Norsk Regnesentral
NORWEGIAN COMPUTING CENTER

Dialogue management, system design & evaluation

Pierre Lison

IN4080: Natural Language Processing (Fall 2020)

19.10.2020

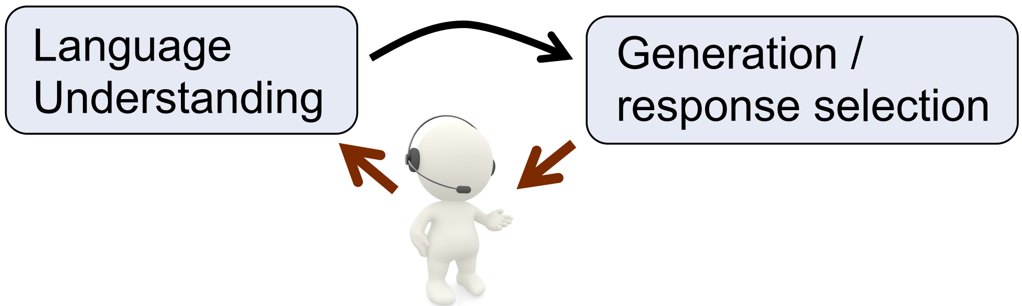
Plan for today

- Dialogue management
  - Handcrafted approaches
  - Data-driven approaches
- Design of dialogue systems
  - Architectures
  - Evaluation

Plan for today

- **Dialogue management**
  - Handcrafted approaches
  - Data-driven approaches
- Design of dialogue systems
  - Architectures
  - Evaluation

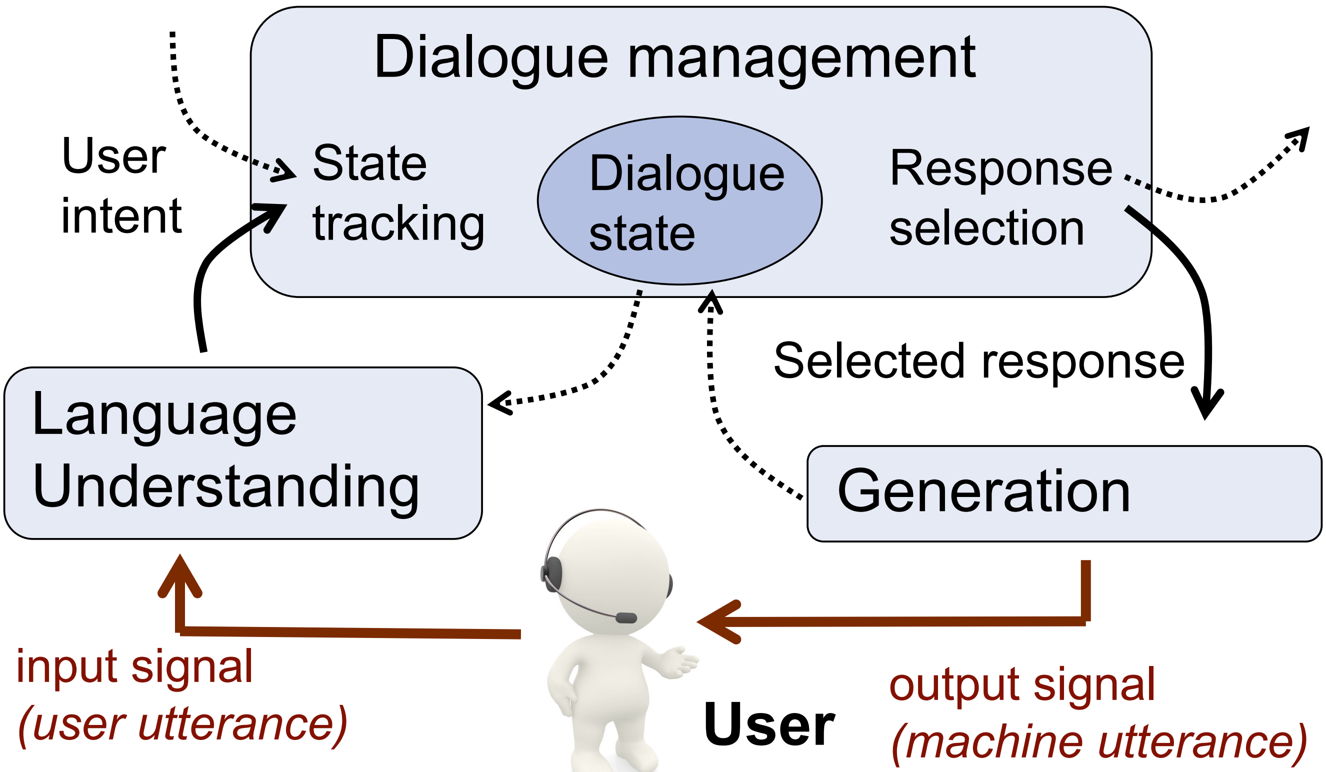
Basic architecture



This pipeline is often used for chatbots

- **Main limitation**: no management of the dialogue itself (beyond current utterance)
- Most appropriate for short interactions

