Chapter 4
Proof-of-Concept: Information Seeking Strategies

Reinforcement Learning for Adaptive Dialogue Systems

Verena Rieser, Oliver Lemon

JAEHO SHIN

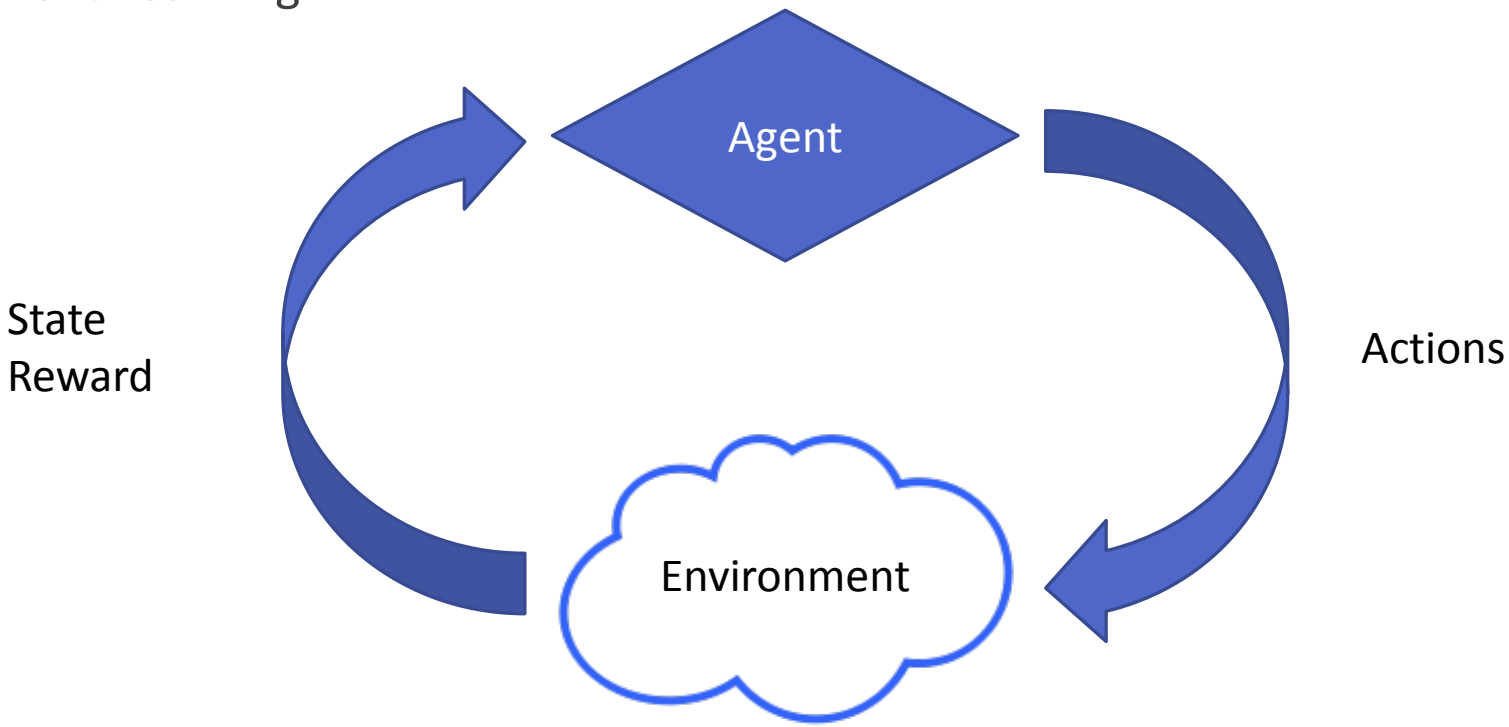
contents

What do we know so far?

Reinforcement Learning

What do we know so far?

