Language Technology II: Dialogue Learning

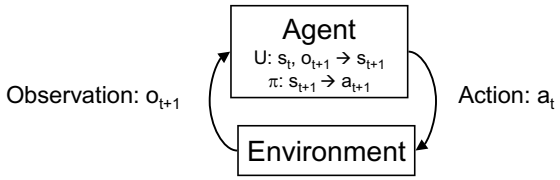
Summer 2008

Manfred Pinkal

Dialogue Design

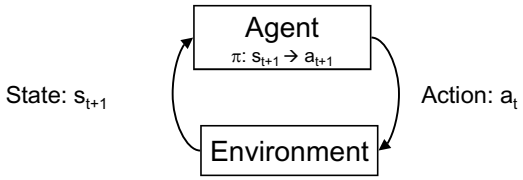
- Global design decisions:
  - Specify a set of dialogue states S, the State Space
  - Specify a set of possible system actions, the Actions Set
  - Range of possible actions A that can be in principle carried out in a state s.
- Dialogue strategy:
  - Specify a dialogue strategy or dialogue policy π, which defines an action $a \in A$ of the dialogue system for each possible dialogue state $s \in S$.
  - $\pi: S \rightarrow A$
- The principal problem of dialogue design:
  - Find a policy π which is optimal with respect to the purposes of the dialogue.

A simple model for a decision process



- The agent interacts with a stochastic environment:
  - At time t, it carries out an action a_t, which (non-deterministically) influences the subsequent behaviour of the environment.
  - It observes behaviour o_t of the environment, which (deterministically) leads to an update of the state:
    $U(s_t, o_{t+1}) = s_{t+1}$
  - According to strategy π, it (deterministically) selects and carries out an action s_{t+1}
    $\pi(s_{t+1}) = a_{t+1}$

Simple model – further simplified

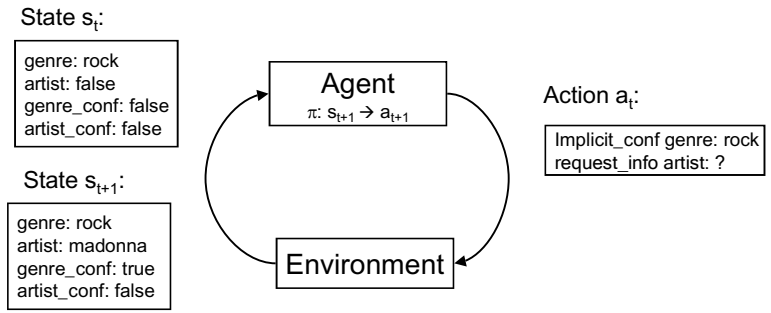


- The environment is everything which is outside the immediate access of the agent.
- The state is not the real state of the environment, but represents the agent's view of the environment (its „context model").
- This allows to assume a deterministic update function, and to simplify the scheme by assuming that th new state is directly observed.

Dialogue as a decision process

- States: Atomic states in simple finite-automaton dialogue models, in more elaborate models, e.g. ISU, defined by the current statur of a finite number of predefined state variables:
  - Task-level information like features for filled and confirmed slots in information-seeking dialogue.
  - Current user input information, like user's dialogue act, and ASR convidence values
  - Information about user,e.g., user expertise, preferences, beliefs
  - Information about dialogue history
- Actions: Dialogue moves, typically specified on an abstract level by:
  - a set of speech/dialogue act types: „request_info", „implicit confirm", „present_info"
  - A slot name (e.g.: „time of-departure", „genre")
  - A slot value
  - Example: „implicit_confirm genre: rock"
- Environment: Includes everything that is not in direct control of the dialogue manager, including the user, the general situation (noise, distraction), the ASR system.

