Language Technology II: Dialogue Learning

Summer 2009

Manfred Pinkal

Learning of Dialogue Strategies

- Can dialogue behaviour be learnt automatically?
- We cannot learn a complete dialogue model from the scratch.
  - What is a reasonable learning task?

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 $s \in S$.
  - $\pi: S \rightarrow A$
- Global design decisions leave dialogue strategy underspecified:
  - Perform explicit grounding act/ implicit grounding act/ no grounding
  - Select presentation mode and modality for alternative user options.
- The principal problem of dialogue design:
  - Find a policy π which is optimal with respect to the purposes of the dialogue.

Determining Dialogue Policies

Option 1: Manual tuning

- Setting thresholds / parameters manually (e.g.: minimum confidence value, maximum number of items to be graphically displayed). Problems:
  - Either: Non-adaptive, too little context-sensitivity.
  - Or: A highly complex design task which is difficult to control.

Option 2: Supervised learning

- Supervised learning on WoZ data: Learning average action decisions of wizards for a given state. Problems:
  - Pointwise decisions. - No optimisation on dialogue as a whole.
  - Mimicking of average wizard's behaviour: No evaluation of actual behaviour of the wizard, no exploration of new strategies.

Determining Dialogue Policies

- Option 3:
  System learns dialogue policies guided by feedback on its own dialogue performance.
- Challenges:
  - Automatic evaluation of dialogue performance (usability).
    → PARADISE
  - Automatic exploration of dialogue strategies: Generation of dialogues without a human dialogue partner.
    → USER SIMULATION
  - Assessment of the usefulness of an individual dialogue move by estimating its impact on the final outcome of the dialogue.
    → REINFORCEMENT LEARNING

Dialogue evaluation

- Technical evaluation
  - Typically component evaluation
    - ASR: Word-Error Rate
    - TTS: Intelligibility, Pleasantness, Naturalness
    - Grammar Coverage, etc.
- Usability evaluation
  - Typically end-to-end "black box" evaluation
  - Main criteria are:
    - Effectiveness (dialogue goals fully / partially accomplished?)
    - Efficiency (Number of turns? Dialogue duration?)
    - User satisfaction
- Customer evaluation
  - Leading principle is Return on Investment (ROI)
  - Includes: Costs, Platform compatibility, Maintenance properties

Usability Evaluation

- Mostly soft criteria:
  - "Usability Guidelines", best-practice rules, form the basis of expert evaluation or user questionnaires (cf. SASSI)
- Is it possible to evaluate usability in a objective, predictive, and general way?
- Can we measure user satisfaction?

PARADISE: The Idea

- 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).
- 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 can be automatically extracted 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

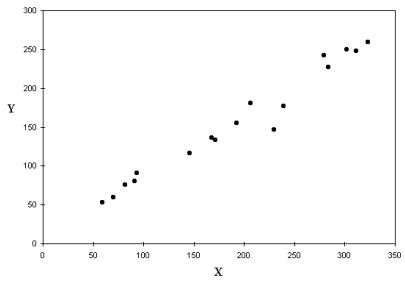
The PARADISE questionnaire

- 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?

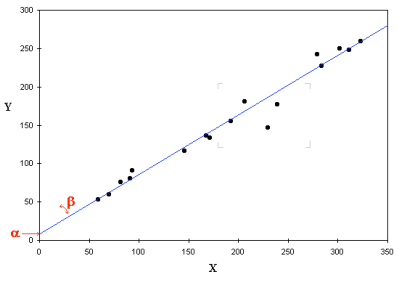
PARADISE: Details

- Basis: 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 ∈ {0, 1}, Succeed or Fail
  - Or ∈ [0, 1], 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.

Linear Regression



y = ax + b



y = 0.5x + 8

The Performance Function

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

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

Determining Dialogue Policies

Option 1:

- Setting thresholds / parameters manually (e.g.: minimum confidence value, maximum number of items to be graphically displayed). Problems:
  - Either: Non-adaptive, too little context-sensitivity.
  - Or: A highly complex design task which is difficult to control.

