework for learning to make decisions through experiencing
- Model
  - Mathematical models of dynamics and reward
- Policy
  - Function mapping agent's states to actions
- Value
  - Future rewards from being in a state and/or action when following a particular policy

Aspects of RL

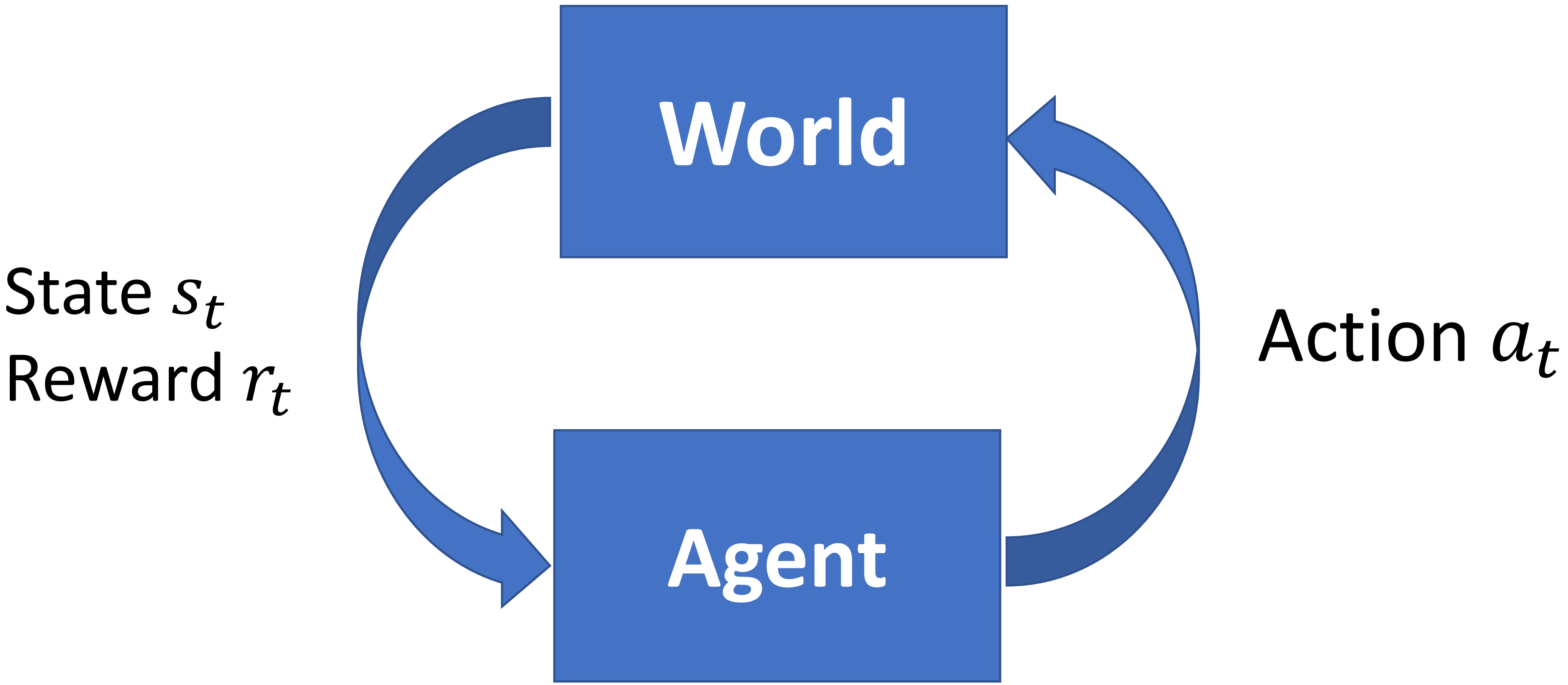
- Optimization:
  - Find an optimal to make decision, or yield a good outcome
- Delayed consequence:
  - Decisions made earlier have consequence on the future
- Exploration:
  - Learning the word by making decisions, i.e. decisions made previously determines what the agent learns
- Generalization:
  - Mapping from previous experience to action

Types of RL Agents

- Model-based
  - Known: Model
  - Learned: Policy and/or value function (can use model to plan: compute a policy and/or value function)
- Valued-based
  - Known: Value function
  - Learned: Policy (can derive a policy from value function)
- Policy-based
  - Known: Policy
  - No value function
- Actor-Critic
  - Known: Policy
  - Known: Value function

Full Observability: Markov Decision Process (MDP)

- Agent makes a decision (action) and observes output from the world.