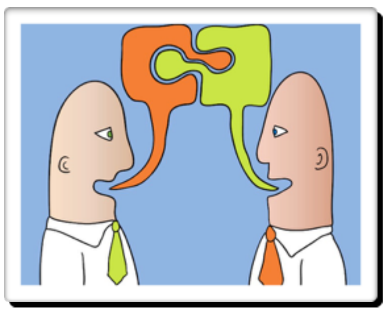
More advanced architecture



5

Dialogue manager

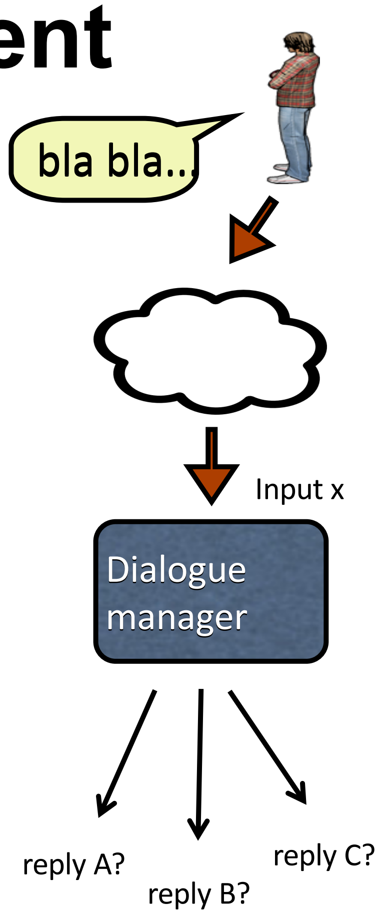
- The dialogue manager is responsible for controlling the *flow* of the interaction
- Conversational skills to emulate:
  - Interpret utterances *contextually*
  - Manage *turn-taking*
  - Fulfill conversational *obligations* & social *conventions*
  - *Plan* multi-utterance responses
  - Manage the system *uncertainty*



Dialogue management

… is about **decision-making:**

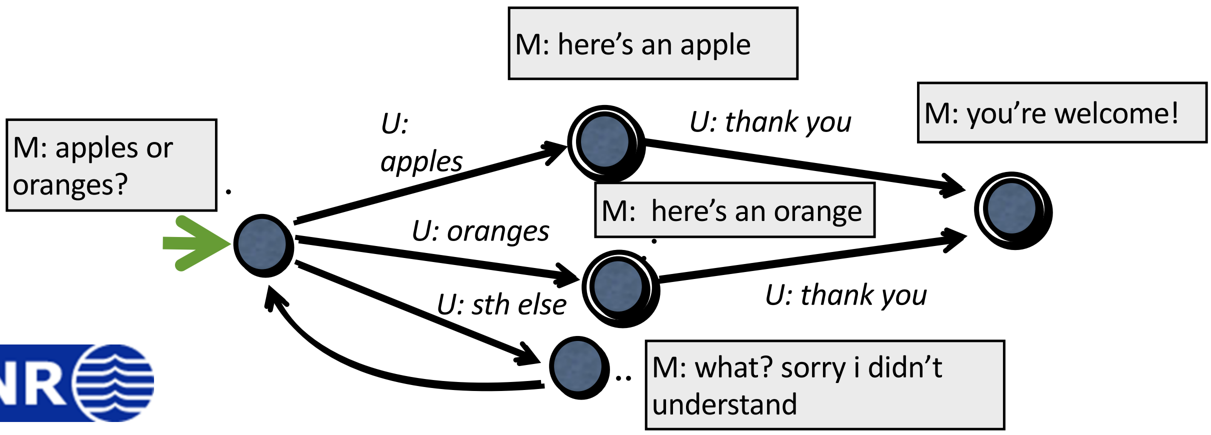
- i.e. what should the system decide to *say* or *do* at a given point
- decision-making *under uncertainty*, since the communication channel is “noisy” (errors, ambiguities, etc.)
- Actions can be both linguistic and non-linguistic (booking a flight ticket, picking up an object, etc.)
- The same holds for observations (visual input, external events, etc.)



