

NPFL099 Statistical Dialogue Systems

6. Dialogue Management (1)

mostly Dialogue State Tracking

<http://ufal.cz/npfl099>

Ondřej Dušek

3. 11. 2025



Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics



unless otherwise stated

Dialogue Management & State

- Dialogue management consists of:
 - **State update** ← we need to **track dialogue state** over time
 - Action selection (discussed later)
- **Dialogue state** needed to remember what was said in the past
 - tracking the dialogue progress
 - summary of the whole dialogue history
 - basis for action selection decisions

U: I'm looking for a restaurant in the city centre.

S: OK, what kind of food do you like?

U: Chinese.

✗ S: What part of town do you have in mind?

✗ S: Sure, the Golden Dragon is a good Chinese restaurant. It is located in the west part of town.

✓ S: Sure, the Golden Dragon is a good Chinese restaurant. It is located in the city centre.

Dialogue State Contents

- “All that is used when the system decides what to say next” (Henderson, 2015)
- **User goal**/preferences ~ NLU output
 - slots & values provided (search constraints)
 - information requested
- Past **system actions**
 - information provided
 - slots and values
 - list of venues offered
 - slots confirmed
 - slots requested
- **Other** semantic context
 - user/system utterance: bye, thank you, repeat, restart etc.

U: Give me the address of the first one you talked about.

U: Is there any other place in this area?

S: OK, Chinese food. [...]

S: What time would you like to leave?

Problems with Dialogue State

- NLU is unreliable

- takes unreliable ASR output
- makes mistakes by itself – some utterances are ambiguous
- output might conflict with ontology

ASR: 0.5 I'm looking for an expensive hotel
0.5 I'm looking for inexpensive hotels

- Possible solutions:

- detect contradictions, ask for confirmation
- ignore low-confidence NLU input
 - what's "low"?
 - what if we ignore 10x the same thing?

NLU: 0.3 inform(type=restaurant, stars=5)

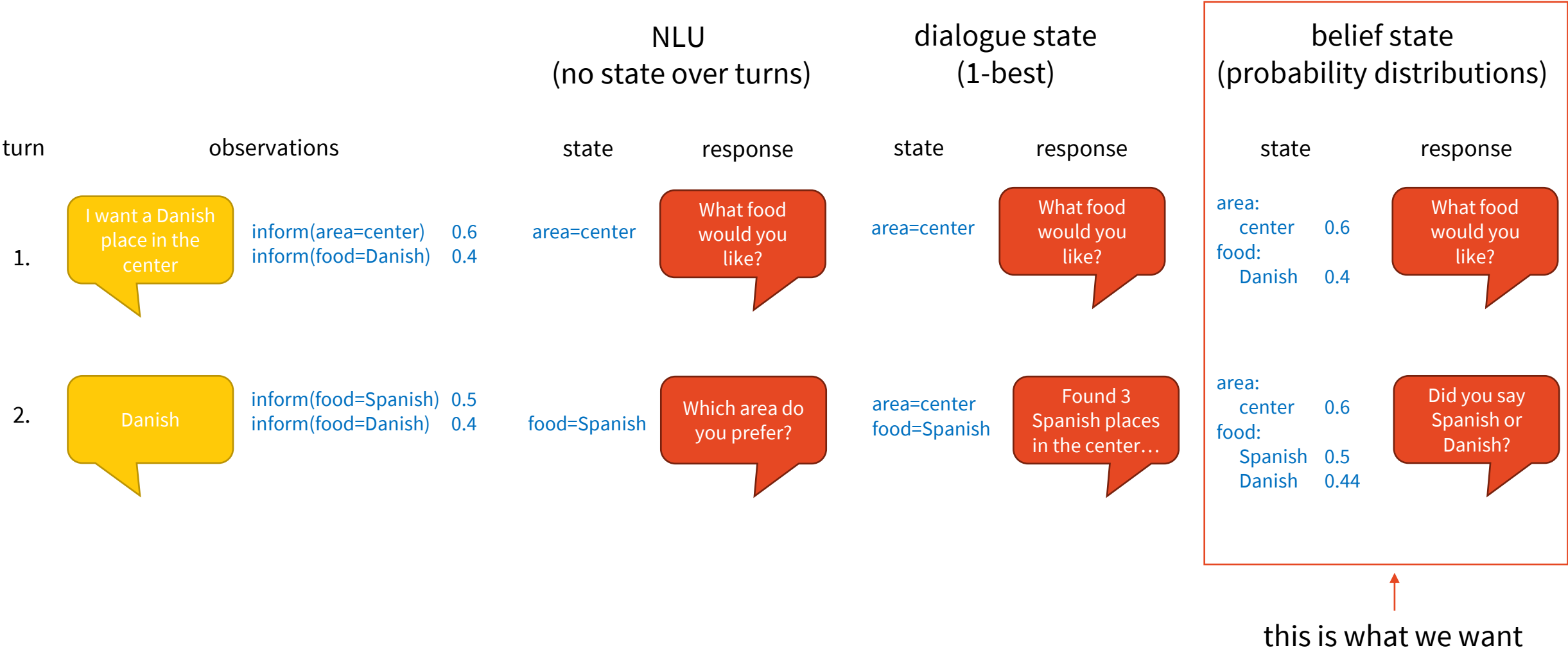
only hotels have stars!

- Better solution: make the state probabilistic – **belief state**

Belief State

- Assume we don't know the true current dialogue state s_t
 - states (what the user wants) influence **observations** o_t (what the system hears)
 - based on observations o_t & system actions a_t , we can estimate a probability distribution $b(s)$ over all possible states – **belief state**
- More robust than using dialogue state directly
 - accumulates probability mass over multiple turns
 - low confidence – if the user repeats it, we get it the 2nd time
 - accumulates probability over NLU n-best lists
- Plays well with probabilistic dialogue policies (POMDPs)
 - but not only them – rule-based, too

Belief State



(based on Milica Gašić's slides)

Basic Discriminative Belief Tracker (= what we used on the previous slide)

- **Partition the state** by assuming conditional independence

- simplify – assume each slot is independent:

- state $\mathbf{s} = [s^1, \dots, s^N]$, belief $b(\mathbf{s}_t) = \prod_i b(s_t^i)$

- **Always trust the NLU**

- this makes the model parameter-free
 - ...and basically rule-based
 - but very fast, with reasonable performance

NLU output

“user mentioned this value”

$$p(s_t^i | a_{t-1}^i, s_{t-1}^i, o_t^i) = \begin{cases} p(o_t^i) & \text{if } s_t^i = o_t^i \wedge o_t^i \neq \text{👉} \\ p(o_t^i) & \text{if } s_t^i = s_{t-1}^i \wedge o_t^i = \text{👉} \\ 0 & \text{otherwise} \end{cases}$$

“no change”

user silent about slot i

update rule

$$b(s_t^i) = \sum_{s_{t-1}^i, o_t^i} \underbrace{p(s_t^i | a_{t-1}^i, s_{t-1}^i, o_t^i)}_{\text{discriminative model}} b(s_{t-1}^i)$$

substitution

$$b(s_t^i) = \begin{cases} \text{“null value”} & s_t^i = \text{👉} : p(s_{t-1}^i = \text{👉}) p(o_t^i = \text{👉}) \\ \text{“not mentioned earlier”} & \text{“not mentioned now”} \\ \text{“non-null”} & \text{else: } p(o_t^i = s_{t-1}^i) + p(o_t^i = \text{👉}) p(s_{t-1}^i = s_{t-1}^i) \\ & \text{“mentioned now”} \quad \text{“carry-over”} \end{cases}$$