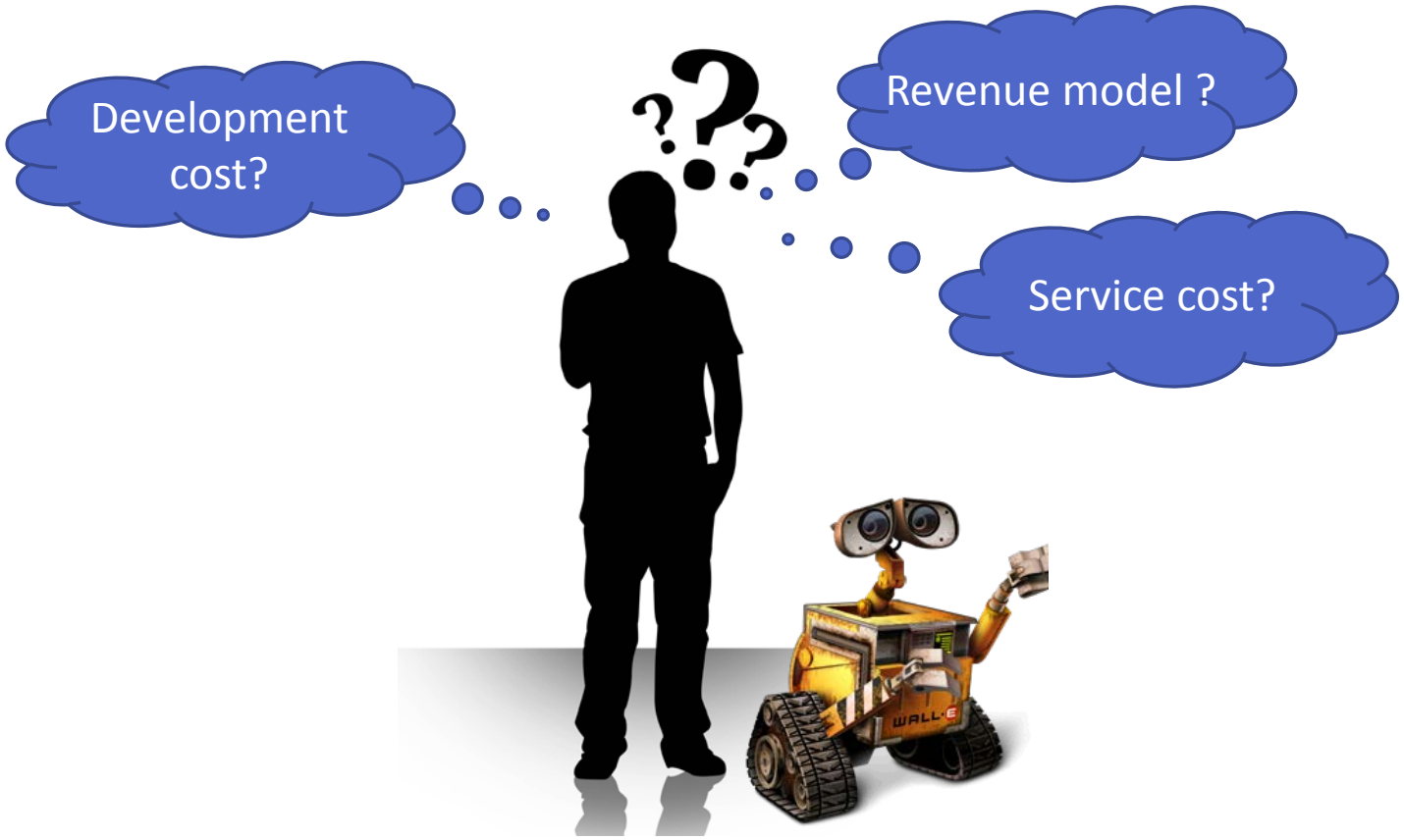
Reinforcement Learning



Goal: Maximize the expected rewards

What Is Proof-of-Concept?

Check **feasibility** of certain concepts or theories



Introduction

Prove reinforcement learning works better than hand-coded strategies

What about previous works?

- Wrong evaluation
  - hand-coded strategies are not tuned to the same reward function

For proof-of-concept

- Learn the decision of when to end the ‘information acquisition’ phase
- Commence with ’information presentation’

Example

Hello, how may I help you?

"I want an IU song"

Ok, an IU song. From what album? (db: 77)

"From Modern Times"

OK, from Modern Times. What song title?(db: 13)