Finite-state automata

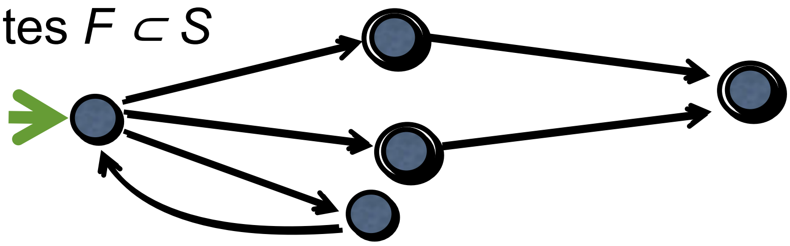
The simplest approach is to encode dialogue strategies as **finite-state automata**

- the nodes represent *machine actions*
- and the edges possible (mutually exclusive) *user responses*



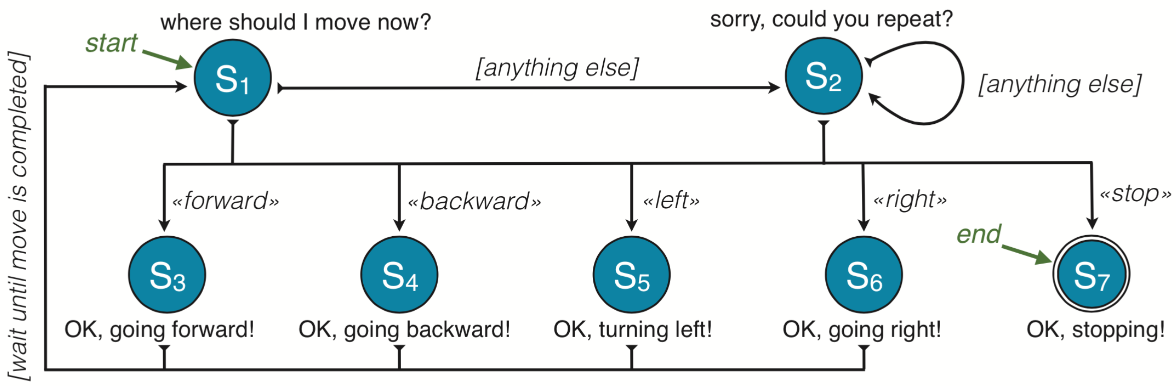
Formalisation of an FSA

1. Finite, non-empty set *S* of (atomic) *states*, each associated with a specific machine action.
2. A finite, non-empty set Σ of possible *user inputs* accepted by the automaton
3. A (partial) function $\delta : S \times \Sigma \rightarrow S$ defining the *transitions* between states
4. An *initial* state $s_0 \in S$
5. A set of *final* states $F \subset S$



Finite-state automata

- ► Transitions can relate to other signals than user inputs (for instance, external events)
- ► And can also express complex conditions (pattern matching on the user input, confidence thresholds, etc.)



Finite-state automata

Advantages	Limitations
• Easy to design • Fast, efficient • Does not require dialogue data • *Predictable* system behaviour (both for the user and for the system designer)	• Only allows for *scripted* interactions - not "true" conversation • No principled account of uncertainties • Difficult to scale to complex domains with many variables and alternative inputs

Frame-based managers

- ► The interaction flow can be made slightly more flexible in *frame-based systems*
- ► The state is represented as a **frame** with **slots** to be filled by the user's answers

Slot	Question
ORIGIN CITY	«From what city are you leaving?»
DESTINATION CITY	«Where are you going?»
DEPARTURE TIME	«When would you like to leave?»
ARRIVAL TIME	«When do you want to arrive?»

Frame-based managers

- ► The user will sometimes provide additional information to the system's questions

  System: What is your departure?
  User: I want to leave from Oslo before 9:00 AM»

- ► The system should fills the appropriate slots with all available information
- ► **VoiceXML**: *Voice-extensible Markup Language*
  - Markup language for basic slot-filling systems
  - Allows mixed initiative

VoiceXML

```
<form>
	<field name="transporttype">
		<prompt>Please choose airline, hotel, or rental car. </prompt>
		<grammar type="application/x=nuance-gsl">
			[airline hotel "rental car"]
		</grammar>
	</field>
	<block>
		<prompt>You have chosen <value expr="transporttype">.
		</prompt>
	</block>
</form>
```

Logic-based reasoning

- ► Difficult to capture complex interactions with finite-state automata or frames
  - Crude notion of a *dialogue state*
  - Crude notion of a *dialogue state transition*: only a few «hard» transitions possible for each node
- ► Possible solution: use richer (more expressive) representations of the state
  - & enable more sophisticated forms of *reasoning*

