Language Technology II: Dialogue Design, Evaluation, Learning

Summer 2010

Manfred Pinkal

Determining Dialogue Policies

How do we find the optimal dialogue policy?

- Set alternative parameters by hand; examples:

  Minimum confidence value, in dependence of the importance of the decision

  Maximum number of items for which graphical display is appropriate (in dependence of actual user situation)
- Run full implemented system or WoZ experiment with human users, evaluate, modify or refine.

Comment:

- This is the way how things are done in real world, but:
- with high development costs and limited success
- Adaptive dialogue behaviour requires the dynamic combination of a larger number of features.
- Is machine learning an alternative?

Dialogue Policy

- Global design decisions include the specification of
  - a set of dialogue states S, the State Space (a set of nodes in a FSA, or a set of structured information states)
  - a set of possible system actions, the Actions Set (transitions, ISU operations)
  - a range of admissible possible actions A for each state s
- A dialogue policy is a decision procedure that selects specific actions $a \in A$ for possible dialogue states $s \in S$, more technically: a function
  - $\pi: S \rightarrow A$
- Examples for alternatives to be decided by a dialogue policy:
  - Grounding: Explicit grounding act/ implicit grounding act/ no grounding
  - Selection of presentation mode and modality for alternative user options.
- The objective of dialogue design:
  - Find a policy π which is optimal with respect to the purposes of the dialogue.

Supervised Dialogue Learning

Supervised learning of dialogue policies:

- Collect data from WoZ experiments, which are set up in an alternative way:
- Work with several wizards. Don't impose a specific strategy on the wizards' behavior, but just give them a general instruction ("Help the user reach its goal!")
- Derive a n-gram based statistical model from the data that proposes the most probable system move as the appropriate system reaction.

Comment:

- Learnt dialogue behaviour is locally consistent, global optimisation is not supported (due to n-gram constraint).
- System reproduces the average wizards' behavior, it cannot exclude bad decisions, or include new, unseen decisions.

Reinforcement Learning

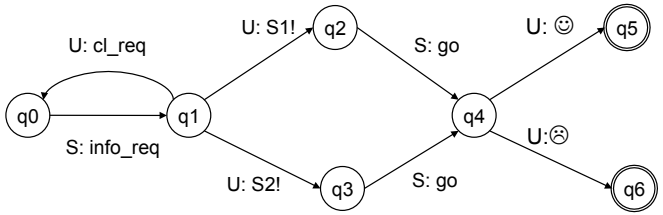
- System learns optimal dialogue policy by executing dialogues and getting feedback on its performance: Reinforcement Learning.
- Reinforcement learning builds on the concept of Markov Decision Process, a framework for modeling decision-making by an agent in an environment whose behaviour is (partly) random.
- The agent selects an action based on the current state (plus reward information).
- The environment emits information about the state it has adopted (and assigns a reward for the agent's last action).

Dialogue System as MDP

- Agent: The dialogue manager
- Environment: Everything that is not in direct control of the dialogue manager, including the user, the general situation (noise, distraction), the ASR system.
- States: Atomic states in simple finite-automaton dialogue models, structured states in ISU Framework.
- Actions: Dialogue moves.
- Reward: Will come later!

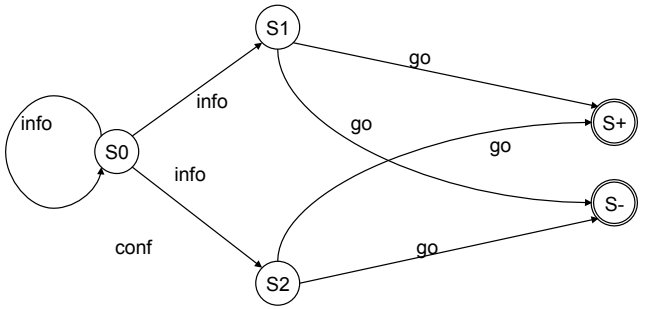
Elevator, Dialogue Model (1)

S: Where do you want to go?
U: Second floor, please.
S: <goes to 2nd floor>
U: Thank you, that's great!