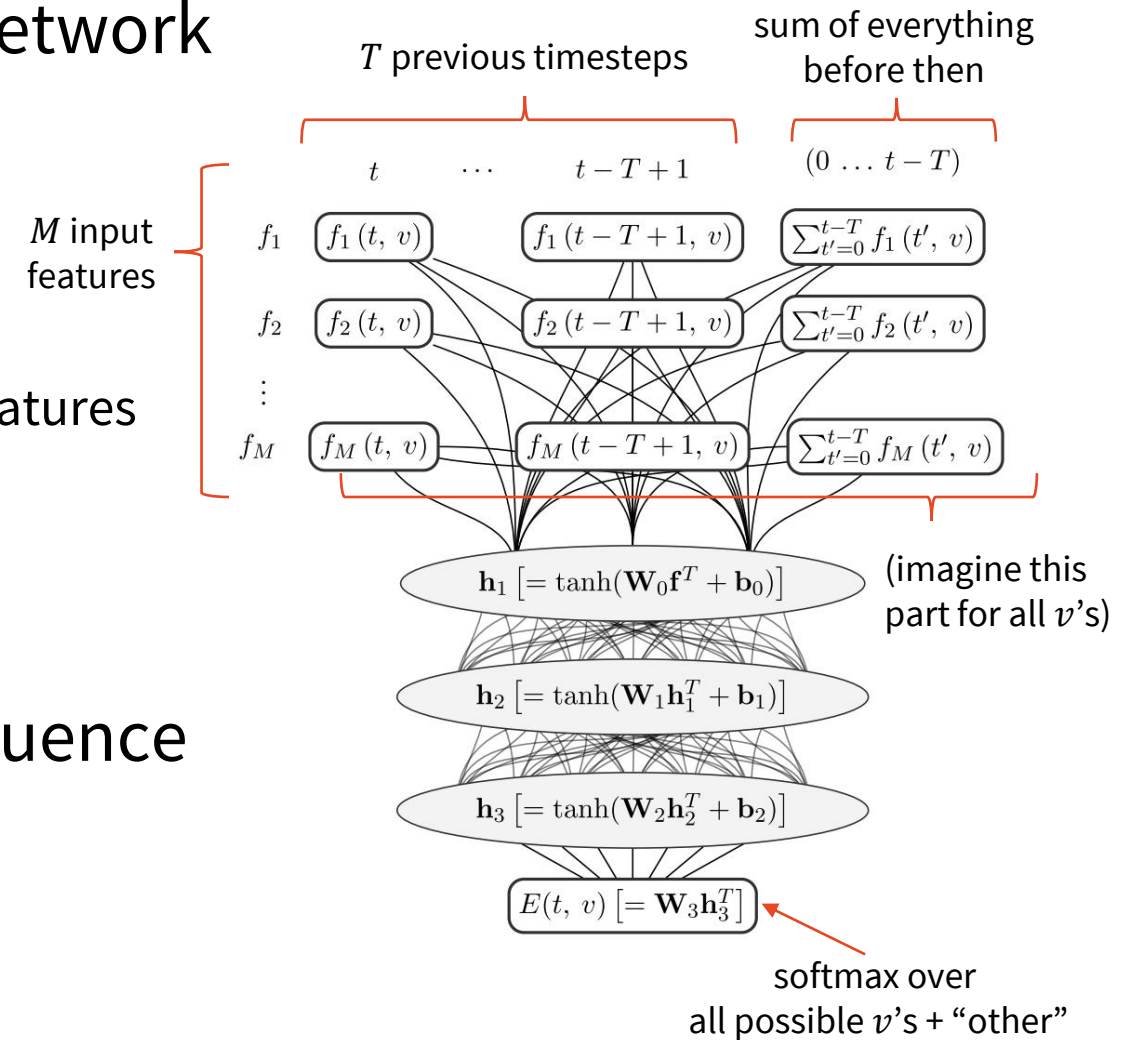
(Žilka et al., 2013)

<http://www.aclweb.org/anthology/W13-4070>

the belief state update rule is deterministic

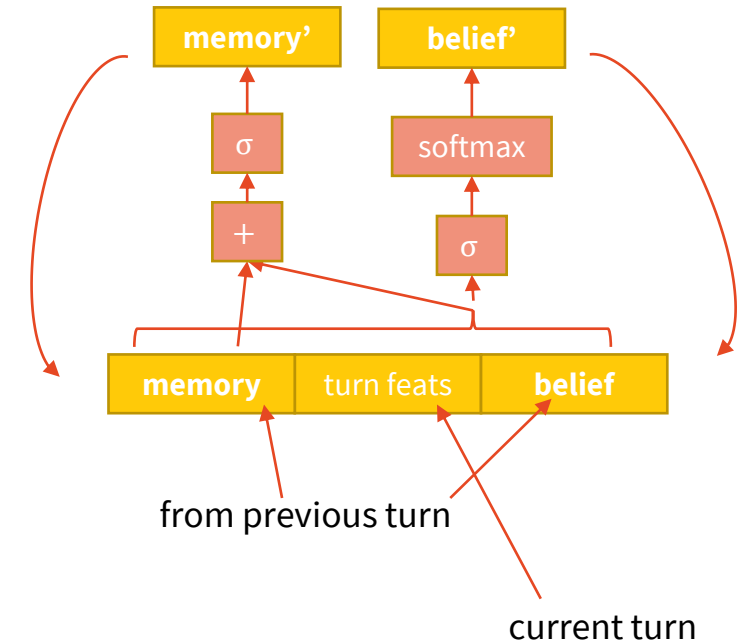
Basic Feed-forward Neural Tracker

- a simple feed-forward (fully connected) network
 - input – features (w.r.t. slot-value v & time t)
 - NLU score of v
 - n-best rank of v
 - user & system intent (*inform/request*)
 - ... – other domain-independent, low-level NLU features
 - 3 tanh layers
 - output – softmax
(= probability distribution over values)
- **static** – does not model dialogue as a sequence
 - uses a **sliding window**:
current time t + few steps back + \sum previous



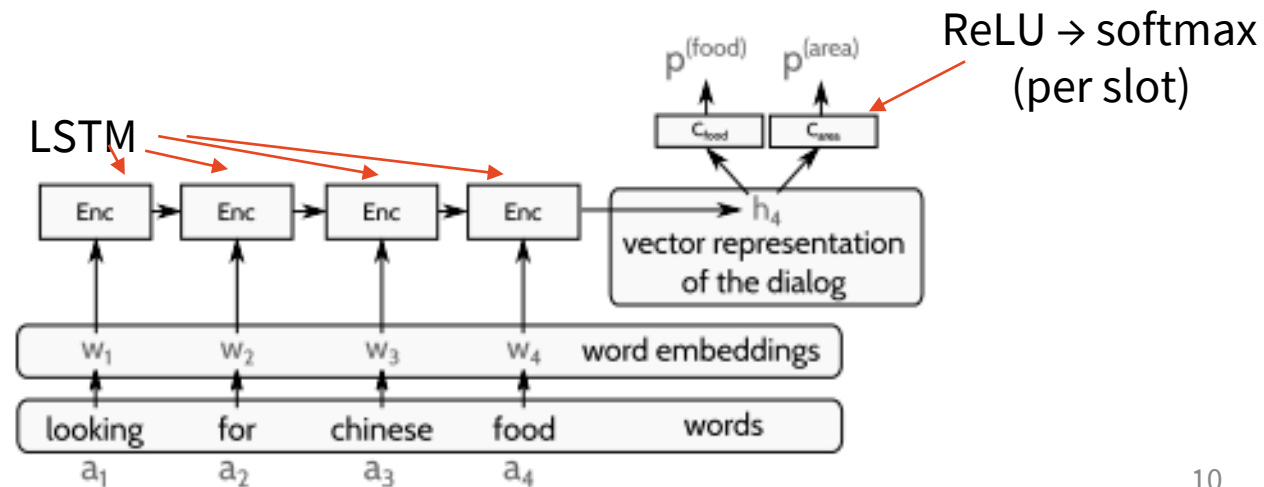
Basic RNN Tracker

- plain sigmoid RNN with a memory vector
 - not quite LSTM/GRU, but close
 - memory updated separately, used in belief update
 - turn-level LSTM would work similarly
- does not need NLU
 - turn features = lexicalized + delexicalized n -grams from ASR n-best list, weighted by confidence
- delexicalization is very harsh: <slot> <value>
 - you don't even know which slot it is
 - this apparently somewhat helps the system generalize across domains
- **dynamic** – explicitly models dialogue as sequence
 - using the network recurrence