Option 2:

- Supervised learning on WoZ data: Learning average action decisions of wizards for a given state. Problems:
  - Pointwise decisions. - No optimisation on dialogue as a whole.
  - Mimicking of average wizard's behaviour: No evaluation of actual behaviour of the wizard, no exploration of new strategies.

Determining Dialogue Policies

Option 3:

- System learns dialogue policies guided by feedback on its own dialogue performance. Challenges:
- Automatic evaluation of dialogue performance (usability).
- Automatic exploration of dialogue strategies: Generation of dialogues without a human dialogue partner.

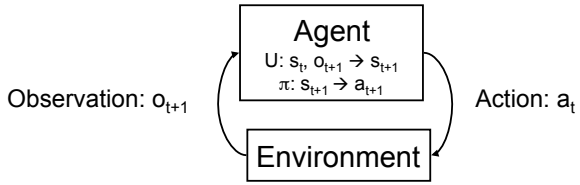
  (Everything else would be unrealistically inefficient and expensive)
- Assessment of the usefulness of
  - Measures are available only after the execution of the full dialogue. To evaluate the usefulness or utility of a specific dialogue move, the system must be able to anticipate its impact on the global outcome.
  - The system cannot determine the final outcome: It cannot predict the user's reaction to its action, and thus can only estimate the state resulting from it, which again is input state for its subsequent decision, etc.

  Solution through Reinforcement Learning, which is based on the concept of Markov Decision Process.

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

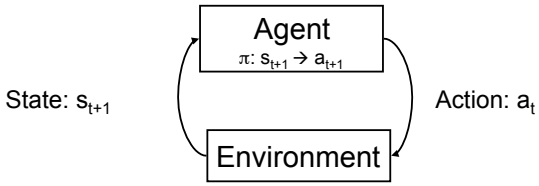


- The agent interacts with a stochastic environment:
  - At time t, it picks an action a_t (on the basis of its current state s_t and strategy π), which (non-deterministically) influences the subsequent behaviour of the environment.
  - On the basis of the observation o_t of the environment's behaviour/ reaction, it updates the state to s_{t+1}:

    $U(s_t, o_{t+1}) = s_{t+1}$
  - According to strategy π, it (deterministically) selects and carries out an action a_{t+1}

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

Simple model – further simplified



- We asume that the environment is directly observable by the agent:
- This implies that the state of the environment and the agent's „context model" coincide.
- The update function is deterministic, which allows us to simplify the scheme (omitting o_t and assuming that the environment directly emits the resulting state s_{t+1}).

Dialogue as a decision process

- Agent: The dialogue manager
- Environment: Includes 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, in more elaborate models, e.g. ISU, defined by the current values 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 confidence 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"

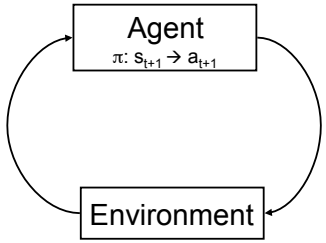
An Example

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



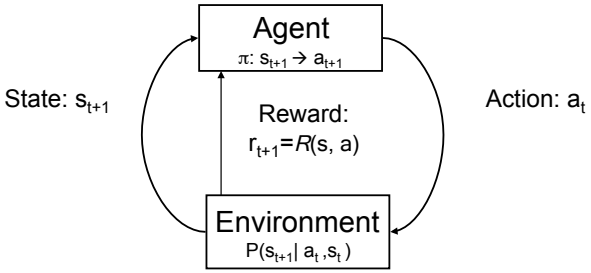
Action a_t:

Implicit_conf genre: rock
request_info artist: ?

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.