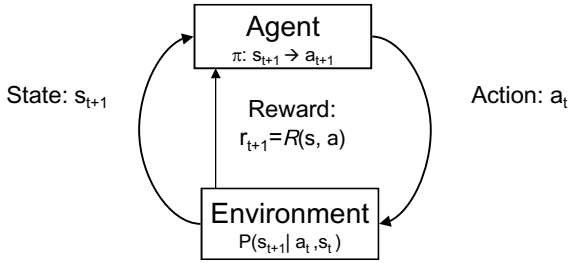
An Example



Markov Decision Process (MDP)

- Part of the specific dialogue policy used in this example might be a rule like:
  - „If a slot value is filled but unconfirmed, and there are further unfilled slot values, select an utterance that asks for information about the next slot, at the same time implcitly confirming the filled one".
  - By which artist do you want to hear a rock song?
- How do we evaluate a specific policy? How do we get at the optimal policy?
- The next step to the answer is „Markov Decision Processes"
- MDPs extend the simple model looked at so far by
  - A state transition function, giving a probabilistic model for the dynamics of the environment
  - A reward function, giving feedback about the advantage of an action and its resulting state.

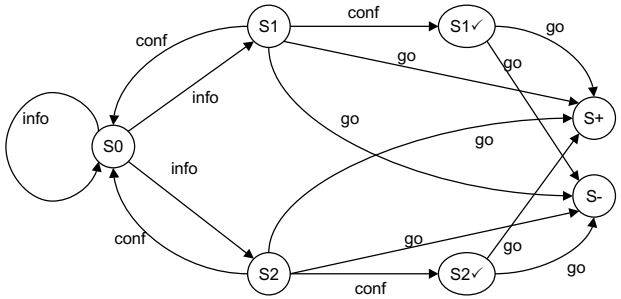
Markov Decision Processes (MDP)



Extension of a simple decision process by:

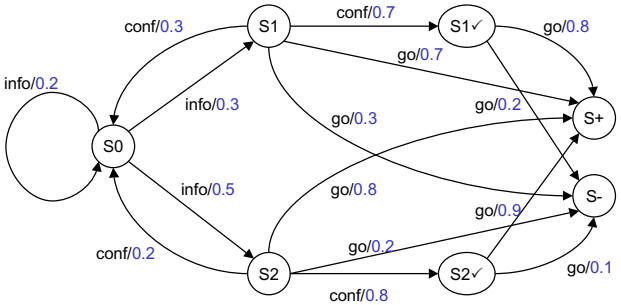
- State transition function T: $S \times A \times S \rightarrow [0,1]$
  - $T(s, a, s') = P(s'|a, s)$, the probability that action a causes the environment to change from s to s'.
- Reward function R:
  - $R(s, a)$ is a real number, specifying the expected advantage of reaching state s through action a

An example: Elevator dialogue

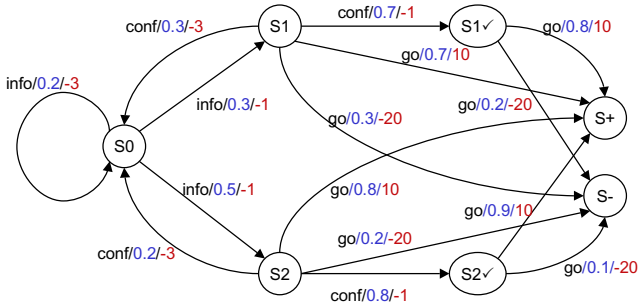


- Simple Decision Process: The environment's behaviour can be modelled as a non-deterministic finite automaton.
- Agent's actions are input symbols. Transitions give possible (non-deterministic) environments reactions.
- On the previous slide you see a model of a simple elevator dialogue:
  - Two floors
  - Three available dialogue acts:
    - Information Request: „Which floor do you want to go?"
    - Confirmation: „Is xth floor correct?"
    - Command to elevator driver
  - Seven states:
    - Initial (S0)
    - User wants floor x - uncomfirmed (S1, S2)
    - Floor x - comfirmed (S1✓, S2✓)
    - Correct/incorrect command executed (S+, S-)

Adding weights



Adding immediate rewards



Here, reward values are assigned according to considerations about what might affect dialogue quality:
Each move makes dialogue longer: Reward -1
Recognition failures are unpleasant: Reward -3
Final task success is fine: Reward +10
Final task failure is very annoying: Reward -20

Learning the optimal policy π

- The immediate reward can be used to directly learn successful dialogue behaviour.
- But the best solution of a task as a whole may not be achieved by selecting locally best decisions.
- Global quality measures (like PARADISE) are not easily translatable into local design decisions.
- A solution to the question of how to combine local and global quality measures/rewards to improve strategy design is Reinforcement Learning.