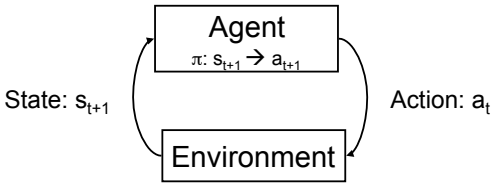


Notational Variant (MDP style)

Edge labels indicate actions of agent.
Nodes represent states of environment (observed by the agent).



Decision Process



- Agent interacts with stochastic environment:
  - At time t, agent picks an action a_t (on the basis of its current state s_t and strategy π), which (non-deterministically) influences the subsequent behaviour of the environment.
  - Environment assumes state s_{t+1}, agent updates state to s_{t+1}:
  - According to strategy π (hardwired in the automaton), agent (deterministically) selects and carries out an action a_{t+1}

    $\pi(s_{t+1}) = a_{t+1}$

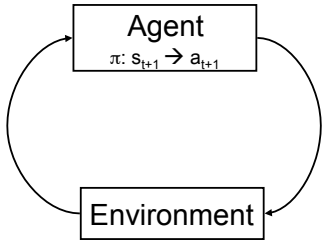
A more complex example

State s_t:

genre: rock
artist: false
genre_conf: false
artist_conf: false

State s_{t+1}:

genre: rock
artist: madonna
genre_conf: true
artist_conf: false



Action a_t:

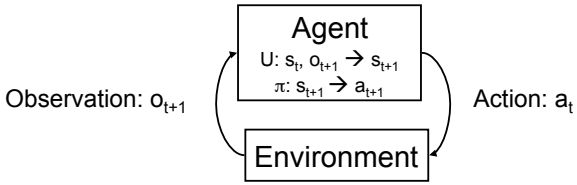
Implicit_conf genre: rock
request_info artist: ?

MP3-Player Scenario, ISU Framework
Example Rule in system's policy:

„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?

Partially observable states



- A more realistic picture:
  - The state of the environment (user) is hidden, not directly observable.
  - Environment emits signals (speech signal). On the basis of its observation o_t of the environment's behaviour, agent updates its hypothesis of the new state of the environment to s_{t+1}.

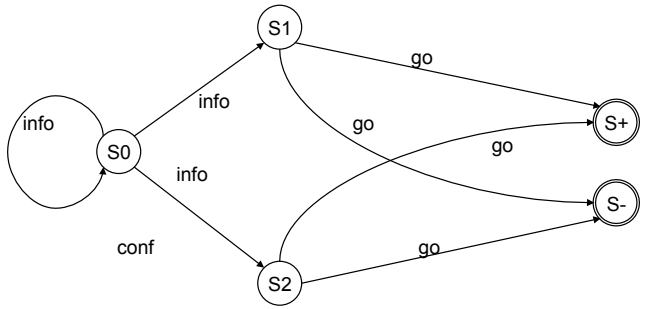
    Update function: $U(s_t, o_{t+1}) = s_{t+1}$
  - As before: According to strategy π, agent selects and carries out an action a_{t+1}

    $\pi(s_{t+1}) = a_{t+1}$

Notational Variant (MDP style)

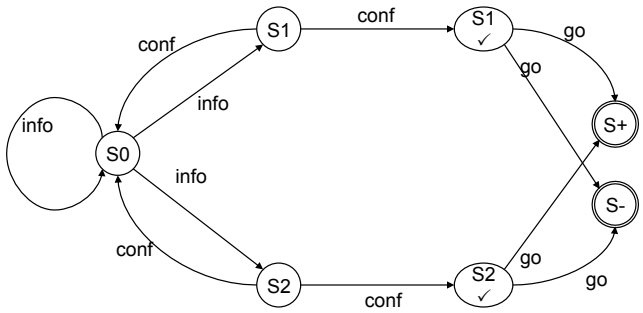
Edge labels indicate actions of agent.
Nodes represent states of environment (observed by the agent).

