NPFL099 Statistical Dialogue Systems
6. Dialogue Management (1)
mostly Dialogue State Tracking

http://ufal.cz/npfl099

Ondřej Dušek, , Simone Balloccu, Zdeněk Kasner, Mateusz Lango,
Ondřej Plátek, Patrícia Schmidtová

7. 11. 2023

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

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 <u>the first one</u> you talked about.*
U: *Is there <u>any other</u> 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
- Possible solutions:
  - detect contradictions, ask for confirmation
  - ignore low-confidence NLU input
    - what's "low"?
    - what if we ignore 10x the same thing?
- Better solution: make the state probabilistic – **belief state**

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

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

only hotels have stars!

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

			NLU (no state over turns)		dialogue state (1-best)		belief state (probability distributions)	
turn	observations		state	response	state	response	state	response
1.	I want a Danish place in the center	inform(area=center) 0.6 inform(food=Danish) 0.4	area=center	What food would you like?	area=center	What food would you like?	area: center 0.6 food: Danish 0.4	What food would you like?
2.	Danish	inform(food=Spanish) 0.5 inform(food=Danish) 0.4	food=Spanish	Which area do you prefer?	area=center food=Spanish	Found 3 Spanish places in the center…	area: center 0.6 food: Spanish 0.5 Danish 0.44	Did you say Spanish or Danish?

this is what we want

(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

$$p(s_t^i | a_{t-1}^i, s_{t-1}^i, o_t^i) = \begin{cases} p(o_t^i) \text{ if } \overbrace{s_t^i = o_t^i \land o_t^i \neq \text{🤫}}^{\text{“user mentioned this value”}} \\ p(o_t^i) \text{ if } \underbrace{s_t^i = s_{t-1}^i \land o_t^i = \text{🤫}}_{\text{“no change”}} \\ 0 \text{ otherwise} \end{cases}$$

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

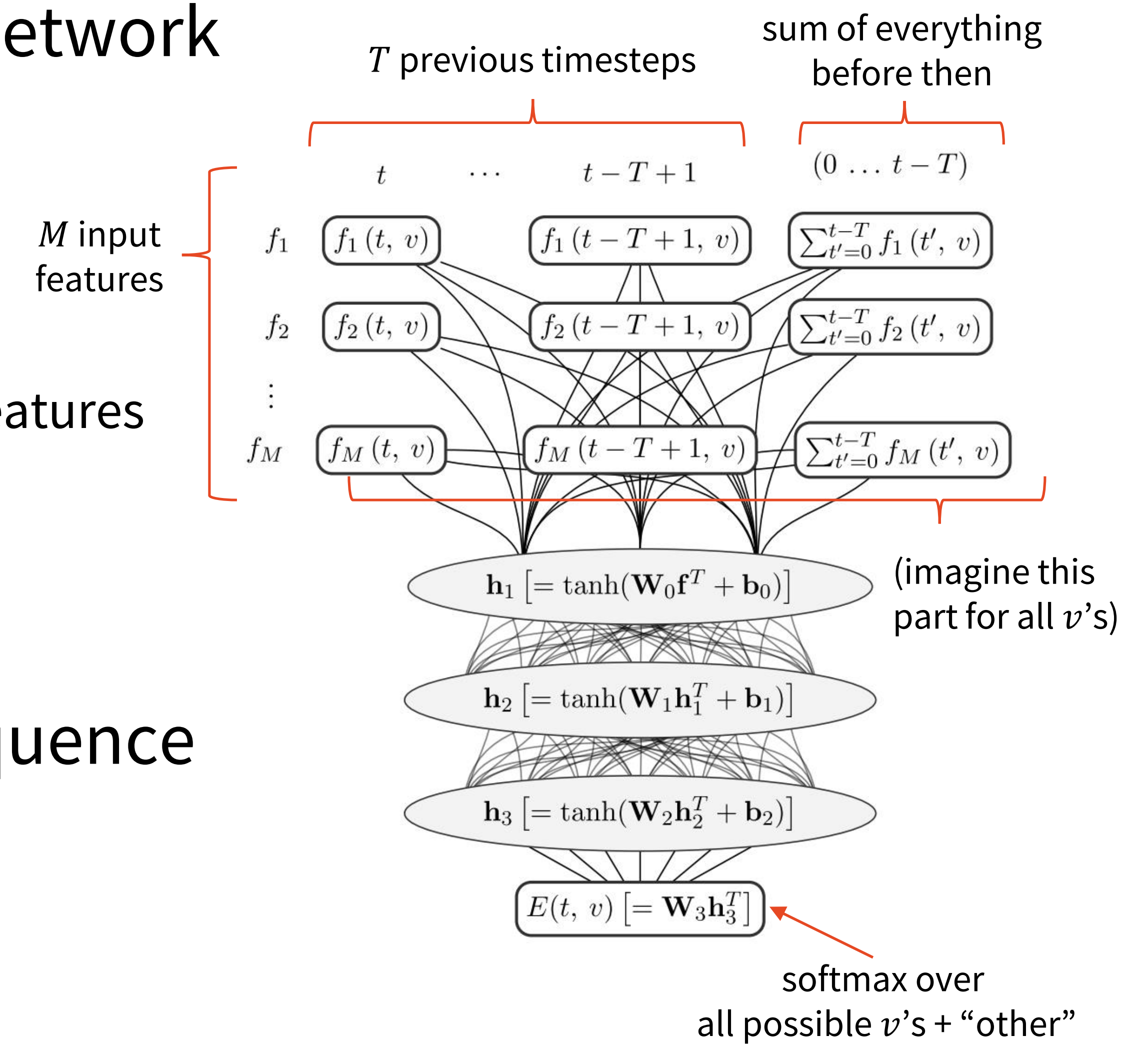
$$b(s_t^i) = \begin{cases} \overset{\text{“null value”}}{s_t^i = \text{🤫}}: & \overset{\text{“not mentioned earlier”}}{p(s_{t-1}^i = \text{🤫})} \overset{\text{“not mentioned now”}}{p(o_t^i = \text{🤫})} \\ \underset{\text{“non-null”}}{\text{else}}: & \underset{\text{“mentioned now”}}{p(o_t^i = s_t^i)}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