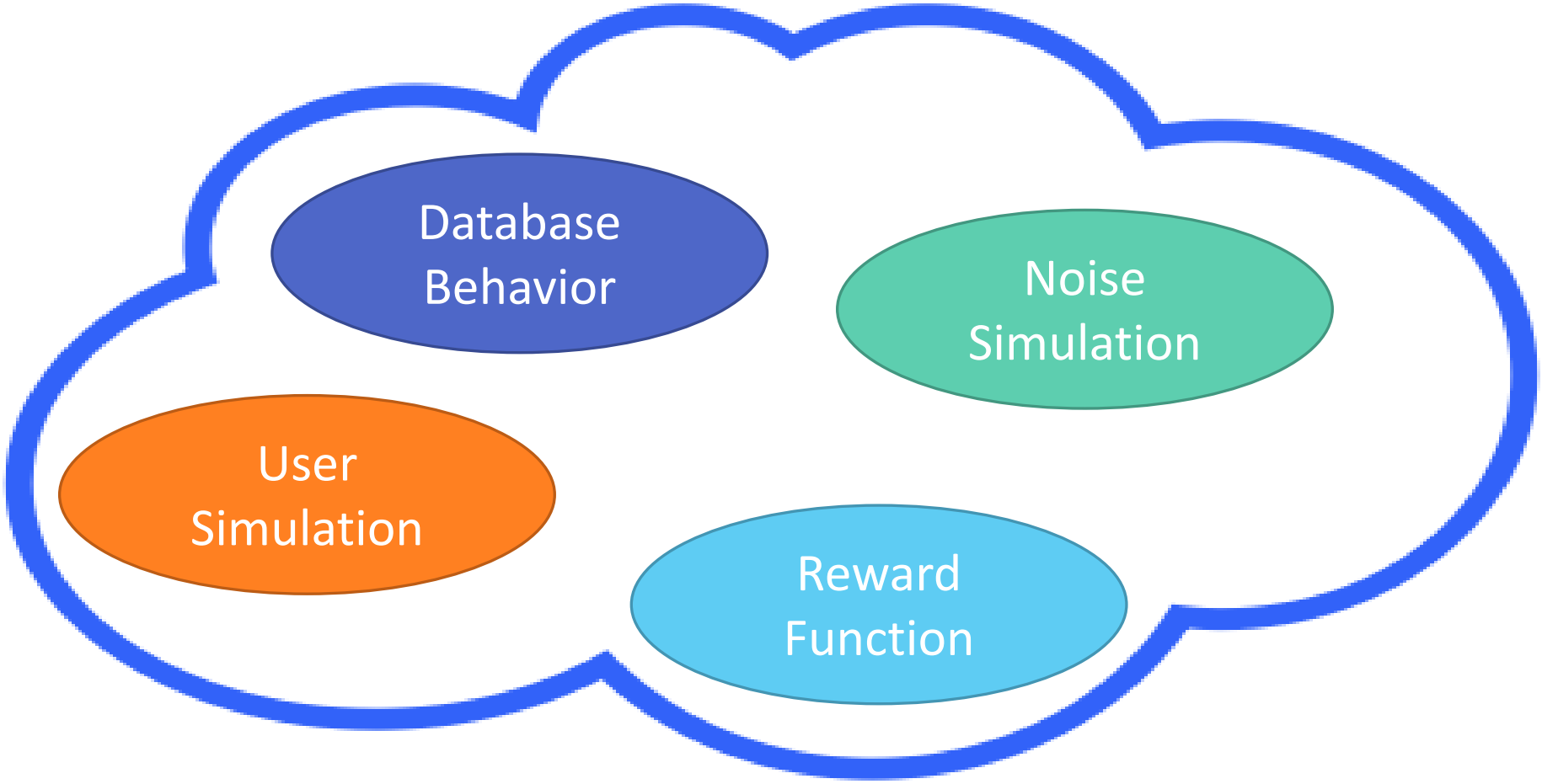
"Obliviate"

Did you say obese? (db: 1)

"No"

Ok, there are 13 items you might be looking for, which are now shown on the screen. (db: 13)

Simulated Learning Environment



Problem Representation

Greet

(e.g. What can I help you with?)

Ask A Slot

(e.g. What kind of music would you like?)

Explicit Confirm

(e.g. Did you say Jazz?)

Implicit Confirm

(e.g. Okay, Jazz music. Which artist?)

Close & Present Information

(e.g. The following items match your query…)

Database Retrieval Simulations

MONOTONIC DATABASE SIMULATION

- Well-behaved search tasks
- Fill a search slot -> fewer # of results
- Boolean AND
- For question A -> answer B