No grounding: Hard-wired "optimistic " policy



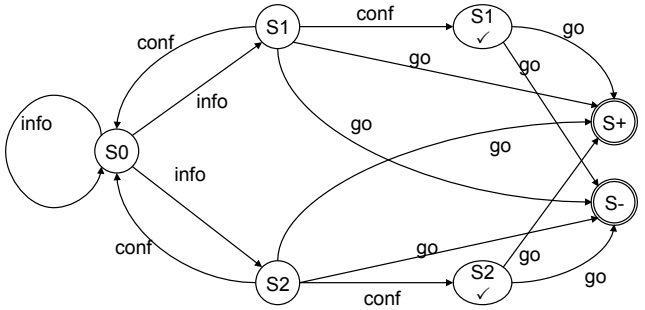
Elevator, Dialogue Model (2)

Obligatory confirmation request: Hard-wired cautious policy



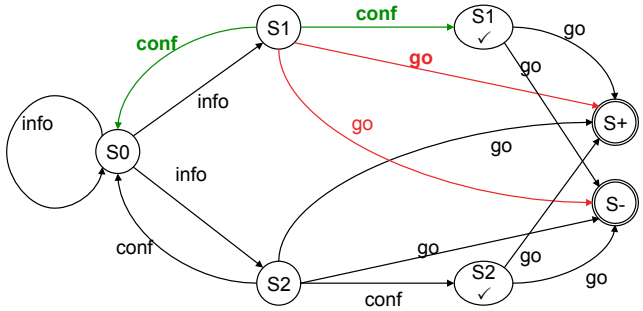
Elevator, Dialogue Model (3)

Combination of Models (1) and (2):



An example: Elevator dialogue

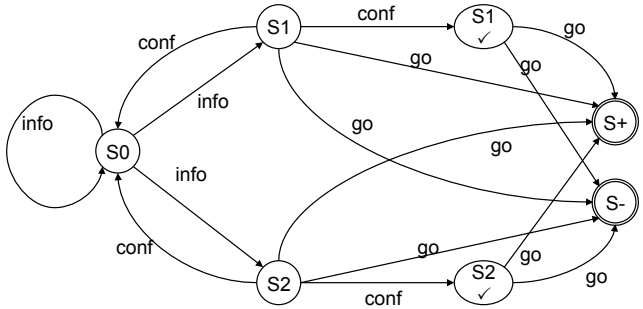
Combination of Models (1) and (2):
π(S1) is not uniquely defined.
Flexible/ underspecified grounding policy.



MDP: Completing the Picture

- Dialogue policy is not given by Model (3) directly, but must be separately. determined.
- As a precondition, we have to assess the usefulness or utility of the alternative dialogue moves (Confirmation Request or Direct Execution).
- The full MDP framework extend our simple model 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.

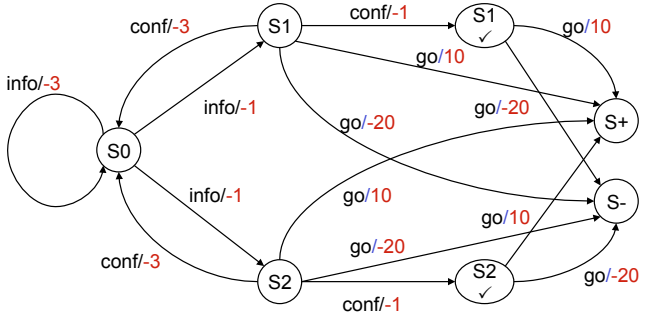
Elevator, Dialogue Model (3)



Adding Rewards

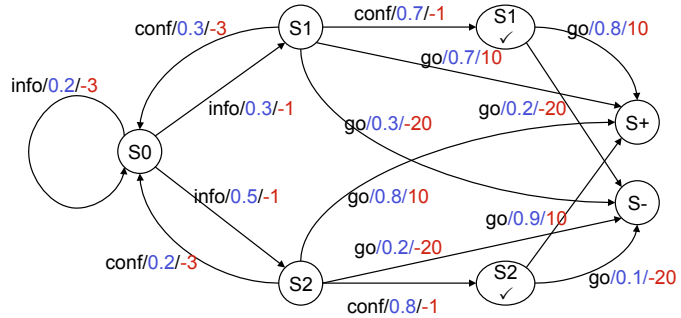
Example:

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



Adding Transition Probabilities

Transition probabilities between states of the environment through n-gram based model of user behaviour.