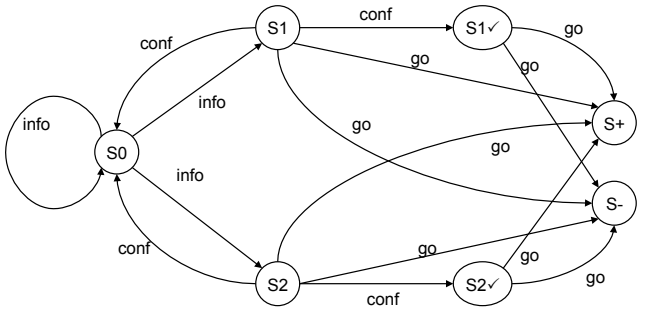
Markov Decision Processes (MDP)



Extension of a simple decision process by:

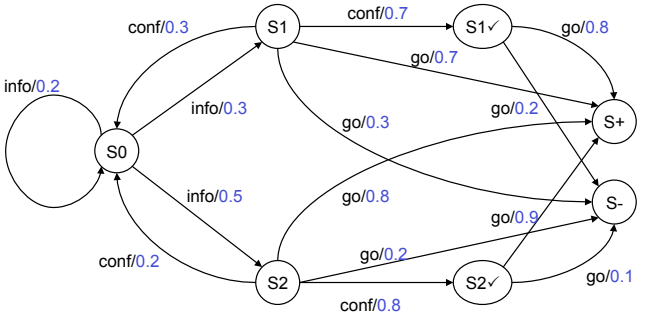
- State transition function *T*: S×A×S→[0,1]
  - *T*(s, a, s') = *P*(s'|a, s), the probability that action a causes the environment to change from s to s'.
- Reward function *R*:
  - *R*(s, a) is a real number, specifying the expected advantage of reaching state s through action a

An example: Elevator dialogue

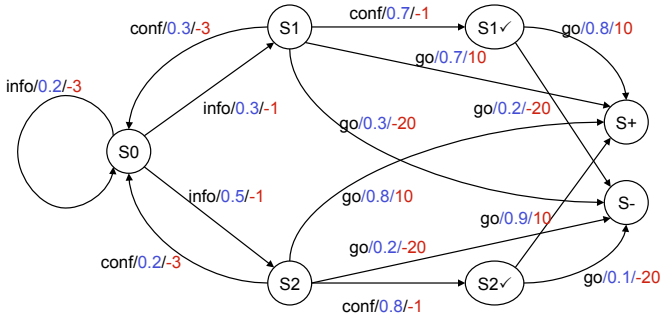


- Simple Decision Process: The environment's behaviour can be modelled as a non-deterministic finite automaton.
- Agent's actions are input symbols. Transitions give possible (non-deterministic) environments reactions.
- On the last slide you see a model of a simple elevator dialogue:
  - Two floors
  - Three available types of dialogue acts, 5 specific acts:
    - Information Request: „Which floor do you want to go?" (info)
    - Comfirmation: „Is xth flor correct?" (conf)
    - Command to elevator driver (go)
  - 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.

Reward

- The immediate reward r(s, a) is given by the immediate reward function for every state and action, it is standing for the action's "intrinsic desirability".
- The reward at time t of an interaction:
  - $R_t(<s_t, a_t, s_{t+1}, a_{t+1}, s_{t+2}, a_{t+2}, \ldots>)$
    $= r(s_t, a_t) + \gamma\, r(s_{t+1}, a_{t+1}) + \gamma^2\, r(s_{t+2}, a_{t+2}) + \ldots$
  - γ is discount factor between 0 and 1, intended to model the fact that immediate rewards are potentially more important for a decision than those received in a more distant future.
  - γ=0: cumulative reward is immediate reward
  - Reinforcement learning for dialogue typically sets the discount factor γ to 1 (or close to one).

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 behaves non-deterministic: We do not know the result state of our action.
- To estimate the cumulative reward, 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 a given state *s* and following π thereafter is written as $Q^{\pi}(s,a)$
- It can be defined as follows (a simplified version of one of the "Bellman equations"):

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

Obtaining the Optimal Strategy

- The optimal strategy π^* is the one which maximises the expected cumulative reward.
- How can we find it?
- Two alternative options:
  - Model-based methods: If we have a model of the environment, i.e., we know transition probabilities and rewards, we can compute the optimal strategy offline.