Logic-based reasoning

- ► «*Information-state update*» (ISU) is an example of approach based on a rich state representation
  - Encodes the mental states, beliefs and intentions of the speakers, the common ground, dialogue context
- ► This state is read/written by two types of rules:
  - *Update rules* modify the current state upon the observation of new user dialogue move
  - *Action selection rules* then select the system action based on the information present in this updated state

[S. Larsson and D. R. Traum (2000), «Information state and dialogue management in the TRINDI dialogue move engine toolkit» in *Natural Language Engineering*]

Logic-based reasoning

Advantages	Limitations
• Rich representation of the dialogue state that can capture user intents, background knowledge, grounding status, etc. • Powerful tools for interpretation & decision • Can (in theory) perform long-term planning	• No account of uncertainty • Requires detailed descriptions of the dialogue domain • More difficult to design (logical abstractions) • Hard to scale!

Interaction style

- ► Rigid, repetitive structure of the interaction
- ► Irritating confirmations & acknowledgements
- ► No user or context adaptivity

“Saturday night live” sketch comedy, 2005

Plan for today

- Dialogue management
  - Handcrafted approaches
  - **Data-driven approaches**
- Design of dialogue systems
  - Architectures
  - Evaluation

Data-driven techniques

The approaches presented so far suffer from a number of limitations:

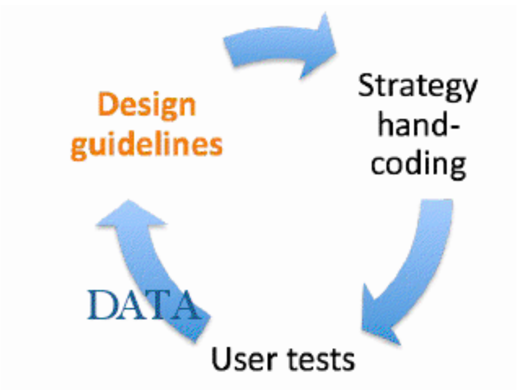
- Difficult to predict the user behaviour in advance
- They ignore all the *uncertainties* appearing through the dialogue (ASR errors, ambiguities, etc.)
- Unable to *learn* or adapt to the users or the environment (leading to rigid/repetitive behaviour)
- Limited to one goal... but real interactions are trade-offs between *various competing objectives*

Data-driven techniques

- ► *Solution*: perform **automatic optimisation** of the «dialogue policies» from experience:
  - Often based on *reinforcement learning* techniques
  - "Experience": interactions with real or simulated users
- ► General procedure:
  - Dialogue manager starts with «dumb» dialogue policy
  - It interacts with users and receives a **feedback**
  - It can then correct his policy based on this feedback
  - Repeat process until policy is fully optimised

Data-driven techniques

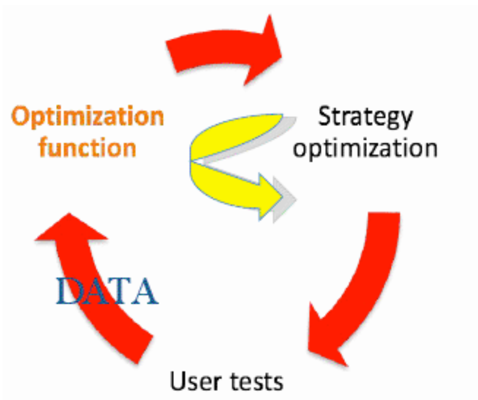
Conventional software life cycle



Design by "Best practices"

(Paek 2007)

Automatic strategy optimisation



Automatic design by optimization function

(= "programming by reward")

[slide borrowed from O. Lemon]