Questions Concerning Reward

- Immediate reward *r(s, a)* is locally assigned to state-action pairs.
- To decide between two actions, local comparison of immediate reward is insufficient. Ultimately, the utility of a dialogue move can only be assessed by the usefulness or the satisfaction of the user with the full dialogue.
- Option 1: Take the sum of immediate rewards obtained through dialogue execution: the Cumulative Reward.
- For a dialogue sequence $D = \langle s_0, a_1, s_1, \ldots, s_{n-1}, a_n, s_n \rangle$ the cumulative reward is

$$R(D) = \sum_{1 \le i \le n} r(s_i, a_i)$$

- Option 2a: Let dialogues be carried out/ assessed by human subjects.
- Option 2b: Approximate human usability assessment through automatically accessible features of dialogue. (→ PARADISE)

Expected Cumulative Reward

- Problem: We know the cumulative award or the final user satisfaction score only, after the dialogue is completed. Thus, the system has to compute or guess the cumulative reward by anticipating the future course of the dialogue.
- We refer to the Expected Cumulative Reward for state a and action s with

$$Q(s, a)$$

PARADISE: The Idea

No exam questions about paradise! (next 4 slides)

- PARADISE („Paradigm for Dialogue System Evaluation“) is an attempt to provide an objective, quantitative, operational basis for qualitative user assessments
- The foremost criterion for usability evaluation is user satisfaction
  - an intuitive criterion which can not be directly measured, but is only accessible through qualitative user judgments (obtained by user questionnaires, see above).
- But: User satisfaction is
  - correlated to task success (≈ effectiveness)
  - inversely correlated to the dialogue costs (≈ efficiency)
  - Task success and dialogue costs can be approximated by objective features that are available for automatic extraction from dialogue log-files.
- Reference: M. Walker/ D. Litman/C.Kamm/A.Abella: "PARADISE: A framework for evaluating spoken dialogue agents", Proc. of ACL 1997

The PARADISE questionnaire

- 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 information 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 in the dialogue?
- System Response: How often was the system sluggish and slow to reply to you?
- Expected Behaviour: Did the system work the way you expected it to?
- Comparable Interface: How did the system's vioce interface compare to other systems?
- Future Use: From your current experience with using the system, do you think you would use the system regularly?

PARADISE: Details

- Training data: A set of dialogues (including log-files) produced by interaction of a dialogue system A with different subjects.
- Assessment of user satisfaction through questionnaire
  - User satisfaction := the arithmetic mean of numeric values assigned to the nine questions of the questionnaire
- Task success information:
  - Either $\in \{0, 1\}$, Succeed or Fail
  - Or $\in [0, 1]$, e.g., the proportion of appropriately filled slots (for information-seeking/form-filling dialogue)
  - Or some measure for the agreement between actual and correct slot fillers
- Indicators for dialogue costs:
  - Efficiency measures: Elapsed time, # of System turns, # of user turns
  - Qualitative measures: # of timeout prompts, # of rejects, # of helps, # of cancels, # of barge-ins, mean ASR score
- Compute the best fitting function from task success and dialogue cost information to satisfaction value via linear regression.

The Performance Function

$$US = \left(\alpha * N(\kappa)\right) - \sum_{i=1}^{n} w_i * N(c_i)$$

- κ is task success, c_i are the cost factors. *N* is normalisation function ($N(\kappa)$ and $N(c_i)$ normalised task success and cost factors, respectively.
- α and w_i are weights on κ and the c_i, determined by linear regression.

Expected Cumulative Reward

- When selecting an action in a given state at time t, we do not know the outcome of the full interaction at t because the environment has non-deterministic behaviour: We do not know the result state of our action.
- To estimate the cumulative reward of an action, we have to take the alternative future developments of the interaction into account, and weight them by their probability.
- The expected cumulative reward of taking action *a* in state s is