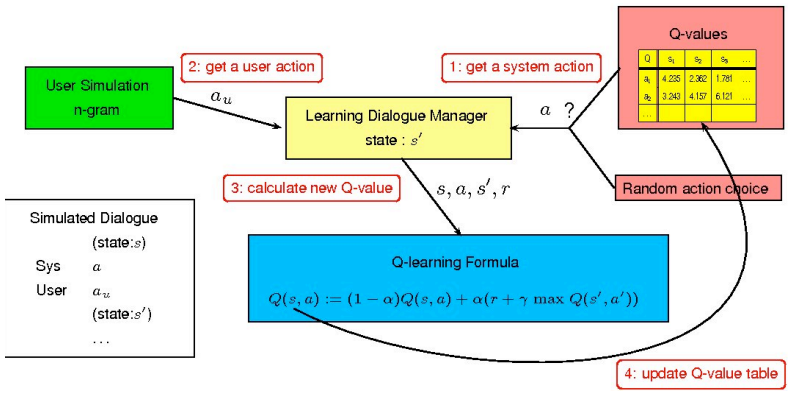
Model-based RL

- Produce a dialogue corpus by WoZ experiments
- Learn 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.
- Advantages:
  - Realistic data through real dialogue loggings 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 explored: No exploration of novel strategies, no guarantee that the optimal strategy occurs in the data.

Obtaining the Optimal Strategy

- The optimal strategy π^* is the one which maximises the expected cumulative reward.
- How can we find it?
- Two alternative options:
  - Model-based methods: If we know transition probabilities and rewards, we can compute the optimal strategy offline
  - Simulation-based methods: Gradual approximation of the optimal strategy by exploring different policies in interaction with a simulated user, and updating the (preliminary) estimate of cumulative reward (the Q-value).

The Basic Q-learning Cycle



User Simulation

- Two options for the simulation of the user / the environment behavior:
  - An implemented dialogue system
  - An n-gram based statistical model, learnt from WoZ data.

The Q-Table

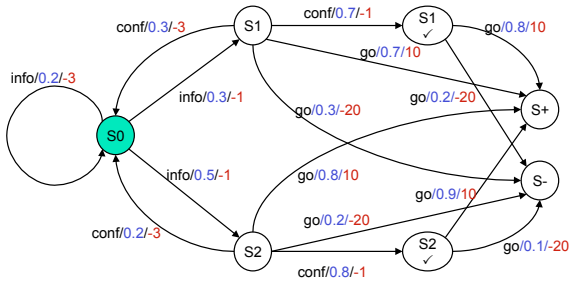
- We maintain a table of Q-values: expected rewards for state/action pairs.

	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

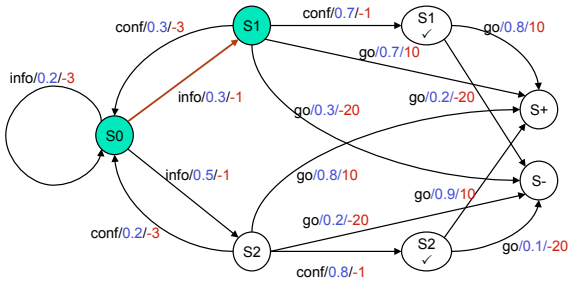
•Set s to S0
•Choose action from S0 (here, only 'info' is admissible)



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

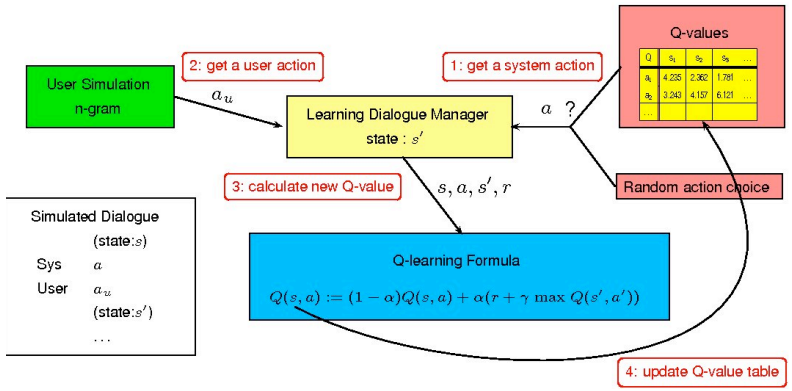
Example Cont'd

•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

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

- General scheme:

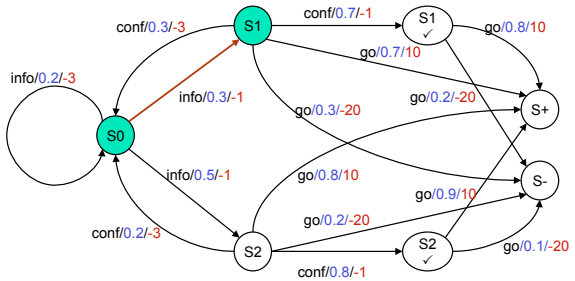
  *Q-New := (1-StepSize)*Q-Old + StepSize*(Target-Q-Old)*

  - 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)

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



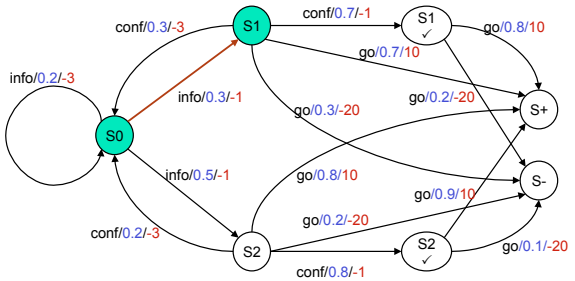
	S0	S1	S1✓
info	0.00	0.00	0.00
conf	0.00	0.00	0.00
go	0.00	0.00	0.00

Example Cont'd

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

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

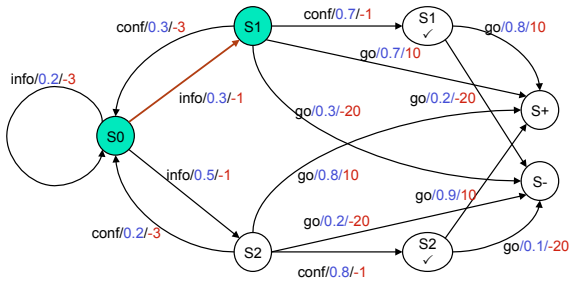
$$= 0 + 0{,}5 * \left(-1 + \max[0,0] - 0\right) = -0{,}5$$



	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



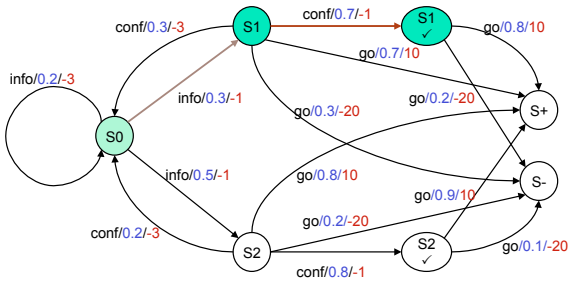
	S0	S1	S1✓
info	-0.5	0.00	0.00
conf	0.00	0.00	0.00
go	0.00	0.00	0.00

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).