Immediate and Cumulative Reward

- The reward function r(s, a) gives the immediate reward for every state and action, standing for its intrinsic desirability.
- The cumulative reward of an individual interaction:
  - $R(<s_1, a_1, s_2, a_2, s_3, a_3, \ldots>) = r(s_1, a_1) + \gamma\, r(s_2, a_2) + \gamma^2\, r(s_3, a_3) + \ldots$
  - γ is a discount factor between 0 and 1, intended to model the fact that imediate rewards are potentially more important for a decision than those received in a more distant future.
  - γ=0: cumulative reward is immediate reward

Expected Cumulative Reward

- The expected return of taking action *a* in a given state *s* and following π therafter is written as $Q^{\pi}(s,a)$ and defined by:

$$Q^{\pi}(s,a) = \sum_{s' \in S} P(s' \mid s,a) * [r(s',a) + \gamma * \max_{a' \in A} Q^{\pi}(s',a')]$$

- It is computed as the sum of the immediate reward and the discounted sum of the values for possible result states, weighted by the probability of these states (given *s* and *a*).
- The value of a result state is computed under the assumption that optimal action is taken by the agent.
- Reinforcement learning for dialogue typically sets the discount factor γ to 1 (or close to one).
- The optimal policy is described by the „Bellmann optimality equation“:

$$Q^{*}(s,a) = \sum_{s' \in S} P(s' \mid s,a) * [r(s',a) + \gamma * \max_{a' \in A} Q^{*}(s',a')]$$

- It can be effectively computed if full knowledge of the dynamics of state transition is available.
- In practical applications, heuristic algorithms are often used to approximate the Q-function.

Reinforcement learning

- Agent‘s knowledge about the environment‘s behaviour is encoded in a weighted finite automaton (see earlier slide), whose transition probabilities are obtained from a dialogue corpus taken from HMI or WoZ experiments.
- This knowedge can be used in two different ways:
  - Offline computation of the optimal Q-function, using Bellmann equation --> Model-based Reinforcement learning
  - Simulation of an artificial user --> Simulation-based Reinforcement learning

Model-based RL

- Produce a dialogue corpus by HMI or WoZ experiments
- Let subjects evaluate with questionnaire (SASSI, PARADISE), take results as final rewards.
- Compute state transition probabilities (the *T* function) from the corpus
- Compute the optimal *Q*-value offline on the basis of state transition model and rewards
- Advantages:
  - Realistic data through real dialogue logging and user evaluation
  - The obtained policy is guaranteed to be optimal w.r.t. the data
- Disadvantages:
  - Computation may be very time-consuming
  - Sparse data: Too few observed transitions for a reliable estimation of state transition probabilities for dialogue models of realistic size.
  - Only observed state-action combinations can be expored: No exploration of novel strategies, no guarantee that the optimal strategy occurs in the data.

Simulation-based RL

- Learning through simulation of (large numbers of) dialogues.
- Also based on transition probabilities computed on the basis of a dialogue corpus.
- These are not used for direct computation of Q-value, but for simulating environment behaviour/an artificial user.
- Conduct a large number of simulated dialogues
- Use utterances from simulated user
- After each system action, compute an expected reward for the state/action pair
- Repeated updating “optimises” the policy.

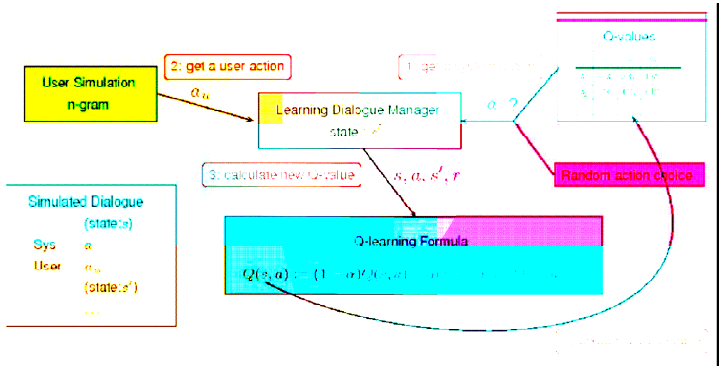
Policy Representation

- We maintain a table of Q-values: expected rewards for state/action pairs.