Incremental Recurrent Tracker

- Simple: LSTM over words + classification on hidden states
 - runs over the whole dialogue history (user utterances + system actions)
 - classification can occur after each word, right as it comes in from ASR
- **Dynamic**/sequential
- Doesn't use any NLU
 - infrequent values are delexicalized (otherwise it can't learn them)
- Slightly worse performance – possible causes:
 - only uses ASR 1-best
 - very long recurrences (no hierarchy)



(Žilka & Jurčiček, 2015)
<https://dl.acm.org/citation.cfm?id=2955040>
<http://arxiv.org/abs/1507.03471>

Candidate Ranking

- Previous systems consider all values for each slot
 - this is a problem for open-ended slots (e.g. restaurant name)
 - enumerating over all takes ages, some are previously unseen
- Alternative: always consider just K candidates
 - use last K candidates from system actions and NLU output
 - NB: only way history is incorporated here (~static)
 - select from them using a per-slot softmax

(Rastogi et al., 2017)

<https://arxiv.org/abs/1712.10224>

(Goel et al., 2018)

<http://arxiv.org/abs/1811.12891>

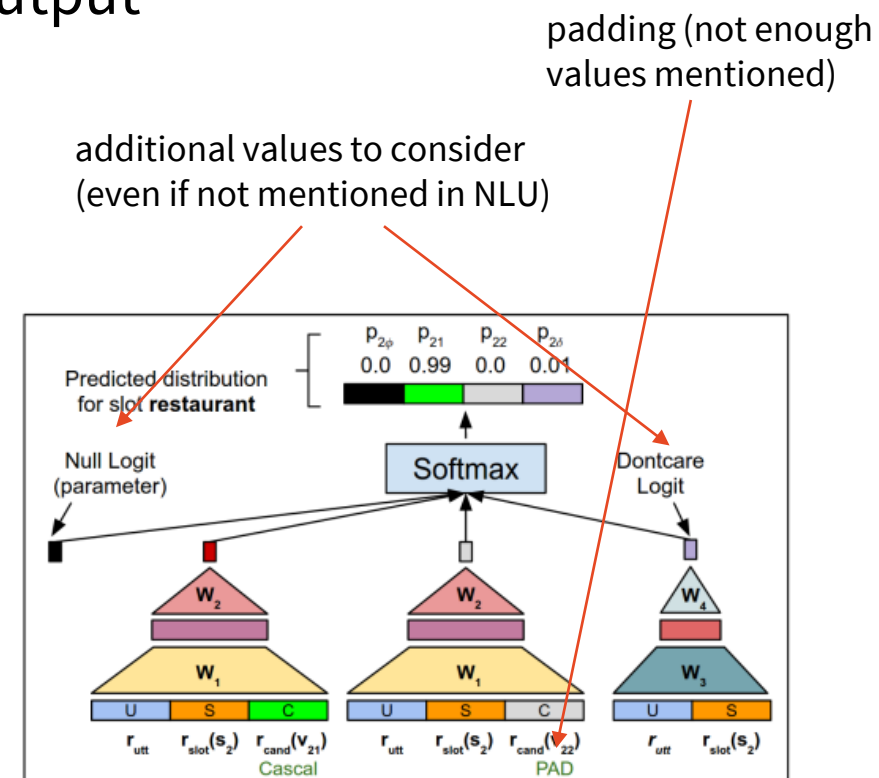
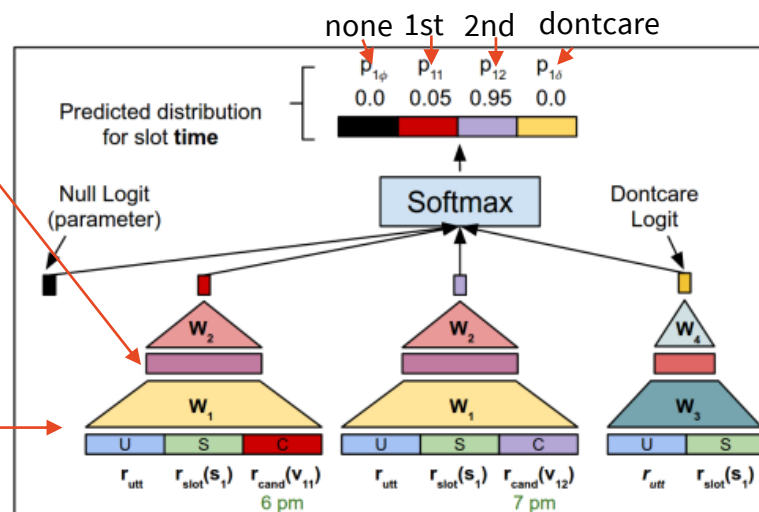
(Goel et al., 2019)

<http://arxiv.org/abs/1907.00883>

2 sigmoid layers

pictures assume $K = 2$

representation of
 i -th candidate:
utterance/slot/candidate features
(RNN hidden states
+ binary slot/val. presence in prev. turn)



BERT & Span Selection

a.k.a. Span Tagging

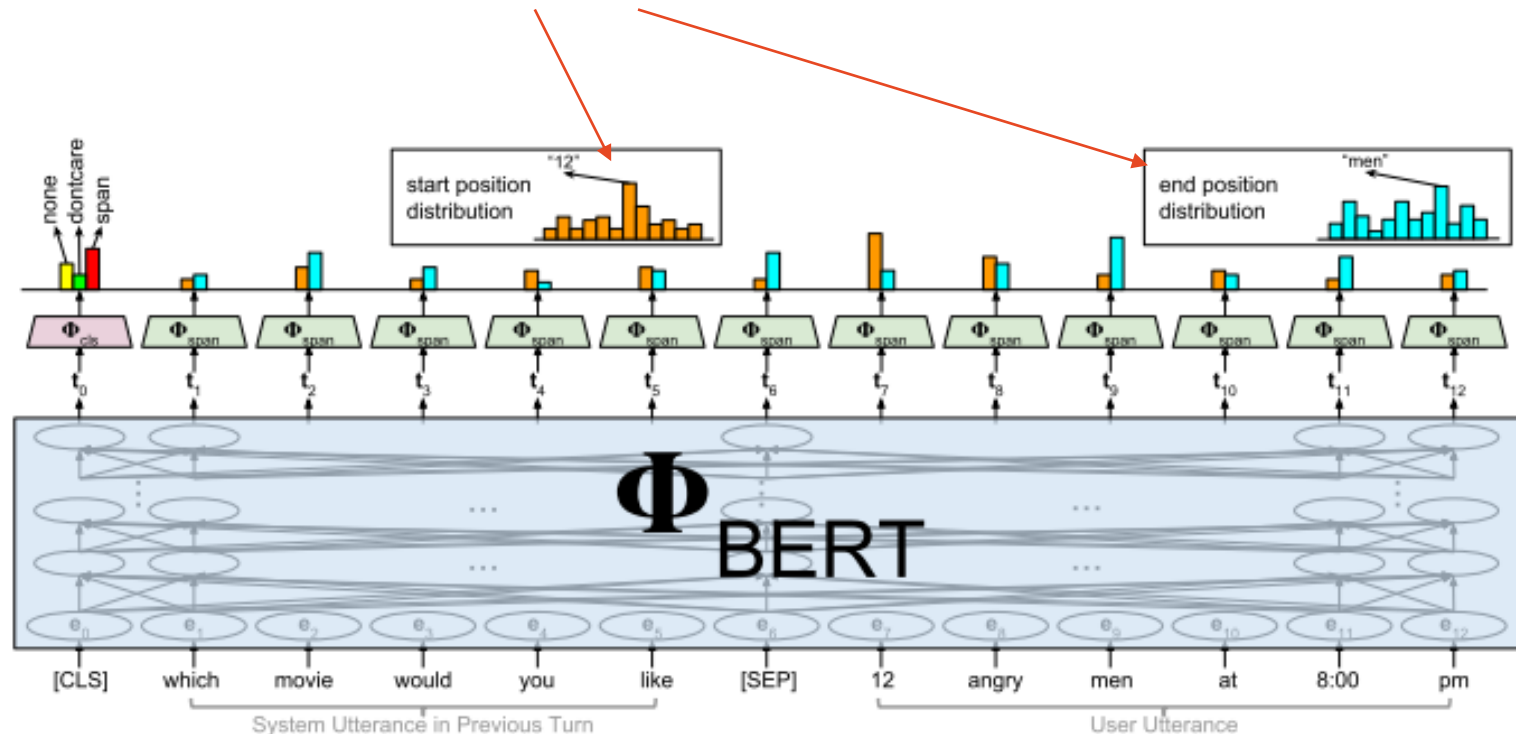
(~question answering/reading comprehension)