Transition Probabilities

- The Markov assumption, or the state is Markov iff

$$P(s_{t+1} \mid s_t, s_{t-1}, \ldots, s_o, a_t, a_{t-1}, \ldots, a_o) = P_T(s_{t+1} \mid s_t, a_t)$$

- Information state: sufficient statistic of history
- Future is independent of past given present

Policy

- Policy specifies what action to take in each state
  - Can be deterministic or stochastic
- Policy can be modelled as a conditional distribution
  - Given a state, specifies a distribution over actions
- Policy:
  - $\pi(a|s) = P(a_t = a | s_t = s)$

Value

- Policy Evaluation:
  - $V^{\pi}(s) = r\big(s, \pi(s)\big) + \gamma \sum_{s'} P(s' | s, \pi(s)) V^{\pi}(s')$
- To compute the optimal policy:
  - $\pi^{*}(s) = argmax_{\pi} V^{\pi}(s)$

State-Action Q

- State-action value of a policy:
  - $Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s\,,a) V^{\pi}(s')$
- To compute the optimal policy:
  - $\pi^{*}(s) = argmax_{a}\, Q^{\pi}(s,a)$
- Usually done using iterative improvement

Dialog Policy Optimization using RL

- The developer specifies
  - a real-valued reward function
  - an optimization algorithm for learning to choose actions that maximize the reward function
- Formulize the dialog as a Markov Decision Process (MDP)
  - S: a set of system states
  - A: a set of actions A the system can take
  - T: a set of transition probabilities $P_T(S_t|S_{t-1}, a_{t-1})$
  - R: an immediate reward that is associated with taking a particular action in a given state

Immediate Reward

- Captures the immediate consequences of executing an action in a state
- Example rewards:
  - task success
  - number of corrections
  - number of accesses to a database
  - speech recognition errors
  - user satisfaction measures
  - …

Usually manually designed, but can also be learned from data

Cumulative Reward

- Captures the reward ("return") for a state sequence
- One common approach: discounted rewards
  - Cumulative reward Q of a sequence is discounted sum of utilities of individual states

$$Q([s_0, a_0, s_1, a_1, s_2, a_2 \cdots]) = R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \cdots,$$

  - Discount factor γ between 0 and 1
  - Makes the system care more about current than future rewards
    - the more future a reward, the more discounted its value

Expected Cumulative Reward

- Expected cumulative reward $Q(s, a)$ for taking a particular action from a particular state can be computed by <u>Bellman equation</u>:

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

immediate reward for current state

expected discounted utility of all possible next states s'

- weighted by probability of moving to that state s'
- assuming once there we take optimal action a'

Solving the Bellman Equation $Q(s, a)$

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

- $P(s'|s, a)$: learned from data
- Optima

Initialize $V(s)$ to arbitrary values
Repeat
  For all $s \in S$
    For all $a \in \mathcal{A}$
      $Q(s,a) \leftarrow E[r|s,a] + \gamma \sum_{s' \in S} P(s'|s,a)V(s')$
    $V(s) \leftarrow \max_a Q(s,a)$
Until $V(s)$ converge

How to learn $P(s'|s, a)$

- Have conversations with real (test) users
  - carefully hand-tune small number of states and policies
  - can build a dialogue system which explores state space by generating a few hundred conversations with real humans
  - expensive
- Have conversations with simulated users
  - can have millions of conversations with simulated users
  - but need to build a simulator first

From MDPs to POMDPs

- MDP assumption
  - the dialog states are fully observable
  - issues: our hypothesis about the dialog state may be incorrect given the uncertainties in ASR and NLU as well as the inherent ambiguity in dialog interactions
- Partially Observable Markov Decision Process (POMDP) assumption
  - the dialog states are partially observable
  - we maintain multiple hypotheses about the current dialog state

From MDPs to POMDPs

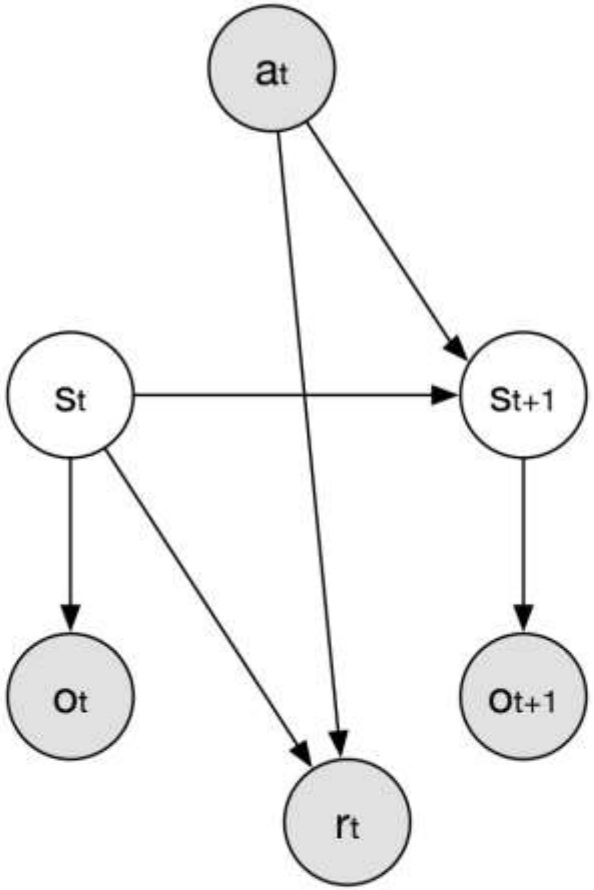
- MDP

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

- POMDP

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

s_t dialogue states
o_t noisy observations
a_t system actions
r_t rewards
$p(s_{t+1}|s_t, a_t)$ transition probability
$p(o_{t+1}|s_{t+1})$ observation probability
$b(s_t)$ distribution over possible states