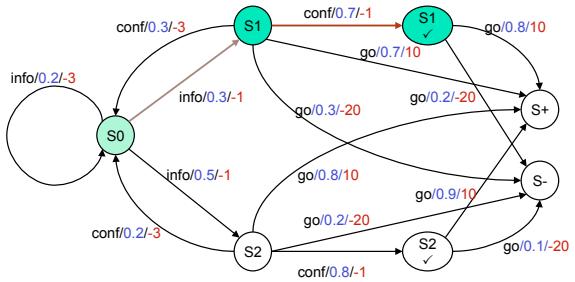


	S0	S1	S1✓
info	-0.5	0.00	0.00
conf	0.00	0.00	0.00
go	0.00	0.00	0.00

Example Cont'd

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

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



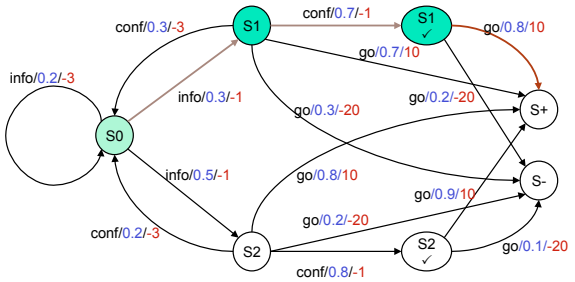
	S0	S1	S1✓
info	-0.5	0.00	0.00
conf	0.00	-0.5	0.00
go	0.00	0.00	0.00

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.

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

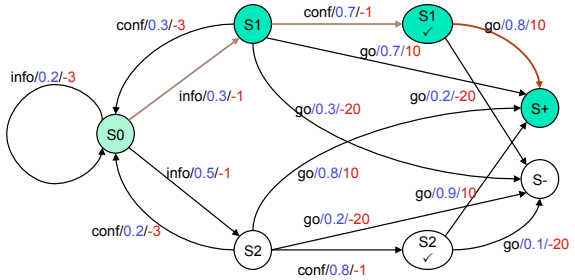
$$= 0 + 0,5 * (10 + 0) = 5$$



	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.

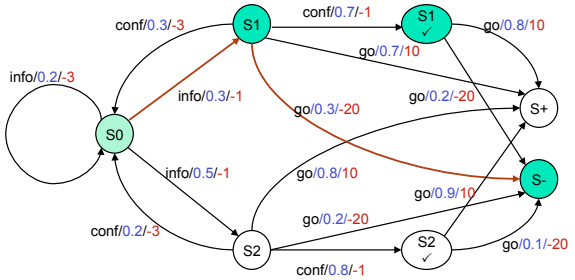


	S0	S1	S1✓
info	-0.75	0.00	0.00
conf	0.00	1.75	0.00
go	0.00	0.00	7.50

	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

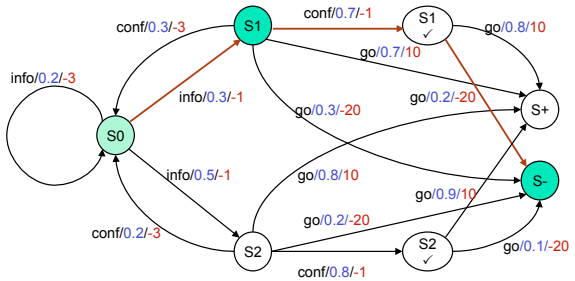
•Now, let the system try a direct, unconfirmed go:
the environments reaction reaction (ASR failure) may lead to a false destination trip of the elevator.



	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

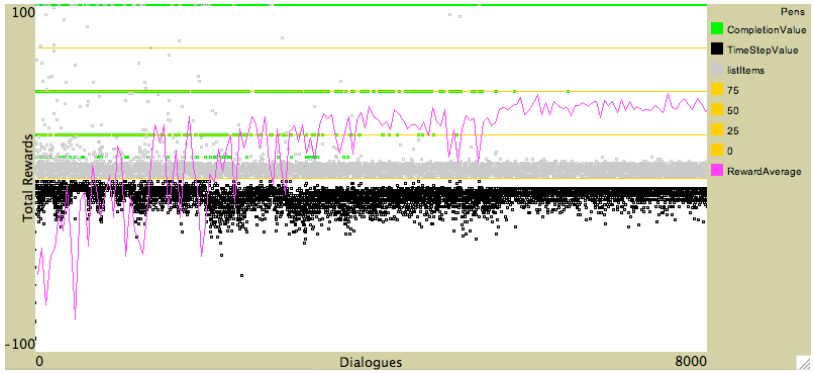
•But cautious version with confirmation may also fail ...



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