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

Model-based RL

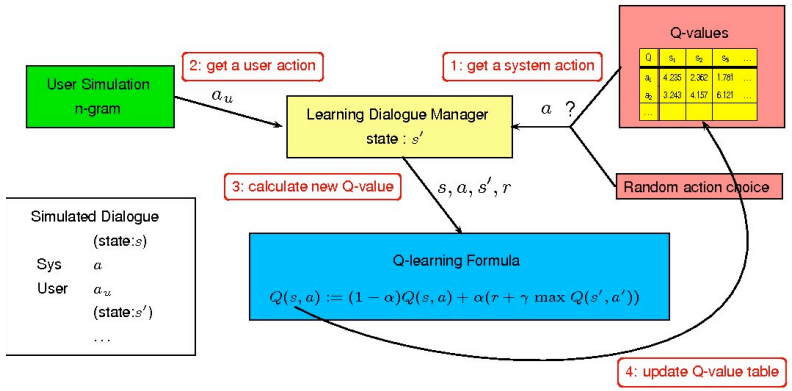
- Produce a dialogue corpus by WoZ experiments
- Learn n-gram state transition probabilities from the corpus
- Take evaluations from questionnaires or automatic user satisfaction assessments (PARADISE) as final rewards.
- Compute the optimal strategy via dynamic programming, taking all states and actions into account.

- Comments:
  - Realistic data through real dialogue loggings and user evaluation
  - The obtained policy is guaranteed to be optimal w.r.t. the data
  - 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 explored: No exploration of novel strategies, no guarantee that the optimal strategy occurs in the data.

Simulation-based RL

- Simulation-based methods:
  - Use the transition probabilities to drive a dialogue system which simulates a user.
  - Initiate the expected cumulative rewards of all state-action pairs by a default (e.g., 0) or randomly.
  - Approximate the optimal strategy by exploring different policies in interaction with the user, and updating the (preliminary) estimate of the Q-value.

The Basic Q-learning Cycle



The Q-Table

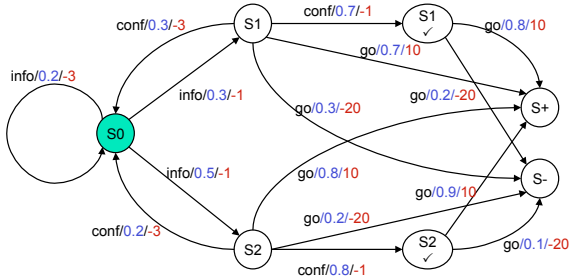
- We maintain a table of Q-values: expected rewards for state/action pairs.

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

- At the beginning, Q-values are either set to an arbitrary value (e.g., 0), or randomly initialised.
- For a given state s, the dialogue learning system either selects the action with the highest Q-value (highlighted in the above table), or chooses an action randomly.
- After each action, the Q-value estimate is updated using the current Q-value estimate of the resulting state.

Q-Learning Algorirhm: An example

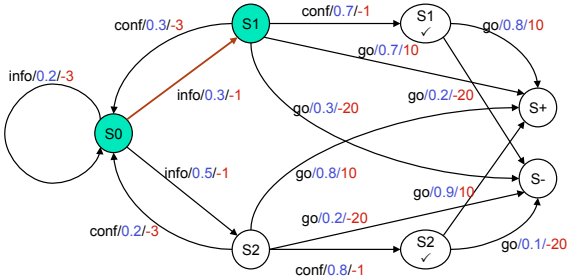
•Set s to S0
•Choose action from S0 (here, only 'info' is admissible)



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

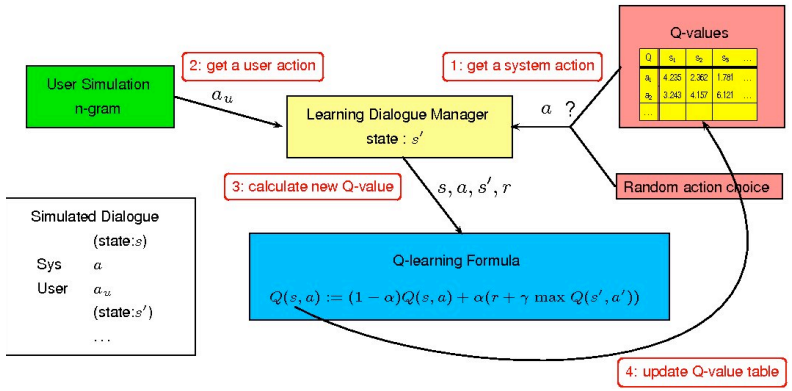
Example Cont'd

•Take the action, observe resulting state s' and reward r.
➢ In our example, 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