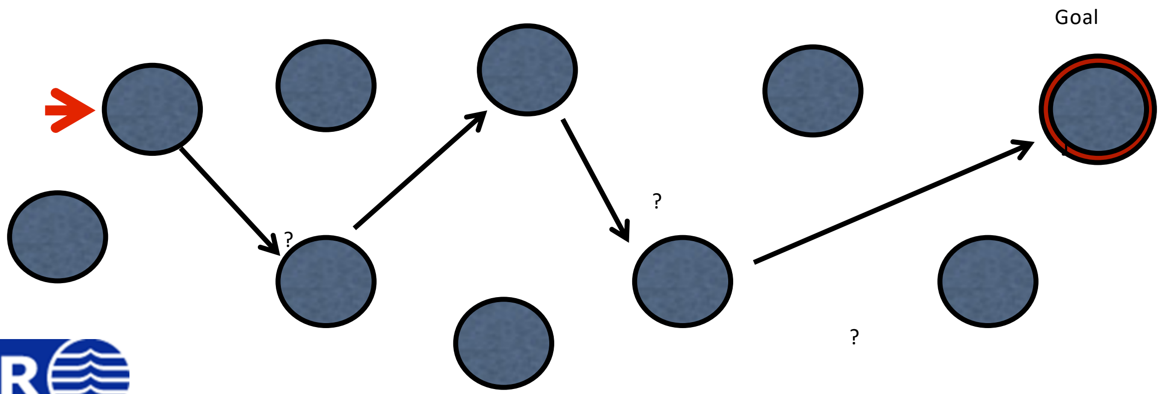
Data-driven techniques

- Dialogue management is again viewed as a **planning/control** problem:
  - Agent must control its *actions*
  - To reach a long-term *goal*
  - In an uncertain *environment*
  - Where there are many possible *paths* to the goal
  - ... and complex *trade-offs* need to be determined
- But this time, planning includes *multiple goals* (encoded in *rewards*), is performed *under uncertainty*, and is *learned* from the agent experience

Data-driven techniques

Planning problems are generally defined with three components:

- A **state space** (the set of all possible states)
- An **action space** (the set of all possible actions)
- The **goals** for the task (encoded here with rewards)



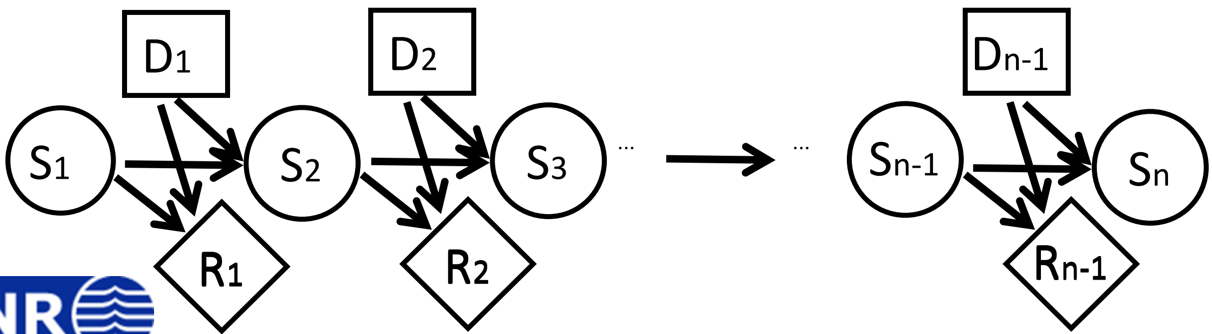
Data-driven techniques

- Most tasks have to encode trade-offs between various, competing objectives
  - A flight booking system must book the right ticket
  - But it must do so with the fewest number of requests
- Typically encoded via **rewards** (utilities) associated to particular state/action pairs

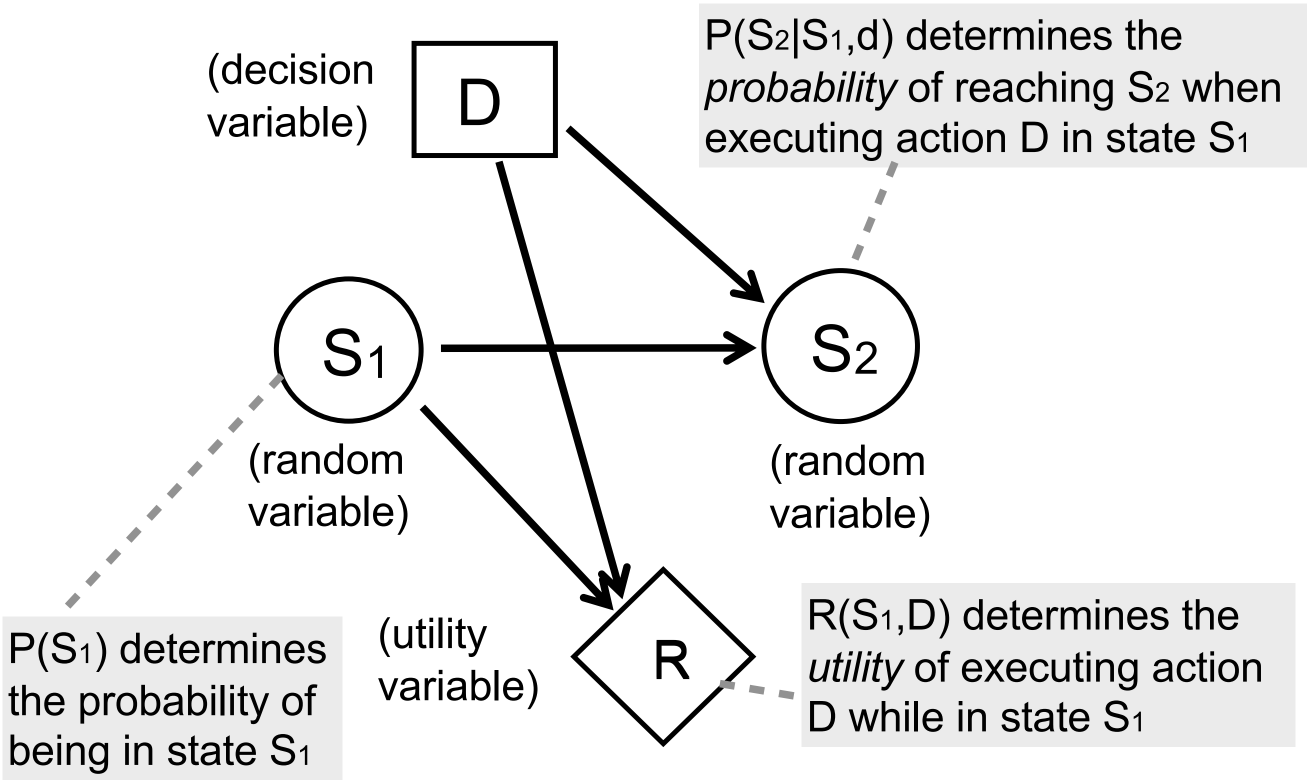
State	Action	Reward
User wants to book ticket *x*	Booking x	+10
User wants to book ticket *x*	Booking y ≠ x	−30
User wants to book ticket *x*	Clarification request	−1