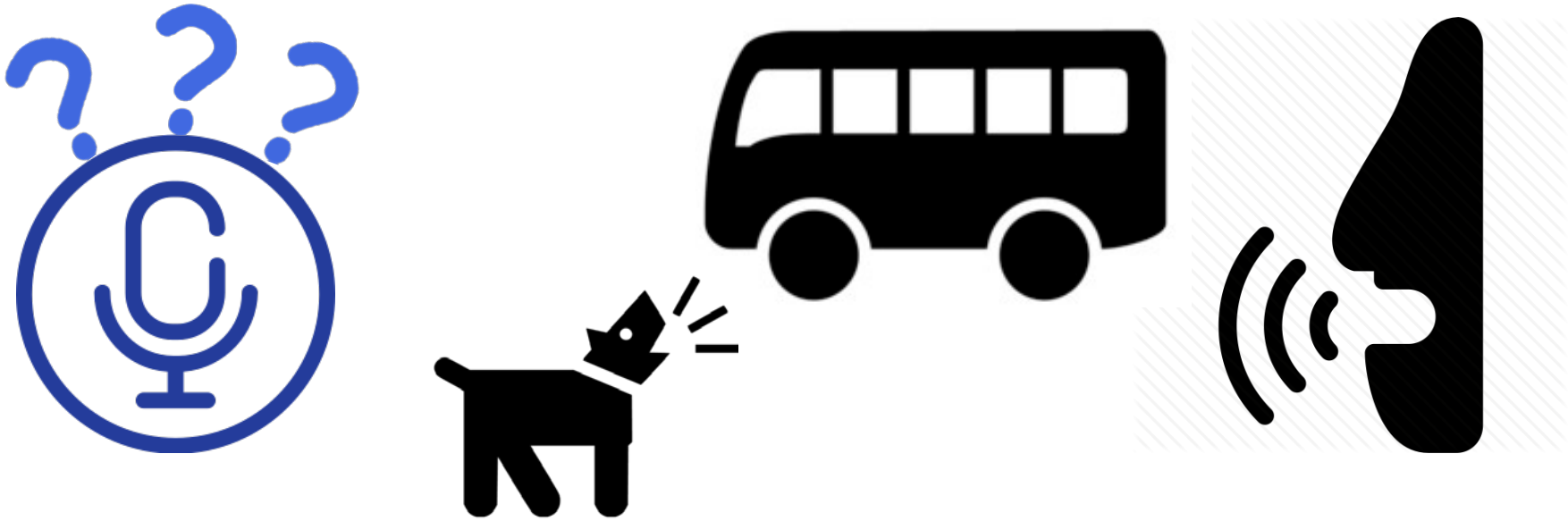
RANDOM DATABASE SIMULATION

- Fill a search slot -> # of results increase/decrease
- Boolean AND or OR
- For question A -> answer B or C or D…

The Size of Database = 100

Noise Model

Simulates the effect of channel noise on the interaction.



High Noise(HN): 50% chance of filled slot being correct

Low Noise(LN): 80% chance of filled slots being correct

User Simulations

Employed bi-gram model

- Bi-gram?
  - a sequence of two adjacent elements
- Estimate possible user acts conditioned on the previous system action

User Simulations

yes-answer

no-answer

provide-asked

(e.g. Q: “What band do you want?” A: “I want ABBA”)

provide-other

(e.g. Q: “What type of music do you want?” A: “I want ABBA”)

re-provide-asked

(e.g. Q: “Did you say Rock music?” A: “No, I want Pop”)

provide-two-slots

(e.g. “I want an ABBA song front the album Waterloo”)

User Simulations

sysAct / userAct	yes	Provide-other	Provide-asked	Re-provide-asked	no	Provide-two	Silent
greet	-	80.0	-	-	-	16.0	4.0
askASlot	-	20.0	70.0	-	-	6.0	4.0
explCont(HN)	40.0	4.0	-	8.0	40.0	4.0	4.0
explConf(LN)	50.0	4.0	-	8.0	10.0	4.0	4.0
implConf(HN)	2.0	2.0	40.0	30.0	18.0	4.0	4.0
implConf(LN)	4.0	4.0	60.0	3.0	15.0	10.0	4.0

Reward Function

$$FinalReward = completionValue - dialogueLengthPenalty - DBhitsPenalty$$

dialogueLengthPenalty: penalize every system turn

DBhitsPenalty: penalize every item presented to user (*itemPenalty IP* per presented item)

completionValue: the percentage probability that the user goal is in the presented result set

$$completionValue = 100 \times (P_c)^C \times \left(P_f\right)^F$$

P_c: the probability of a confirmed slot being correct

P_f: the probability of a filled slot being correct

C: # of confirmed slot

F: # of filled slot (but not confirmed)

Objective and Reward Function

Ex)

4-slot problem where turnPenalty(TP) = 1, itemPenalty = 10, P_c = 1.0, and P_f = 0.5

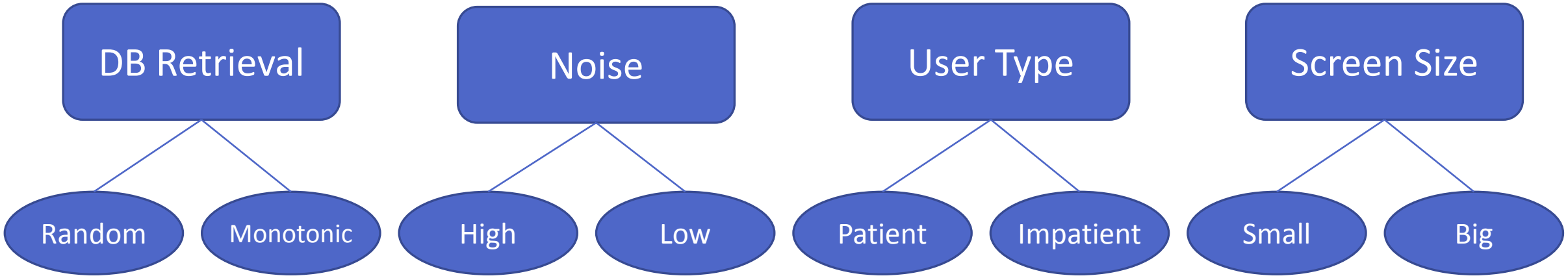
After 6 system turns, 2 items are presented to a user

4 slots are filled, but only 3 were confirmed

- $completionValue = 100 \times 1^3 \times 0.5^1 = 50$
- $dialogueLengthPenalty = 6 \times 1 = 6$
- $DBhitsPenalty = 10 \times 2 = 20$

$$FinalReward = 50 - 6 - 20 = 24$$

Application Scenarios



Threshold-based Baseline

Manually tune different thresholds to maximize the returns from reward functions

Manually set several combinations of different thresholds

Test performance with the simulated user

Repeat this until reaching to the maximum performance.