the belief state update rule is deterministic

(Žilka et al., 2013)
http://www.aclweb.org/anthology/W13-4070

Basic Feed-forward Neural Tracker

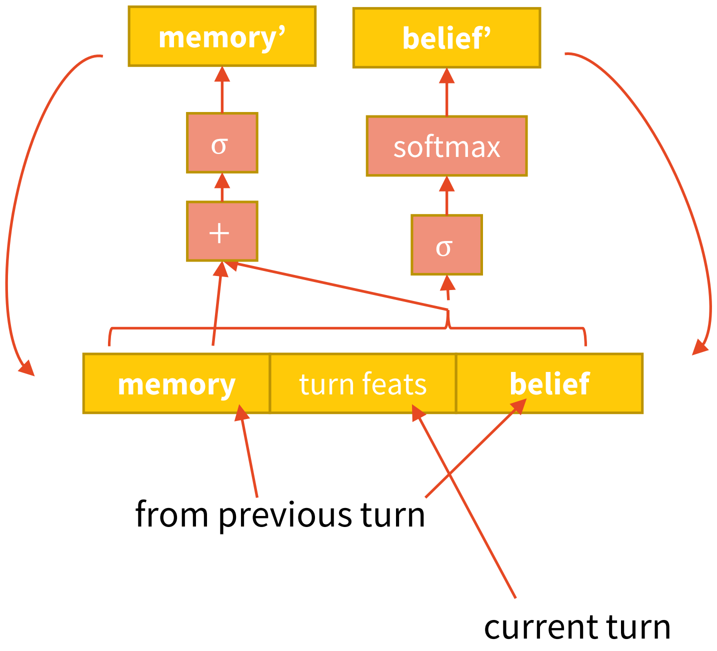
- a simple feed-forward (fully connected) network
  - input – features (w.r.t. slot-value v & time t)
    - NLU score of v
    - n-best rank of v
    - user & system intent (*inform*/*request*)
    - … – other domain-independent, low-level NLU features
  - 3 tanh layers
  - output – softmax
    (= probability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



(Henderson et al., 2013)
https://aclweb.org/anthology/W13-4073

Basic RNN Tracker

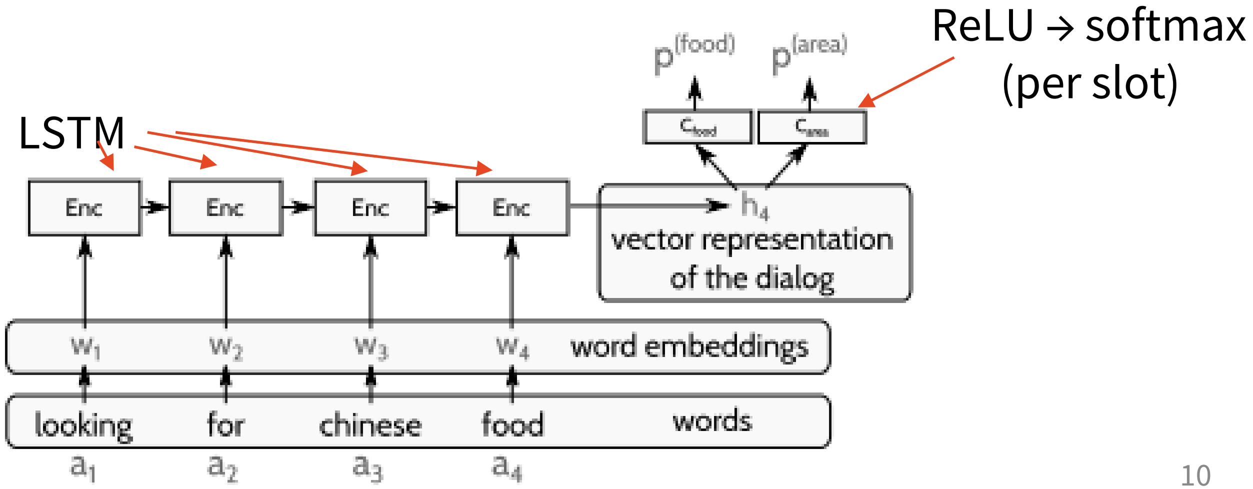
- plain sigmoid RNN with a memory vector
  - not quite LSTM/GRU, but close
  - memory updated separately, used in belief update
  - turn-level LSTM would work similarly
- does not need NLU
  - turn features = lexicalized + delexicalized *n*-grams from ASR n-best list, weighted by confidence
- delexicalization 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