	s1	s2	s3
a1	1.75	0.00	0.00
a2	**2.22**	**4.13**	0.52
a3	-7.00	-5,35	**8.75**

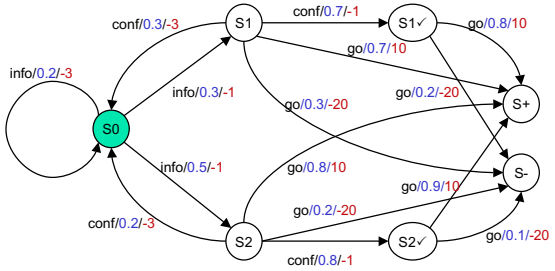
- The dialogue strategy for a given state is then the action with the highest Q-value (highlighted in the above table).
- The result of RL will be an optimised Q-value table, from which the optimal strategy can be read off.
- For the learning process, Q-values are either set to an arbitrary value (e.g., 0), or randomly initialised.

The Basic Q-learning Cycle



Q-Learning Algorirhm: An example

•Set s to S0

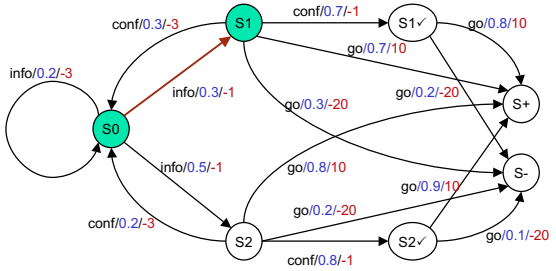


	S0	S1	S1✓
info	0.00	0.00	0.00
conf	0.00	0.00	0.00
go	0.00	0.00	0.00

Example Cont‘d

•Choose action from S0
- ➢ Only available action is *info*

•Take the action, observe resulting state s‘ and reward r.
- ➢ s‘ happens to be S1(chance of 30%), r= -1

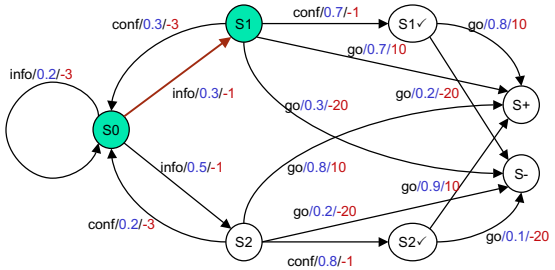


	S0	S1	S1✓
info	0.00	0.00	0.00
conf	0.00	0.00	0.00
go	0.00	0.00	0.00

Example Cont‘d

•Compute new Q-estimate for Q(S0, info)

$$Q(s,a) := Q(s,a) + \alpha * \left(r + \gamma * \max_{a' \in A} Q(s',a') - Q(s,a)\right)$$



	S0	S1	S1✓
info	0.00	0.00	0.00
conf	0.00	0.00	0.00
go	0.00	0.00	0.00

The *Q*-Learning Formula

$$Q(s,a) := (1-\alpha) * Q(s,a) + \alpha * \left(r + \gamma * \max_{a' \in A} Q(s',a')\right)$$

- General scheme is:

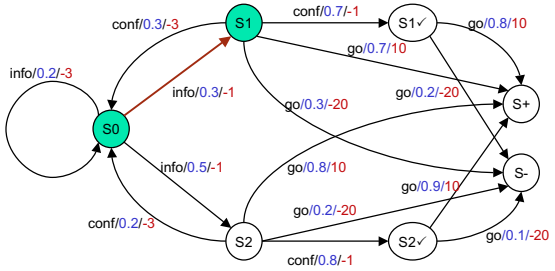
  *Q-New := (1-StepSize)*Q-Old + StepSize*(Target-Q-Old)*
  - The Q-estimate for state s and action a is set to the weighted sum of the old estimate and the Target.
  - The target is composed of the observed immediate reward r and the Q-value for the best choice of an action on state s‘, where „best choice“ means optimal choice w.r.to the current Q-estimates of the Q-table.
  - The target is weighted with a discount factor, which we will set to 1 and thus ignore in the further course of the example computation.
  - The step-size parameter α specifies the relative weight of the new observation, compared to the currently existing old estimate. It is typically set to a rather high value in the beginning of the learning experiment, and becomes consistently smaller, due to the fact that the estimate becomes more and more stable and reliable in the course of the experiment.