1. Greet the user
2. Either AskASlot or ImplConf-AskASlot
3. Then repeat 2 until their no slots left to fill
4. then ExplConf the remaining filled slots
5. Present the answer set to the user

Threshold-based Baseline

Present information if

- All slots are confirmed
- The number of items returned from DB < a threshold
- The dialogue length > a threshold

H_{ss}: short dialogue(≤ 6), short lists(≤ 7)

H_{ls}: long dialogue(≤ 10), short lists(≤ 7)

H_{sl}: short dialogue(≤ 6), long lists(≤ 14)

H_{ll}: long dialogue(≤ 10), long lists(≤ 14)

Threshold-based Baseline

Example) dialogue threshold: 7, DB information presentation threshold: 6

Hello, how may I help you?

"Do you have some modern classical music?"

Ok, modern classical. Which composer? (db: 77)

"Steve Reich"

Do you have a preferred piece by Steve Reich?(db: 25)

"Yes, drumming"

Did you say drumming? (db: 4)

"Yes"

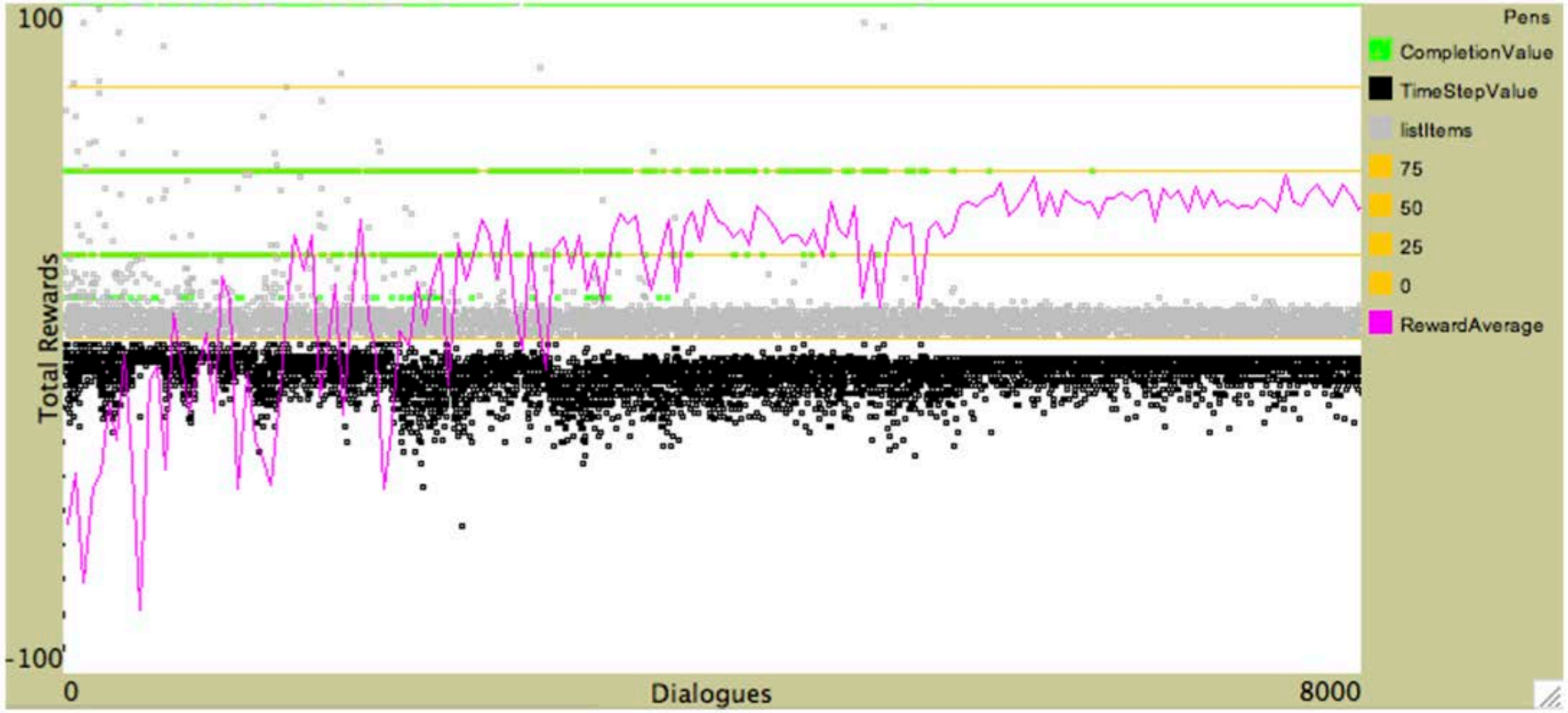
OK, there are 4 parts on this album, which are now shown on the screen. (db: 4)