PLM span select

- BERT over previous system & current user utterance
- from 1st token's representation, get a **decision**: *none*/*dontcare*/*span*
 - per-slot (BERT is shared, but the final decision is slot-specific)
- span = need to find a concrete value as a span somewhere in the text
 - **predict start & end token** of the span using 2 softmaxes over tokens
- rule-based update (static):
 - if *none* is predicted, keep previous value

(Chao & Lane, 2019)

<http://arxiv.org/abs/1907.03040>

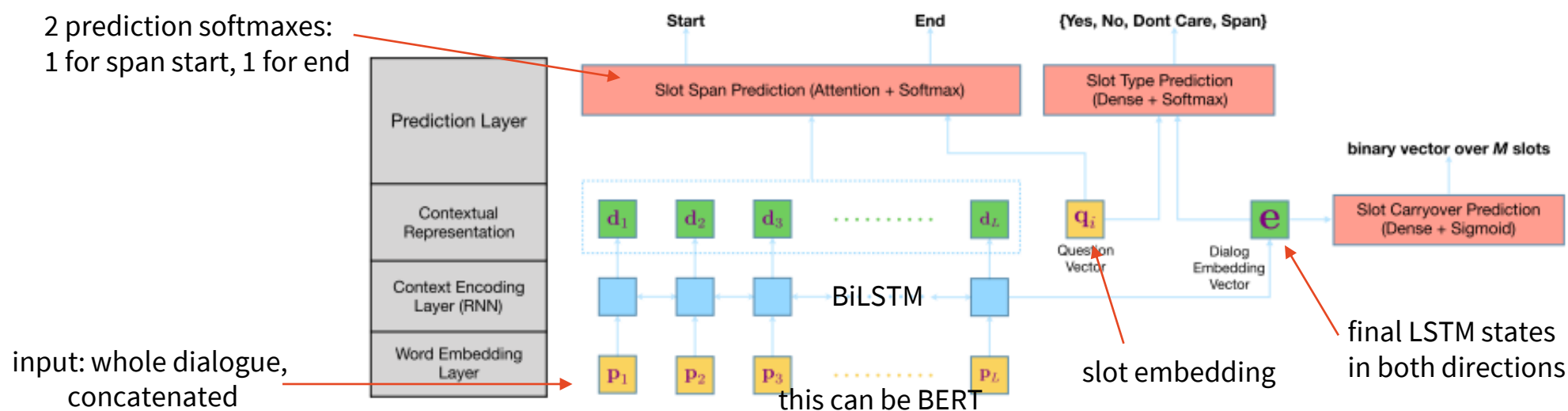


Span Selection with Modelled Update

- Also uses BERT, but not necessarily
 - works slightly worse with random-initialized word embeddings
- sequence of 3 decisions
 - do we carry over last turn's prediction? (Yes/No) (~static tracking, but not so rigid)
 - if no: what kind of answer are we looking for? (*yes/no/dontcare*/span of text)
 - if span: predict span's start and end

(Gao et al., 2019)

<https://www.aclweb.org/anthology/W19-5932/>



Span Selection & Better Copying

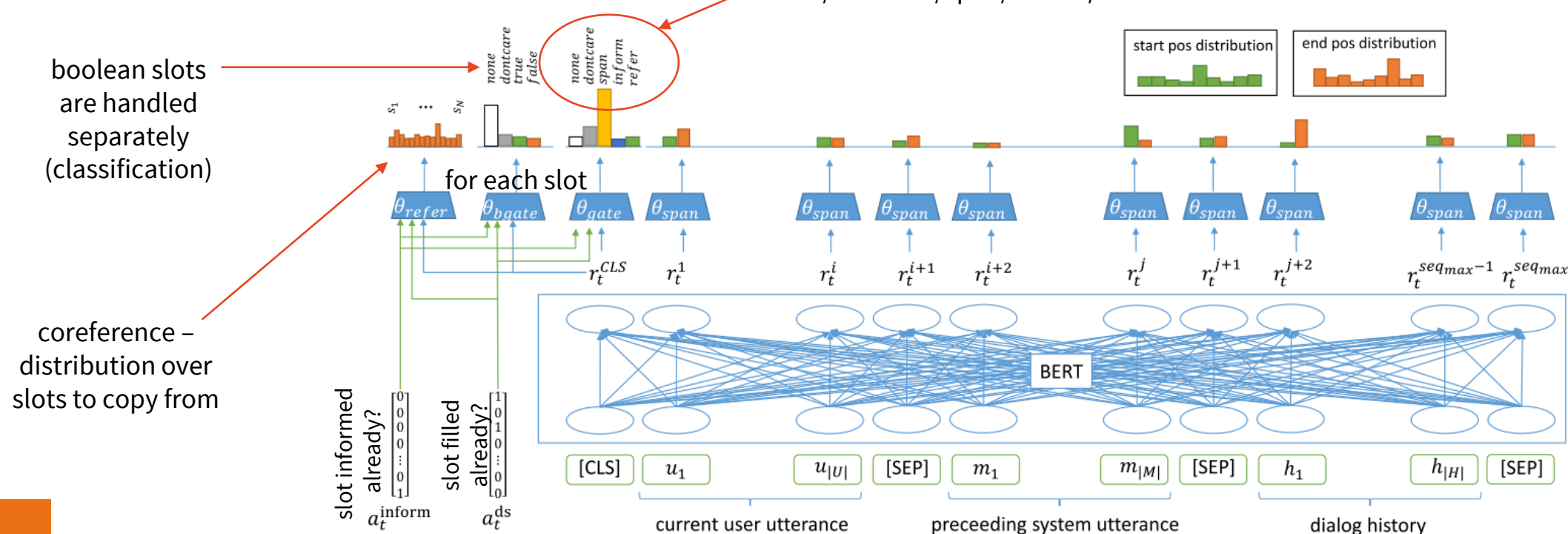
(Heck et al., 2020)

<https://aclweb.org/anthology/2020.sigdial-1.4/>

- “triple-copy” – gets the value from 3 sources:
 - user utterance (same as previous span tagging models)
 - system informs (last value the system mentioned)
 - another slot (coreference), e.g. a taxi ride to a hotel (hotel name = destination)
 - rule-based update (static)
- same decision as previously, just different options:

same decision as previously, just different options:

✓ none/dontcare/span/inform/refer

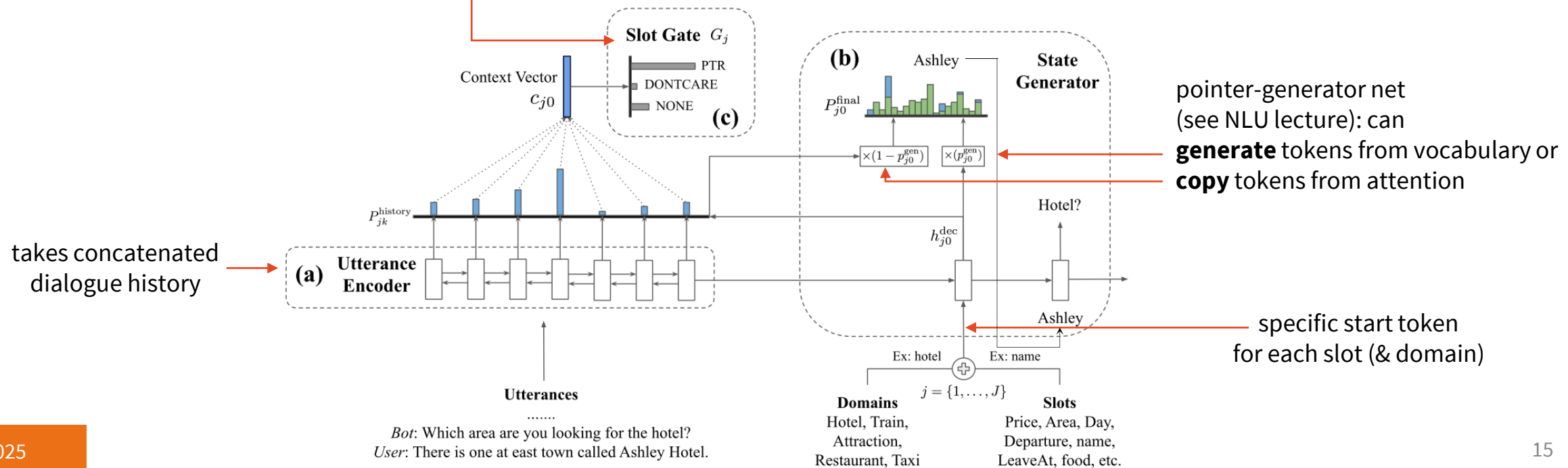


Generator-based Tracker

(Wu et al., 2019)

<https://www.aclweb.org/anthology/P19-1078>

- Similar to span selection: encodes whole dialogue history (static)
- Pointer-generator seq2seq decoder produces values
 - specific start token for each slot -- copies from input & generates new tokens
- Slot gate: “use generated”/*dontcare*/*none*
 - same as the decisions done in span tagging, just applied *after* getting the value

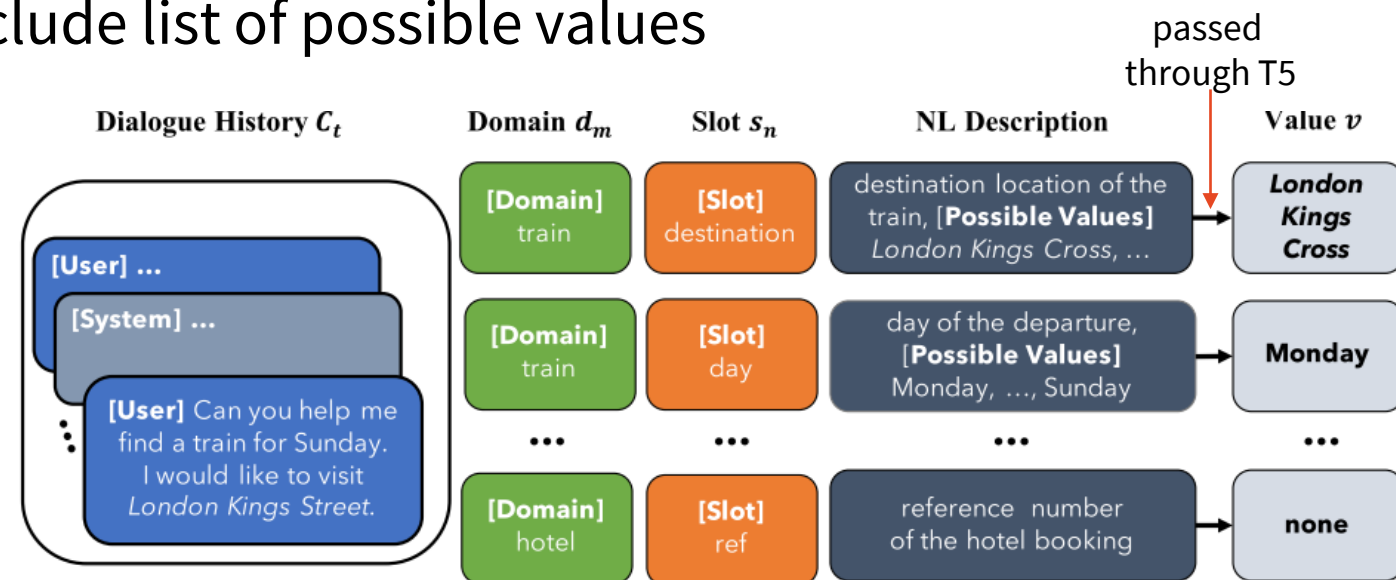
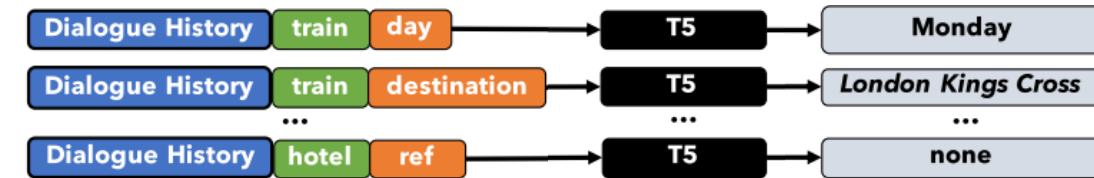


Generator + Pretrained LMs

(Lee et al., 2021)

<https://aclanthology.org/2021.emnlp-main.404/>

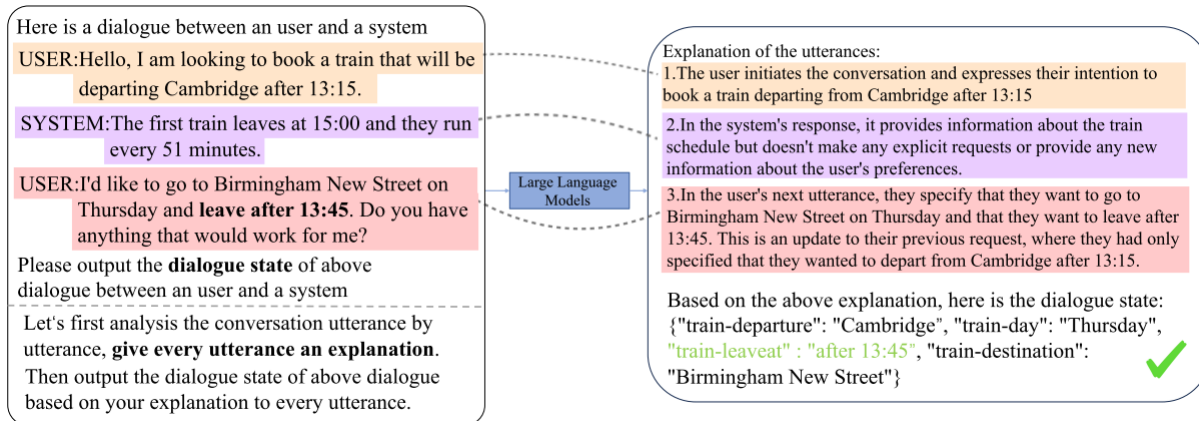
- Same as previous, but use a pretrained model (T5) + make it simpler
 - generate any value, including *none*
 - no explicit copying (T5 can copy itself)
- Finetune T5 with specific inputs (prompts)
 - dialogue history
 - domain + slot
 - (optional) slot description, may include list of possible values
- Generate just the slot value
 - may be multi-word
- T5 learns to use descriptions
- Potential for unseen domains
 - though not explored in the paper