Markov Decision Processes

- We can define these ideas more precisely using a formalism called **Markov Decision Processes** (MDPs)
- Markov Decision Processes are an extension of Markov Chains where the agent *selects an action* at each state
  - This action will then modify the state space
  - And will yield a particular reward for the agent



Graphical notation



Markov Decision Processes

A MDP is as a tuple **<S,A,T,R>**, where:

- **S** is the *state space* (possible states in the domain)
- **A** is the *action space* (possible actions for the agent)
- **T** is the *transition function*, defined as $T(s, a, s') = P(s'|s, a)$. It is the probability of arriving to state **s'** after executing action **a** in state **s**.
- **R** is the *reward function*, defined as $R : S \times A \rightarrow R$. It is a real number encoding the utility for the agent to perform action **a** while in state **s**.

Expected cumulative reward

- In an MDP, the agent seeks to maximise its *expected cumulative reward* **Q(s,a)**

The agent must try to predict future inputs/rewards

The rewards accumulate over time

- How much worth is a reward expected at time (t+i) compared to one received right now?
  - We use a *discount factor* γ to capture this balance
  - Related to *delayed gratification* in psychology

Bellman equation

The *Bellman equation* tells us that we can write the expected cumulative reward Q in a recursive fashion:

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

Notice that we are estimating the Q-values based on... our estimation of the Q-values (can be used to iteratively refine these estimates until convergence)

[R. Bellman (1957): «*Dynamic Programming*»]

MDP policy

- Given an MDP, a (dialogue) policy tells us which action to execute in each state
- A dialogue policy is a *mapping* $\pi: S \rightarrow A$ from states to actions
- An *optimal* dialogue policy π^* is a policy that always outputs the action yielding the maximum expected cumulative reward:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

Reinforcement learning

- **Reinforcement learning** can help us learn these Q values through interaction
- They work by iteratively refining their estimate of the Q values
  - The agent acts in the environment and observes both states and rewards
  - This operation is repeated until convergence
- In dialogue systems: policy learning can be done either in simulation or with real users

[R. Sutton & A. Barto (2018): «*Reinforcement Learning: An Introduction*»]
(complete book available online!)