The Basic Q-learning Cycle



The *Q*-Learning Formula

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

- General scheme:

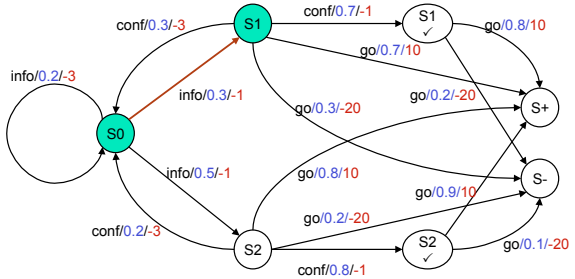
  *Q-New := (1-StepSize)*Q-Old + StepSize*(Target-Q-Old)*

  - 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)

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



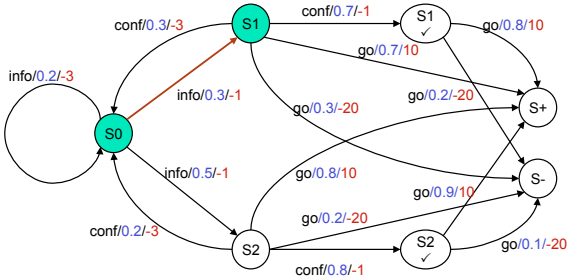
	S0	S1	S1✓
info	0.00	0.00	0.00
conf	0.00	0.00	0.00
go	0.00	0.00	0.00

Example Cont'd

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

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

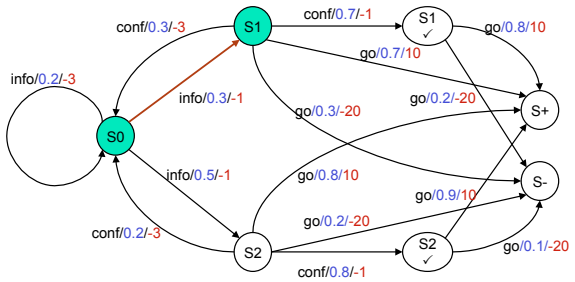
$$= 0 + 0{,}5 * \left(-1 + \max[0,0] - 0\right) = -0{,}5$$



	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



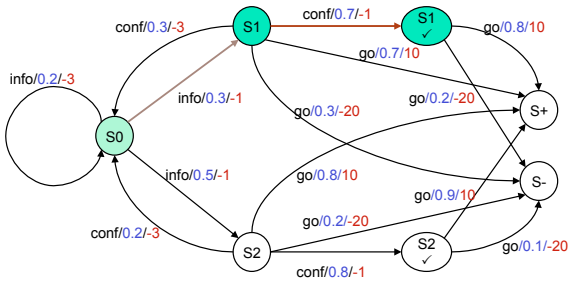
	S0	S1	S1✓
info	-0.5	0.00	0.00
conf	0.00	0.00	0.00
go	0.00	0.00	0.00

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).

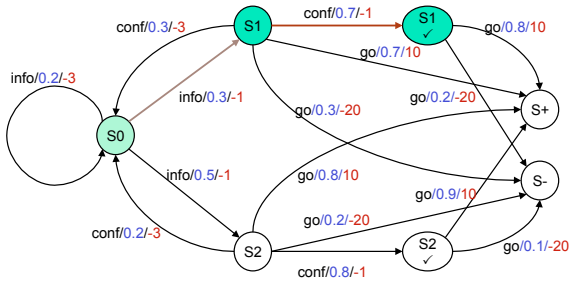


	S0	S1	S1✓
info	-0.5	0.00	0.00
conf	0.00	0.00	0.00
go	0.00	0.00	0.00

Example Cont'd

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

$$= 0 + 0,5 * (-1 + 0) = -0,5$$



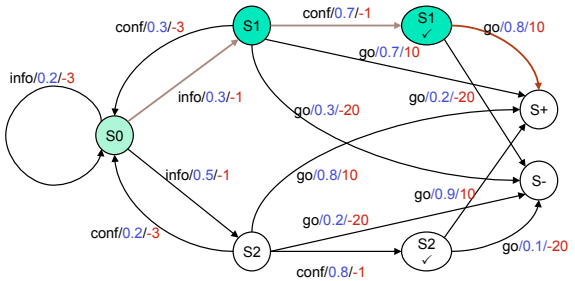
	S0	S1	S1✓
info	-0.5	0.00	0.00
conf	0.00	-0.5	0.00
go	0.00	0.00	0.00

Example Cont'd

•Choose action from S2: *go*
•Take action, observe resulting state s'(S+) and reward r (+10).
• Copmpute new Q-estimate and replace old one.