Reinforcement Learning Method

Training parameters

- Number of cycles = $96{,}000$
- Learning rate $\alpha = 0.2$
  - The step-size parameter for learning
- Discount rate $\gamma = 0.95$
- Eligibility trace parameter $\lambda = 0.9$
  - The trace decay parameter
- Exploration halving time $\varepsilon = \frac{1}{6} \times number\ of\ cycles = 16{,}000$

Reinforcement Learning Method



Results

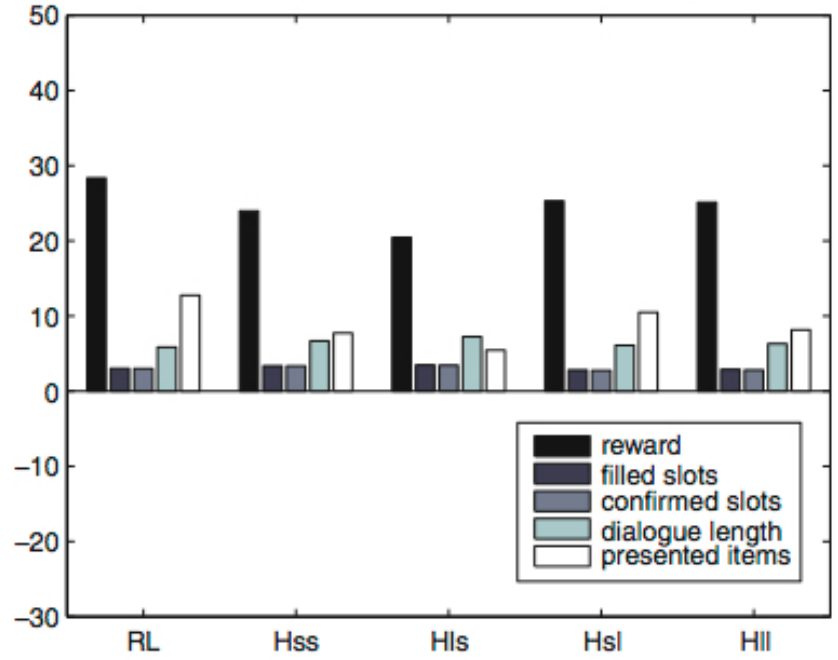
Run 550 test dialogues in simulation

Reward def.	best hand-coded	RL
Monotonic DB		
HiNoise, HiHit, HiTurn	-31.1±54.7	28.1±7.8***
HiNoise, LowHit, HiTurn	25.3±30.2	28.4±11.1**
HiNoise, HiHit, LowTurn	34.9±30.6	41.3±21.3***
HiNoise, LowHit, LowTurn	85.7±11.2	86.0±4.6
LowNoise, HiHit, HiTurn	85.6±5.9	87.2±4.5**
LowNoise, HiHit, LowTurn	38.1±28.7	41.8±45.9
LowNoise, LowHit, HiTurn	31.8±14.4	31.3±10.1
LowNoise, LowHit, LowTurn	-22.6±54.3	-20.2±22.5
Random DB		
HiNoise, HiHit, HiTurn	-373.3±321.4	-159.7±96.3***
HiNoise, LowHit, HiTurn	-275.1±312.0	-115.7±164.9***
HiNoise, HiHit, LowTurn	-1.3±38.8	13.8±25.5***
HiNoise, LowHit, LowTurn	55.2±30.8	62.6±21.6***
LowNoise, HiHit, HiTurn	55.1±32.3	80.5±9.3***
LowNoise, HiHit, LowTurn	-290.2±310.7	-155.3±217.5***
LowNoise, LowHit, HiTurn	1.9±39.5	20.4±23.2***
LowNoise, LowHit, LowTurn	-333.6±316.5	-166.1±206.9***