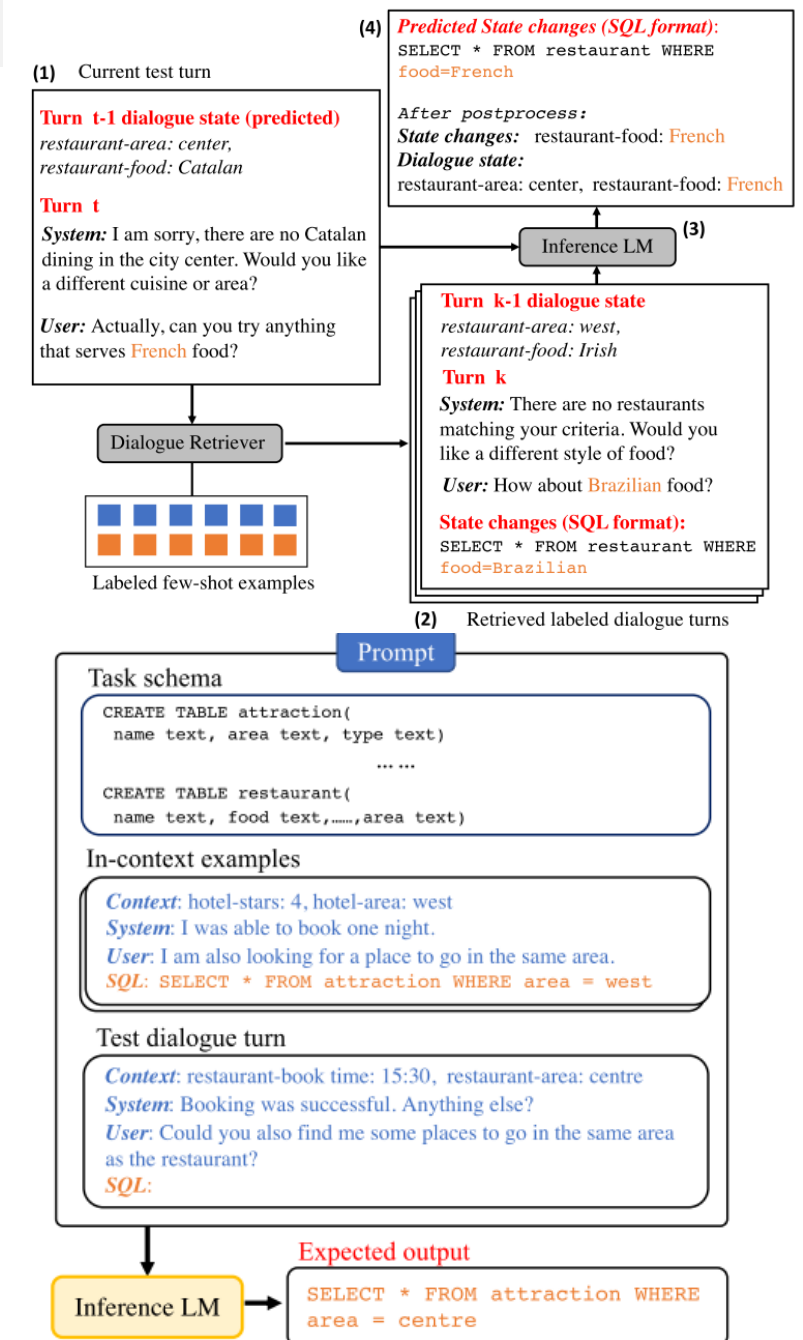
LLM Prompting

(Hu et al., 2022)
<https://aclanthology.org/2022.findings-emnlp.193>

- Prompt LLM to produce state (Neo-GPT, GPT3...)
 - Use SQL for DB schema & state changes
 - Add dialogue context & retrieved few-shot examples
 - Works well few-shot, with small retrieval DB (100-500 dialogues)
- Better prompting: explain before producing state
 - generate turn-level explanations
 - generate state based on them

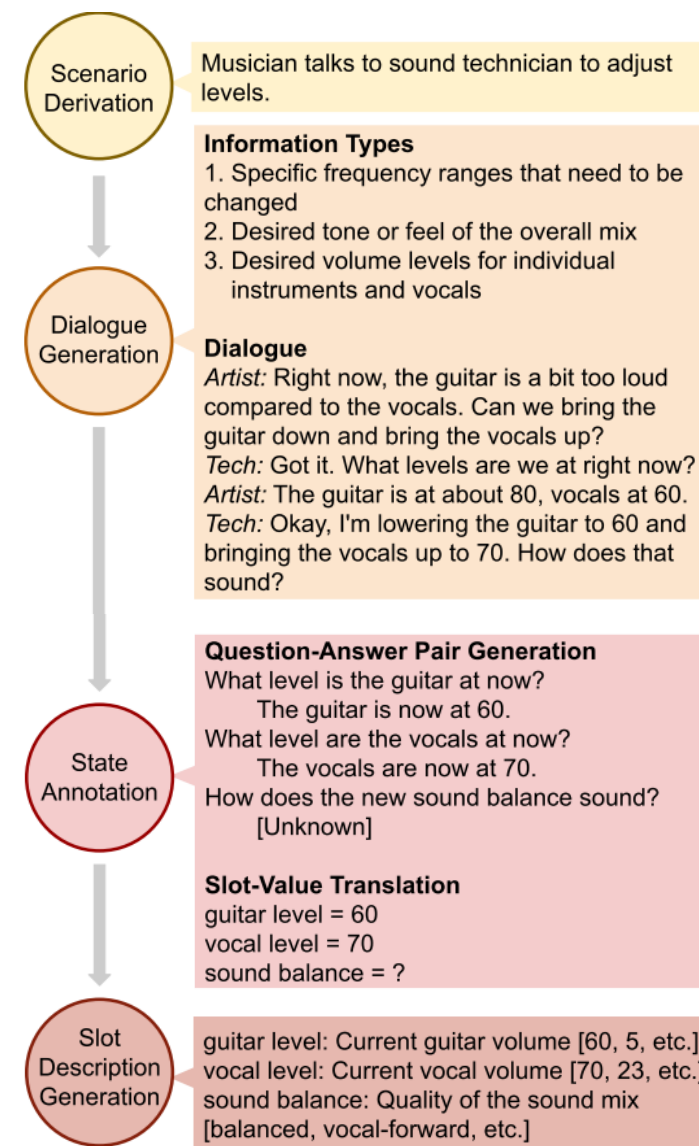


(Gao et al., 2024)
<https://aclanthology.org/2024.lrec-main.1269>



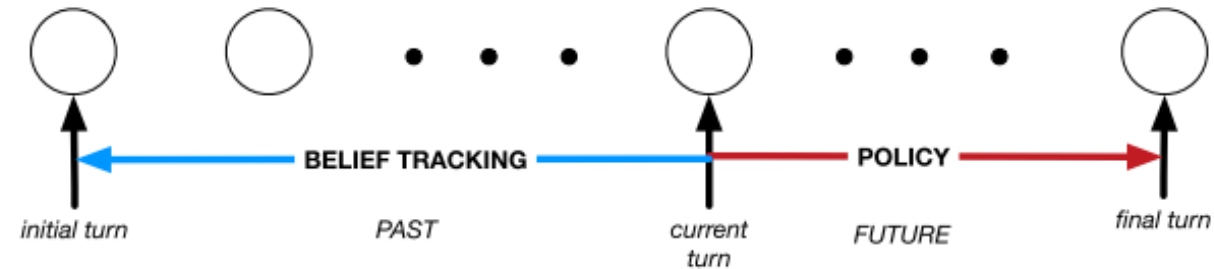
LLMs for data synthesis

- Synthesize data to train a PLM/LLM-based tracker (span select/generate)
- Pipeline – prompt an LLM to:
 - Prepare relevant scenarios/domains
 - additional deduplication step with SBERT
 - Prepare slot candidates
 - Generate full dialogues
 - Produce state annotation
 - Derive slot descriptions based on the annotation
- Needs some human validation



Action Selection / Policy

- Dialogue management:
 - **State tracking** (↑)
 - **Action selection/Policy** (↓)



(from Milica Gašić's slides)