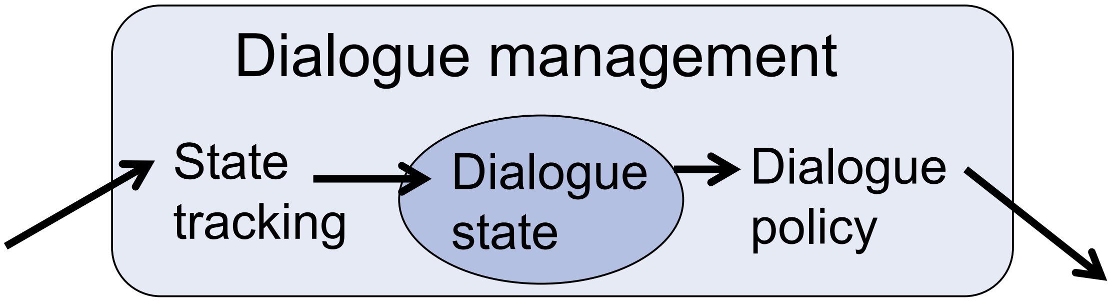
Partially observable MDPs

- ► In an MDP, we assume the current (dialogue) state is fully observable
  - ▪ We may be uncertain about the future, but the current state is assumed to be known with certainty
  - ▪ Often not a reasonable assumption in dialogue!
- ► We can extend MDPs to *Partially Observable Markov Decision Processes* (POMDPs)
  - ▪ In a POMDP, we have a probability distribution P(s) over possible current states

Partially observable MDPs

- ►In a POMDP, : the "true" dialogue state is not directly observable but can only be inferred from observations.
- ►This is expressed by the **belief state**, which represents the information known to the agent
- ►The dialogue policy is then defined as a mapping from *belief states* to *actions*
  - ▪ Much trickier to learn than MDP policies!

(Belief) state tracking



- The belief state is regularly updated with new observations (from e.g. NLU)
- In recent systems, belief state tracking and NLU are often one single (neural) model

Plan for today

- Dialogue management
  - Handcrafted approaches
  - Data-driven approaches
- **Design of dialogue systems**
  - Architectures
  - Evaluation

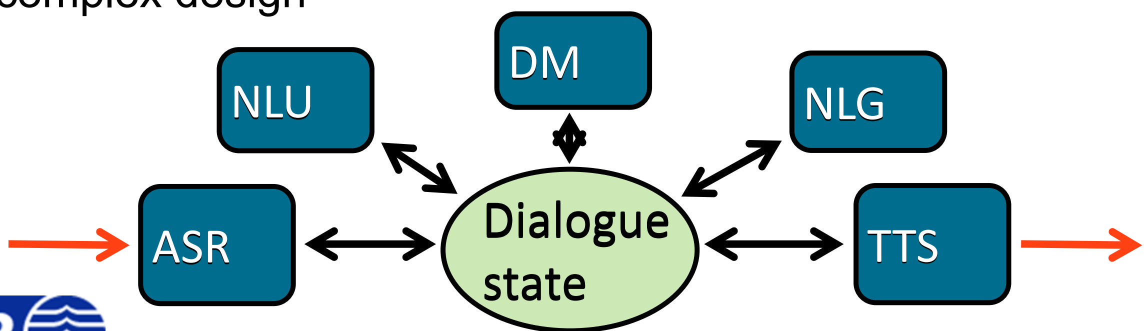
Pipeline architectures

- Components connected in processing chain
- Each component is a black box getting inputs from its predecessor and generating an output

Limitations:
- No feedback between components
- Rigid information flow
- Poor turn-taking behaviour (system does not react until the full pipeline has been traversed)

Blackboard architectures

- Revolves around a *blackboard* (dialogue state) and a set of components
- Modules listen for relevant changes, in which case they do some processing and update the state with the result
- Better information flow and reactivity, but more complex design



Incrementality

Humans process and produce language **incrementally:**

- When listening, we don't wait for an utterance to be fully pronounced to process it!
- We gradually refine our understanding as we go, phoneme by phoneme
- We also continuously provide feedback signals

Human-human dialogues are full of *interruptions*, speech *overlaps*, *backchannels*, and *co-completion* of utterances

Incrementality

- But most dialogue systems operate in «batch mode»
  - NLU expects full utterance as input
  - TTS waits for complete system response to start synthesis
- Leads to «ping-pong» turn-taking behaviour:
  - Alternating turns between user & system, one speaker at a time

Can dialogue systems be made to work *incrementally*, on partial units of content?

[Schlangen, D., & Skantze, G. (2011). A general, abstract model of incremental dialogue processing. *Dialogue & Discourse*]

How to collect data?

► "Chicken-and-egg" problem:

- Need data to train data-driven models
- But to collect data, we need a system that can interact with users

► One solution is to use *Wizard-of-Oz* studies*:*

- Replace the system with a human operator (without the users being aware of it)

Evaluation

► Some dialogue processing tasks have standard evaluation metrics:

- ASR: *Word Error Rate*
- NLU: *[precision, recall, F-score]* for intent recognition and slot-filling
- TTS: evaluation by human listeners on sound intelligibility and quality

► But how do we evaluate the end-to-end the conversational behaviour of the system?