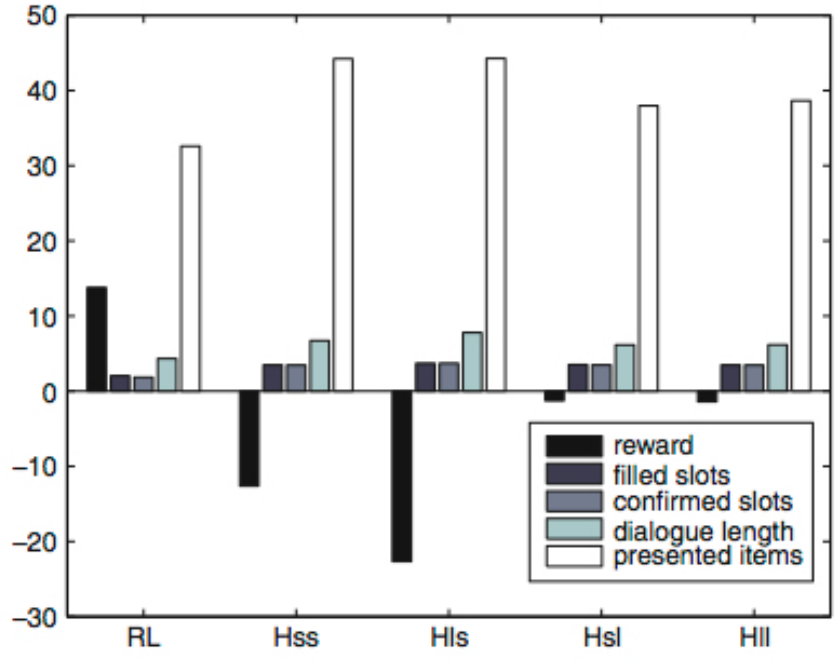
Table 4.2 Direct comparison between mean rewards (with standard deviation ±) obtained by the RL-based and best performing hand-coded policy *for each environment* with results from a paired t-test; ** denotes $p < .005$, *** $p < .001$