Example Cont'd

•Compute new Q-estimate for Q(S0, info)

$$Q(S0, Info) := 0{,}5 * Q(S0, Info) + 0{,}5 * \left(-1 + \max\left[Q(S1, conf), Q(S1, go)\right]\right)$$

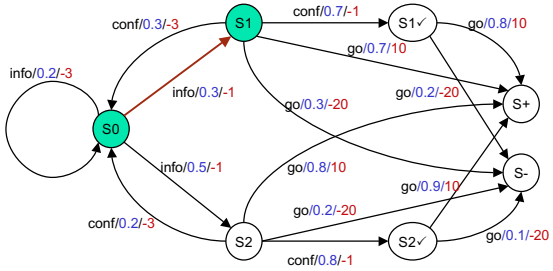
$$= 0 + 0{,}5 * \left(-1 + \max\left[0,0\right]\right) = -0{,}5$$



	S0	S1	S1✓
info	0.00	0.00	0.00
conf	0.00	0.00	0.00
go	0.00	0.00	0.00

Example Cont'd

•Replace old Q-estimate with new Q-estimate



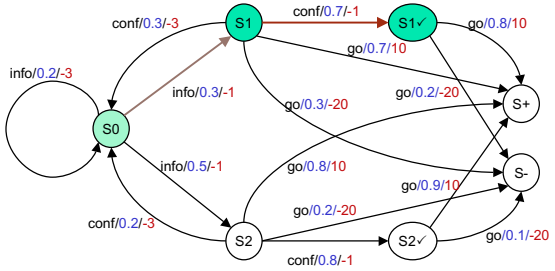
	S0	S1	S1✓
info	-0.5	0.00	0.00
conf	0.00	0.00	0.00
go	0.00	0.00	0.00

Example Cont'd

•Choose action from S1
- ➢ Action chosen is: *conf*

•Take the action, observe resulting state s' and reward r.
- ➢ s' happens to be S1✓ (chance of 70%), r= -1

➢Compute new Q-estimate for Q(S1, conf).

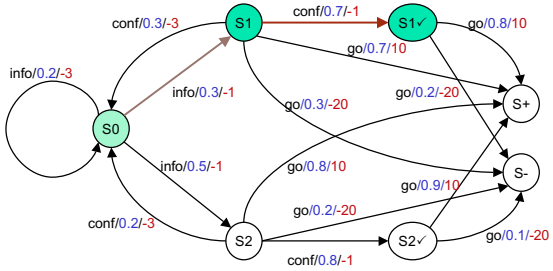


	S0	S1	S1✓
info	-0.5	0.00	0.00
conf	0.00	0.00	0.00
go	0.00	0.00	0.00

Example Cont'd

$$Q(S1, conf) := 0{,}5 * Q(S1, conf) + 0{,}5 * \left(-1 + \max\left[Q(S1v, go)\right]\right)$$

$$= 0 + 0{,}5 * \left(-1 + 0\right) = -0{,}5$$



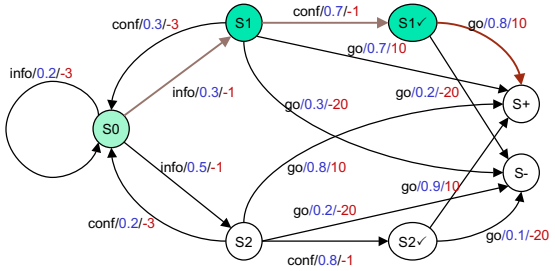
	S0	S1	S1✓
info	-0.5	0.00	0.00
conf	0.00	-0.5	0.00
go	0.00	0.00	0.00

Example Cont'd

•Choose action from S1*v*: *go*
•Take action, observe resulting state s'(S+) and reward r (+10).
• Copmpute new Q-estimate and replace old one.