challenge: tractable only for very simple cases

Dialog Management in Socialbots

Challenges

- Open-domain and mixed-initiative
  - user anticipates many conversation activities and topics
  - complex dialog control
- Non-task-oriented
  - the notion of “task success” is vague
  - difficulty in defining a reward function

more discussions in the “System Evaluation” lecture

Challenge of Complex Dialog Control

- A common management strategy is to break down the problem into a set of interaction modes.
  - Individual interaction modes are handled by corresponding components.
  - A master component is usually used to choose the target interaction modes.
- Different ways of break-down have been used:
  - by topics
  - by conversation activities
  - by response generation methods

Hierarchical Dialog Management in Sounding Board

- Dialog Context Tracker
  - dialog state, topic/content/miniskill history, user personality
- Master Dialog Manager
  - miniskill polling
  - topic and miniskill backoff
- Miniskill Dialog Managers
  - miniskill dialog control as a finite-state machine
  - retrieve content & build response plan

- Greet
- List Topics
- Tell Fun Facts
- Tell Jokes
- Tell Headlines
- Discuss Movies
- Personality Test
- …

Finite-State-Based Dialog Control in Socialbots

- Many socialbots use finite-state-based dialog control
  - The dialog state can be defined based on the progress of a specific conversation activity, and it constitutes a portion of the overall dialog context.
  - The state transitions rely on the dialog context maintained by the dialog manager.
- The state transitions in current socialbots are mostly hand-crafted
  - Allow non-deterministic transitions

Other Commonly Used Techniques

- Artificial Intelligence Markup Language (AIML)
- Response Retrieval

In practice, a hybrid approach is usually used which can involve more than one techniques

Artificial Intelligence Markup Language

- Dialog control is handled by the AIML interpreter using AIML files that contain a collection of knowledge units
- Each knowledge unit defines
  - a pattern to match the user utterance
  - a list of possible bot responses
  - conditions that help the interpreter to select the response to the matched user utterance
- Most knowledge units define two-turn conversations
- Multi-turn control can still be achieved using long-term context variables

AIML Basic Tags

```
<aiml version = "1.0.1" encoding ="UTF-8"?>
  <category>
    <pattern>HI</pattern>
    <template>
      <random>
        <li> Hello! </li>
        <li> Hi! Nice to meet you! </li>
      </random>
    </template>
  <category>
</aiml>
```

user utterance pattern

bot responses

- User: Hi
- Bot: Hi! Nice to meet you!
- User: Hi
- Bot: Hello!

AIML Context Variables

```
<aiml version = "1.0.1" encoding = "UTF-8"?>
  <category>
    <pattern>I am *</pattern>
    <template>
      Hello <set name = "username"> <star/>! </set>
    </template>
  </category>
  <category>
    <pattern>Good Night</pattern>
    <template>
      Good Night <get name = "username"/>! Thanks for the conversation!
    </template>
  </category>
</aiml>
```

- User: I am Allen
- Bot: Hello Allen!
- User: Good Night
- Bot: Good Night Allen! Thanks for the conversation!

AIML <condition> Tag

```
<aiml version = "1.0.1" encoding = "UTF-8"?>
  <category>
    <pattern> HOW ARE YOU FEELING TODAY </pattern>
    <template>
        <condition name = “mood" value = "happy">
         I am happy!
        </condition>
        <condition name = " mood " value = "sad">
         I am sad!
        </condition>
    </template>
  </category>
</aiml>
```

- …
- <set name = “mood"> happy </set>
- …
- User: How are you feeling today
- Bot: I am happy!

AIML

- Advantages: simplicity
  - Used by many socialbots to code the dialog control rules and bot responses
- Issues
  - Difficult to handle all kinds of user requests
  - Less flexible for executing complex actions such as querying back-end databases and APIs

Response Retrieval

- Retrieve human-written responses for the current user utterance
  - directly obtain well-formed responses without the need of realization or generation
- Both dialog control and dialog context tracking are heavily integrated into the retrieval process

Response Retrieval

- Retrieval methods
  - Learned retrieval models
  - Similarity-based using pre-trained embeddings
  - Entity matching
  - Search engine or API
- Sources of human-written responses
  - Responses mined from social media (Twitter, Reddit, …)
  - Public dialog corpus (Cornell movie dialog corpus, DailyDialog, …)
  - Crowd-sourced

Current Directions

- Statistical dialog control for open-domain systems
  - Reinforcement learning
  - End-to-end learning
  - Combination with neural networks
  - …
- Data collection methods for socialbots
  - Recruit two workers to chat with each other
  - Recruit workers to chat with a bot
  - Recruit workers to create a dialog by playing the role of both participants
  - Recruit workers to extend an existing conversation by one turn (Wizard-of-Oz)
  - …