Results

MONOTONIC DB, HIGH NOISE, LOW HIT, HIGH TURN

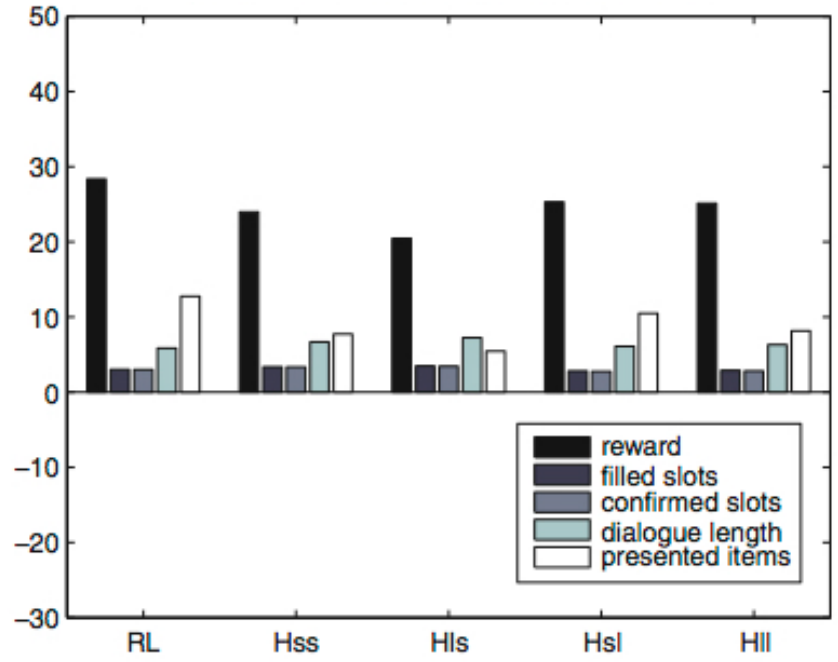


RANDOM DB, HIGH NOISE, LOW HIT, HIGH TURN

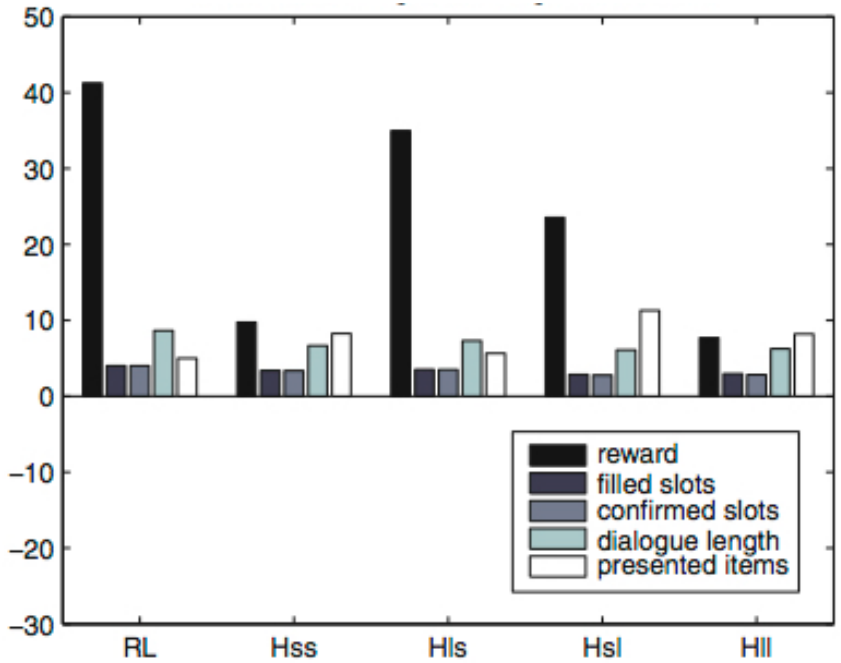


Results

MONOTONIC DB, HIGH NOISE, LOW HIT, HIGH TURN

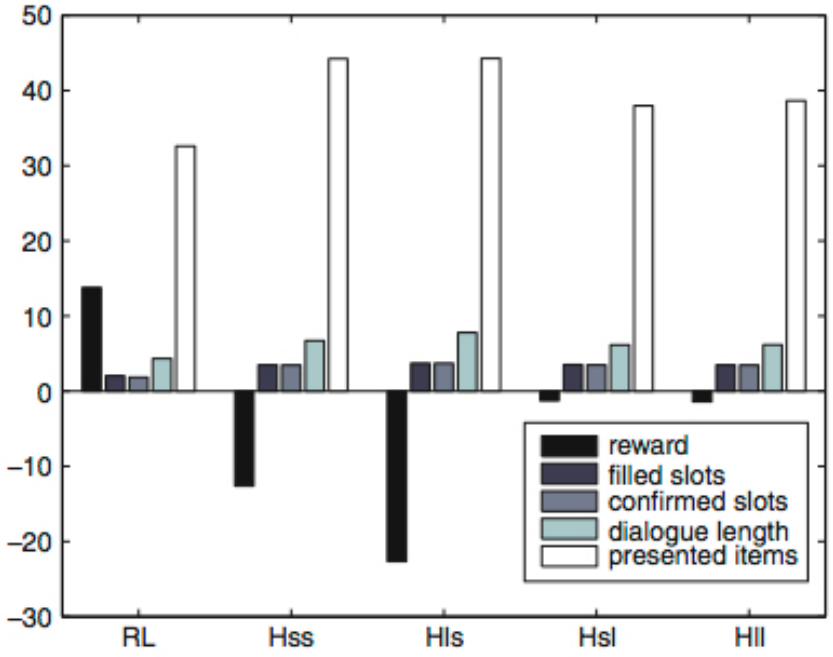


MONOTONIC DB, HIGH NOISE, HIGH HIT, LOW TURN

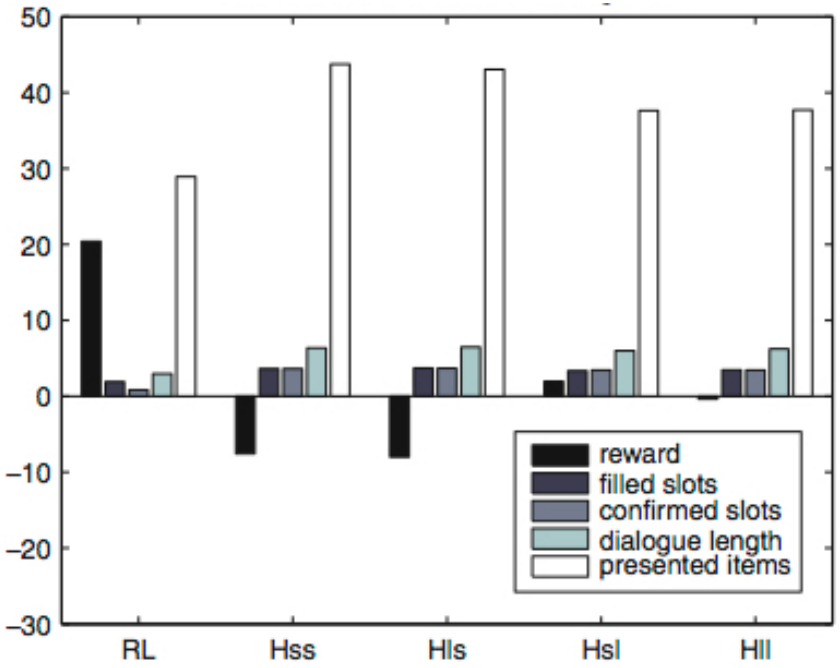


Results

RANDOM DB, HIGH NOISE, LOW HIT, HIGH TURN



RANDOM DB, LOW NOISE, LOW HIT, HIGH TURN



Summary

Provide the general proof-of-concept

- RL-based strategies significantly outperform manually tuned heuristics

How to use a statistical policy learning framework

- What to ask
- How many results to present
- Etc.

In 93% of the cases, the learned policies are better than the hand-coded ones.