$$Q(S1v,go) := 0{,}5 * Q(S1v,go) + 0{,}5 * \left(-1 + \max[\varnothing]\right)$$

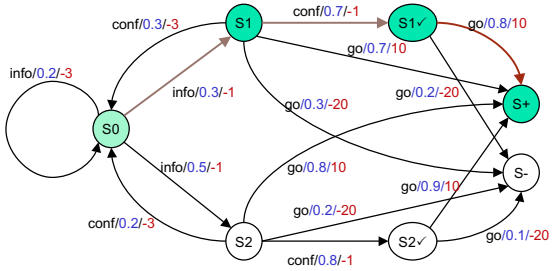
$$= 0 + 0{,}5 * (10 + 0) = 5$$



	S0	S1	S1✓
info	-0.5	0.00	0.00
conf	0.00	-0.5	0.00
go	0.00	0.00	+5.0

Example Cont'd

•If the second and third walk through the dialogue are identical to the first one (i.e., if the system selects the same actions, an the environment happens to react in the same way), the Q-table is updated as can be seen below.

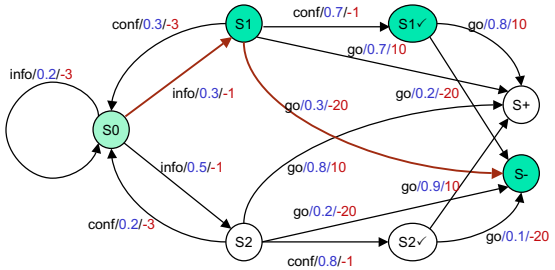


	S0	S1	S1✓
info	-0.75	0.00	0.00
conf	0.00	1.75	0.00
go	0.00	0.00	7.50

	S0	S1	S1✓
info	0.00	0.00	0.00
conf	0.00	4.13	0.00
go	0.00	0.00	8.75

Example Cont'd

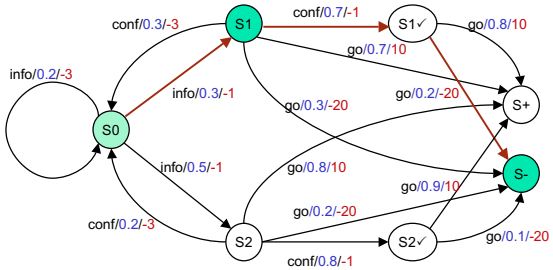
•Now, let the system try a direct, unconfirmed go:
the environments reaction reaction (ASR failure) may lead to a false destination trip of the elevator.



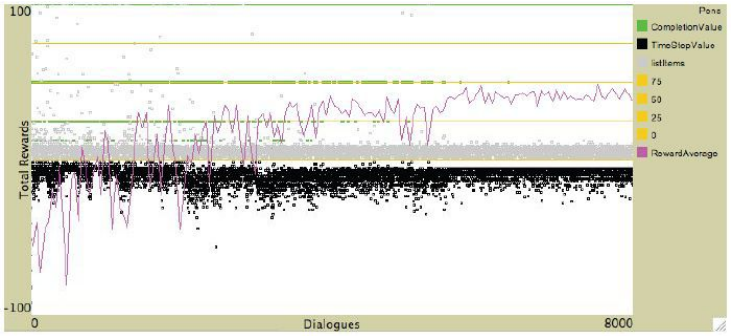
	S0	S1	S1✓
info	1.75	0.00	0.00
conf	0.00	4.13	0.00
go	0.00	-10	8.75

Example Cont'd

•But cautious version with confirmation may also fail ...



Exaple for a Policy Learning Curve



An experiment with 8,000 dialogues
(From Verena Rieser's Dissertation)

Simulation-based RL

- Advantages:
  - Allows exhaustive exploration of the space of possible policies
  - Allows exploration of completely novel strategies
  - Some global design decisions (state space, action set) can be changed without great effort.
- Problems:
  - Quality of learning strategy depends on quality of simulated environment.
  - Reward function must be explicitly constructed.
  - User simulation is based on n-gram probabilities: Global incoherence of simulated user behaviour not excluded
  - Is the optimal strategy gained from simulation experiments also optimal for real users?

How realistic is RL for dialogue?

- A general problem for all dialogue learning methods is the large state space (exponentially growing with number of features/dimensions)
- So far, RL methods are not straightforwardly applicable for the design of dialogue systems with realistic complexity.
- Methods to get around the complexity trap:
  - State space reduction (e.g., linear function approximation)
  - Hierarchical Learning
  - „Sub-Stragtegy Learning": Handcode most of the policy, and leave only special difficult design decisions for automatic optimisation