- action selection – **deciding what to do next**
 - based on the current belief state – under uncertainty
 - following a **policy** (strategy) towards an end **goal** (e.g. book a flight)
 - controlling the coherence & flow of the dialogue
 - actions: linguistic & non-linguistic
- DM/policy should:
 - manage uncertainty from belief state
 - recognize & follow dialogue structure
 - plan actions ahead towards the goal

Did you say Indian or Italian?

follow convention, don't be repetitive

e.g. ask for all information you require

Action Selection Approaches

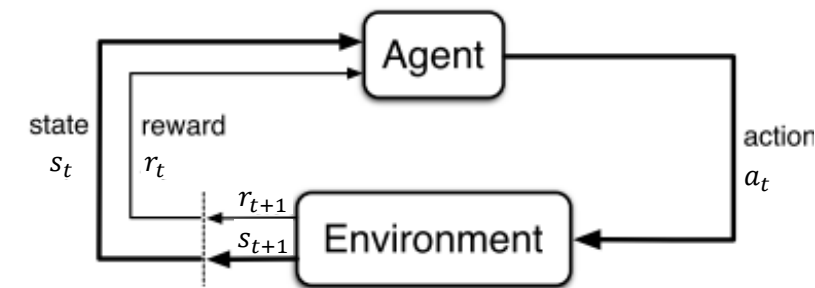
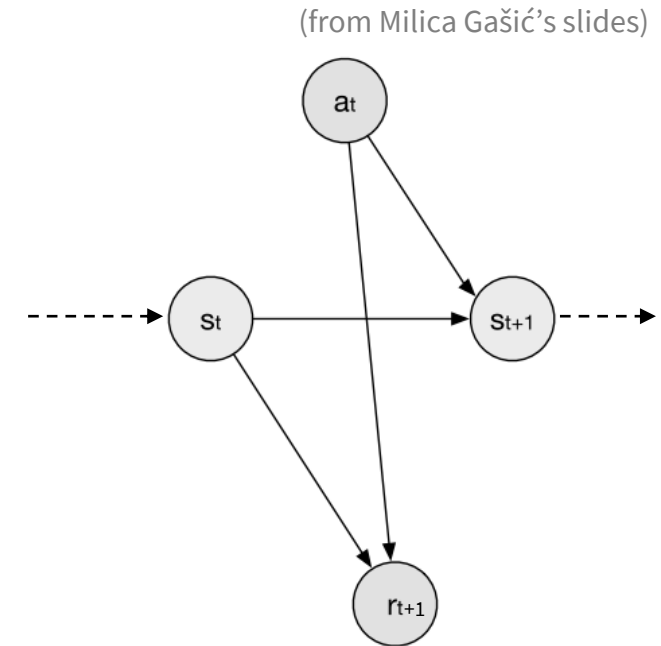
- Finite-state machines
 - simplest possible
 - dialogue state is machine state
- Frame-based (VoiceXML)
 - slot-filling + providing information – basic agenda
 - rule-based in essence
- Rule-based
 - any kind of rules (e.g. Python code)
- **Statistical**
 - typically using **reinforcement learning**

Why Reinforcement Learning

- **Action selection ~ classification** → use supervised learning?
 - set of possible actions is known
 - belief state should provide all necessary features
- Yes, but...
 - You'd **need** sufficiently large **human-human data** – hard to get
 - human-machine would just mimic the original system
 - Dialogue is ambiguous & complex
 - there's **no single correct next action**– multiple options may be equally good
 - but datasets will only have one next action
 - **some paths will be unexplored** in data, but you may encounter them
 - DSs won't behave the same as people
 - ASR errors, limited NLU, limited environment model/actions
 - **DSs should behave differently** – make the best of what they have
 - supervised classification **doesn't plan ahead!**
 - RL optimizes for the whole dialogue, not just the immediate action

RL World Model: Markov Decision Process

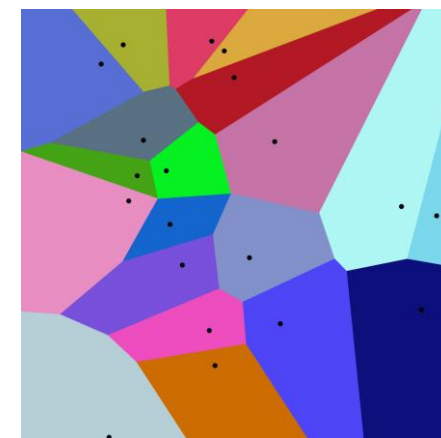
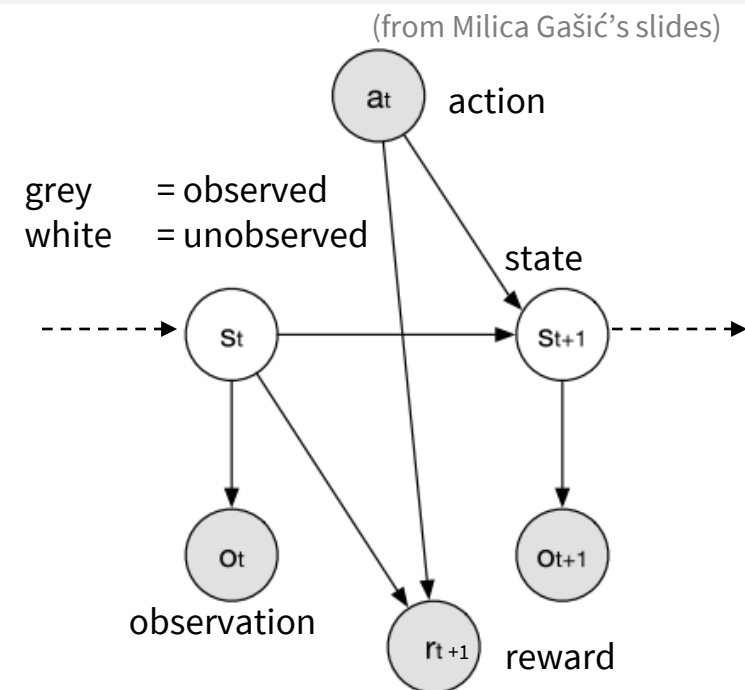
- MDP = probabilistic control process
 - modelling situations that are partly random, partly controlled
 - **agent** in an **environment**:
 - has internal **state** $s_t \in \mathcal{S}$ (\sim dialogue state)
 - takes **actions** $a_t \in \mathcal{A}$ (\sim system dialogue acts)
 - actions chosen according to **policy** $\pi: \mathcal{S} \rightarrow \mathcal{A}$
 - gets **rewards** $r_t \in \mathbb{R}$ & state changes from the environment
 - rewards are typically handcrafted
 - very high positive for a successful dialogue (e.g. +40)
 - high negative for unsuccessful dialogue (-10)
 - small negative for every turn (-1, promote short dialogues)
 - Markov property – state defines everything
 - no other temporal dependency
 - policy may be **deterministic** or **stochastic**
 - stochastic: prob. dist. of actions, sampling



(Sutton & Barto, 2018)