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

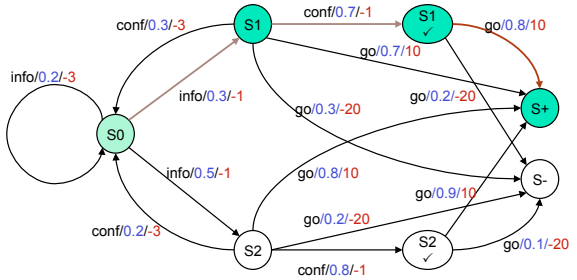
$$= 0 + 0,5 * (10 + 0) = 5$$



	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.

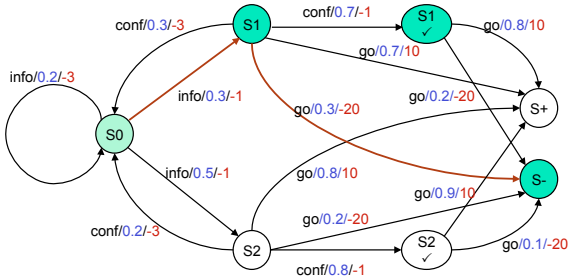


	S0	S1	S1✓
info	-0.75	0.00	0.00
conf	0.00	1.75	0.00
go	0.00	0.00	7.50

	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

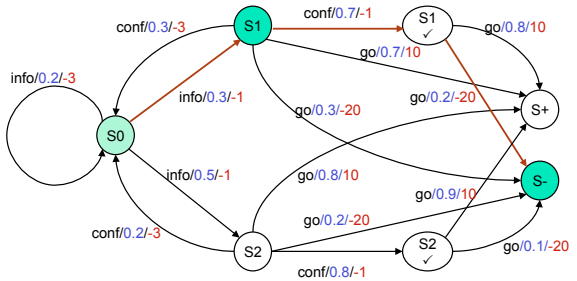
•Now, let the system try a direct, unconfirmed go:
the environments reaction reaction (ASR failure) may lead to a false destination trip of the elevator.



	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

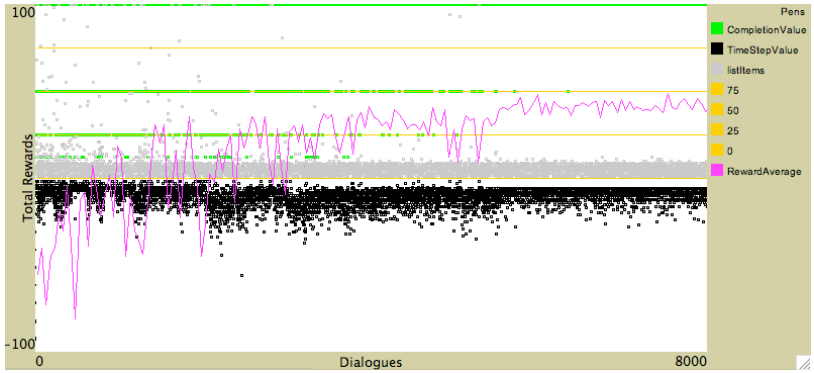
•But cautious version with confirmation may also fail ...



Simulation-based RL

- After a large number of iterations, the Q-table converges, and the optimal strategy can be read off.
- It is crucial that both action choice guided by the Q-table (greedy mode) and random action choice are combined.
- Typically, learning starts with an exploration phase (more random actions), and continues with an optimisation phase (selection by Q-table).

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?

- 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
  - „Sub-Stragtegy Learning": Handcode most of the policy, and leave only special difficult design decisions for automatic optimaisation