Evaluation

One way to evaluate is via **user satisfaction ratings**

The ratings can be obtained from surveys that users are asked to fill after interacting with the system:

TTS Performance	Was the system easy to understand ?
ASR Performance	Did the system understand what you said?
Task Ease	Was it easy to find the message/flight/train you wanted?
Interaction Pace	Was the pace of interaction with the system appropriate?
User Expertise	Did you know what you could say at each point?
System Response	How often was the system sluggish and slow to reply to you?
Expected Behavior	Did the system work the way you expected it to?
Future Use	Do you think you'd use the system in the future?

[M. Walker et al. (2001), «Quantitative and Qualitative Evaluation of Darpa Communicator Spoken Dialogue Systems», *Proceedings of ACL*]

Evaluation

- ► However, user evaluation surveys are expensive and time-consuming
  - Not feasible to conduct after each system change!
  - Can we automate the evaluation process?
- ► Solution: rely on metrics that can be extracted from interaction logs, and are known to *correlate with user satisfaction*
  - Improving these observable metrics should therefore increase user satisfaction

[M. Walker et al. (1997), "PARADISE: A general framework for evaluating spoken dialogue agents", Proceedings of ACL]

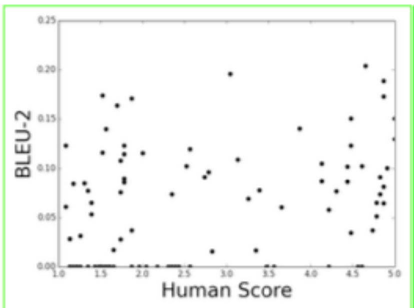
Evaluation

Criteria	Description	Possible metrics
Task completion success	How often did the system complete its task successfully?	- κ agreement on slots - completion ratio
Efficiency costs	How efficient was the system in executing its task?	- nb of turns (from user, system, or both) - total elapsed time
Quality costs	How good was the system interaction?	- nb of ASR rejection prompts - nb of user barge-ins - nb of error messages

NB: this list of metrics is of course not exhaustive!

Evaluation

- Can't we use metrics like BLEU to compare system outputs with human responses?
  - **No**: very weak correlation between BLEU scores and human judgments!
- But alternative metrics exist, like ADEM



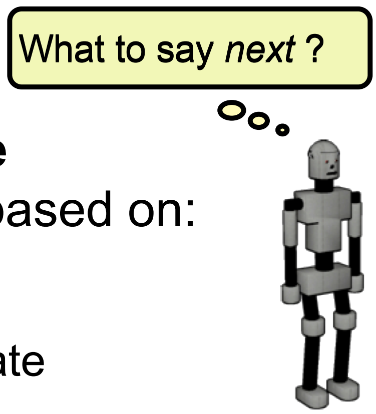
[Lowe et al. (2017). Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses. In *ACL*.]

[Liu et al (2016). How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation. In *EMNLP*.]

Plan for today

- ▶ Dialogue management
  - Handcrafted approaches
  - Data-driven approaches
- ▶ Design of dialogue systems
  - Architectures
  - Evaluation
- ▶ **Summary**

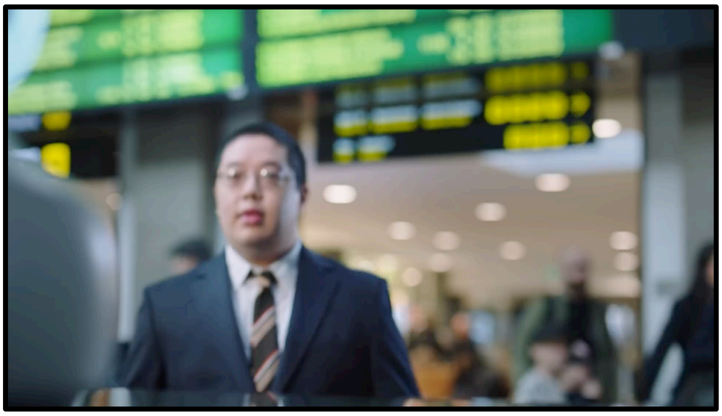
Summary



- ▶ Dialogue management = **decide** what to do/say at a given time, based on:
  - System goals (and trade-offs)
  - Current (uncertain) dialogue state
- ▶ Various approaches:
  - Easiest (but quite rigid): *finite-state* approaches
  - *Frame-based* systems (slightly) more flexible
  - Statistical/neural approaches *optimise* dialogue policies from (real/simulated) interactions
- ▶ Evaluation via objective and subjective metrics

What we haven't covered

- Natural language generation (NLG)
- Speech synthesis
- Multimodal & situated systems



Furhat robot (initially developed at KTH, Stockholm), see www.furhatrobotics.com