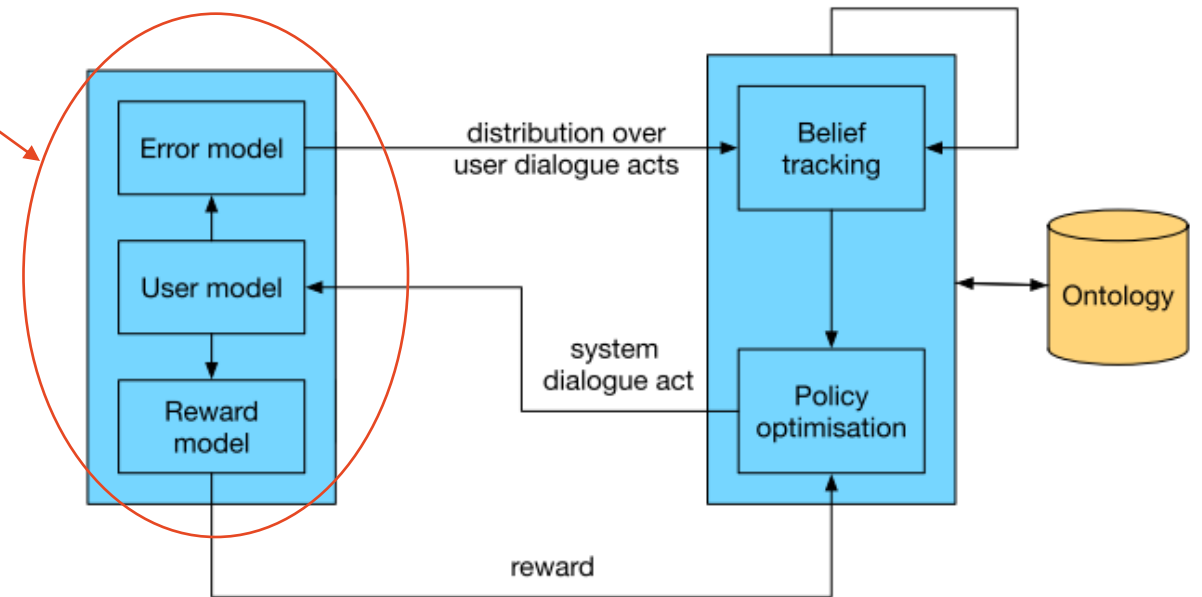
Partially-observable MDPs

- POMDPs – **belief** states instead of dialogue states
 - true states (“what the user wants”) are not observable
 - observations (“what the system hears”) depend on states
 - belief – probability distribution over states
 - can be viewed as **MDPs with continuous-space states**
 - just represent 1 slot as set of binary floats 😊
- All MDP algorithms work...
 - if we **quantize/discretize** the states
 - use grid points & nearest neighbour approaches
 - this might introduce errors / make computation complex
- Deep RL typically works out of the box
 - function approximation approach, allows continuous states



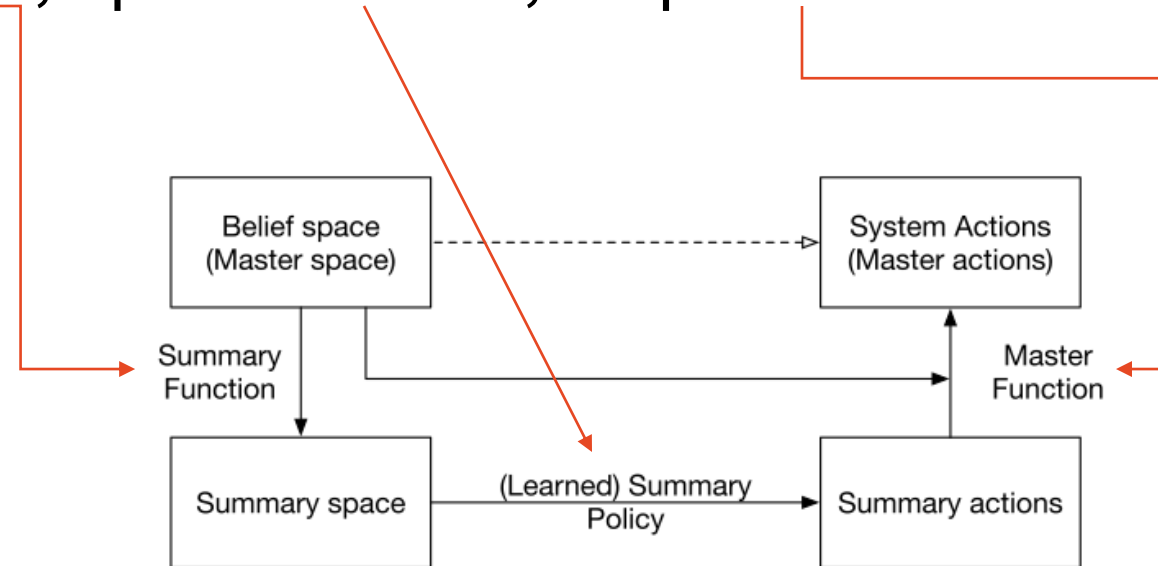
Simulated Users

- Static datasets aren't enough for RL
 - data might not reflect our newly learned behaviour
- RL needs a lot of data, more than real people would handle
 - 1k-100k's dialogues used for training, depending on method
- solution: **user simulation**
 - basically another DS/DM
 - (typically) working on DA level
 - errors injected to simulate ASR/NLU
- approaches:
 - rule-based (frames/agenda)
 - n-grams
 - MLE/supervised policy from data
 - combination (best!)



Summary Space

- for a typical DS, the belief state is too large to make RL tractable
- solution: map state into a reduced space, optimize there, map back
- reduced space = **summary space**
 - handcrafted state features
 - e.g. top slots, # found, slots confirmed...
- reduced action set = **summary actions**
 - e.g. just DA types (*inform, confirm, reject*)
 - remove actions that are not applicable
 - with handcrafted mapping to real actions
- state is still tracked in original space
 - we still need the complete information for accurate updates



(from Milica Gašić's slides)

Reinforcement learning: Definition

- RL = finding a **policy that maximizes long-term reward**
 - unlike supervised learning, we don't know if an action is good
 - immediate reward might be low while long-term reward high

accumulated long-term reward
(from turn t onwards)

$$R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1}$$

alternative – **episodes**: only count to T when we encounter a terminal state
(e.g. 1 episode = 1 dialogue)

$\gamma \in [0,1]$ = **discount factor**
(immediate vs. future reward trade-off)

$\gamma = 1$: no discount, only usable if $i \leq T$
 $\gamma < 1$: R_t is finite (if r_t is finite)
 $\gamma = 0$: greedy approach (ignore future rewards)

- state transition is stochastic \rightarrow maximize **expected return**

$$\mathbb{E}[R_t | \pi, s_0] \quad \leftarrow \text{expected } R_t \text{ if we start from state } s_0 \text{ and follow policy } \pi$$

Summary

- **State tracking:** track user goal over multiple turns (probabilistic – **belief state**)
 - good NLU + rules – works well (and is used frequently)
 - **static** (sliding-window/rule-based update) vs. **dynamic** (explicit modelling)
 - with vs. without NLU
 - **classification** vs. candidate **ranking** vs. span **selection** vs. **generation**
 - classifiers are more accurate than rankers but slower, limited to seen values
 - span selection or generation are the SotA approaches, work nicely but relatively slow
 - many architectures (FC/RNN), newest mostly based on pretrained LMs
- **Action selection:** deciding what to do next (following a **policy**)
 - FSM, frames, rule-based, supervised, **reinforcement learning**
 - **RL** – agent in an environment, taking actions, getting rewards
 - MDP formalism (+POMDP can be converted to it)
 - summary states might be needed
 - trained often with user simulators

Thanks

Contact us:

[https://ufaldsg.slack.com/
odusek@ufal.mff.cuni.cz](https://ufaldsg.slack.com/odusek@ufal.mff.cuni.cz)

Skype/Meet/Zoom/Troja (by agreement)

Labs in 10 minutes

Get these slides here:

<http://ufal.cz/npfl099>

**Next Mon 15:40
Dialogue Policy**

References/Inspiration/Further:

- Filip Jurčiček's slides (Charles University): <https://ufal.mff.cuni.cz/~jurcicek/NPFL099-SDS-2014LS/>
- Milica Gašić's slides (Cambridge University): <http://mi.eng.cam.ac.uk/~mg436/teaching.html>
- Henderson (2015): Machine Learning for Dialog State Tracking: A Review <https://ai.google/research/pubs/pub44018>
- Sutton & Barto (2018): Reinforcement Learning: An Introduction (2nd ed.)
<http://incompleteideas.net/book/the-book.html>
- Heidrich-Meisner et al. (2007): Reinforcement Learning in a Nutshell: <https://christian-igel.github.io/paper/RLiaN.pdf>
- Young et al. (2013): POMDP-Based Statistical Spoken Dialog Systems: A Review:
<http://cs.brown.edu/courses/csci2951-k/papers/young13.pdf>