(Mrkšić et al., 2015)
http://arxiv.org/abs/1506.07190

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 & Jurčíček, 2015)
https://dl.acm.org/citation.cfm?id=2955040
http://arxiv.org/abs/1507.03471

Candidate Ranking

(Rastogi et al., 2017)
https://arxiv.org/abs/1712.10224

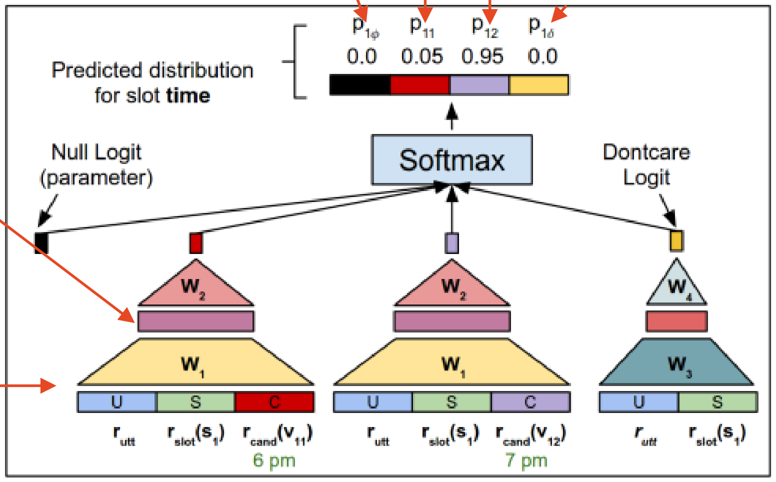
- Previous systems consider all values for each slot
  - this is a problem for open-ended slots (e.g. restaurant name)
  - enumerating over all takes ages, some are previously unseen
- Alternative: always consider just K candidates
  - use last K candidates from system actions and NLU output
    - NB: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

pictures assume $K = 2$

none 1st 2nd dontcare

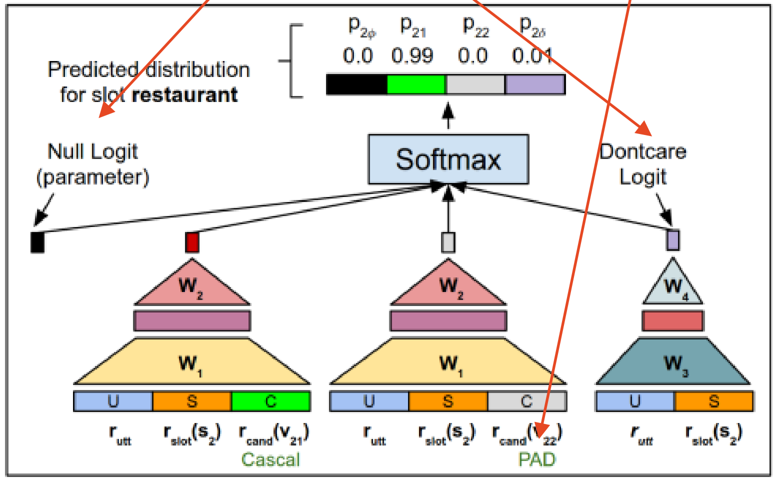
2 sigmoid layers

representation of
i-th candidate:
utterance/slot/candidate
features (next slide)



padding (not enough values mentioned)

additional values to consider (even if not mentioned in NLU)



Candidate Ranking

Representation

- BiGRU lexicalized/delex. utterances + binary (~presence slot/val. in prev. turn)

Extensions

- What if multiple values are true?
  - previous approach picks one (softmax)
  - use set of binary classifiers (log loss) instead

(Goel et al., 2018)
http://arxiv.org/abs/1811.12891

- Making it dynamic
  - embedding previous states, system actions, text of the whole dialogue
- Hybrid classify/rank
  - ranking is faster & more flexible vs. classification can be more accurate for some slots
    - generally ranking better with many values, classification with fewer values
  - check for performance on development data & decide which model to use

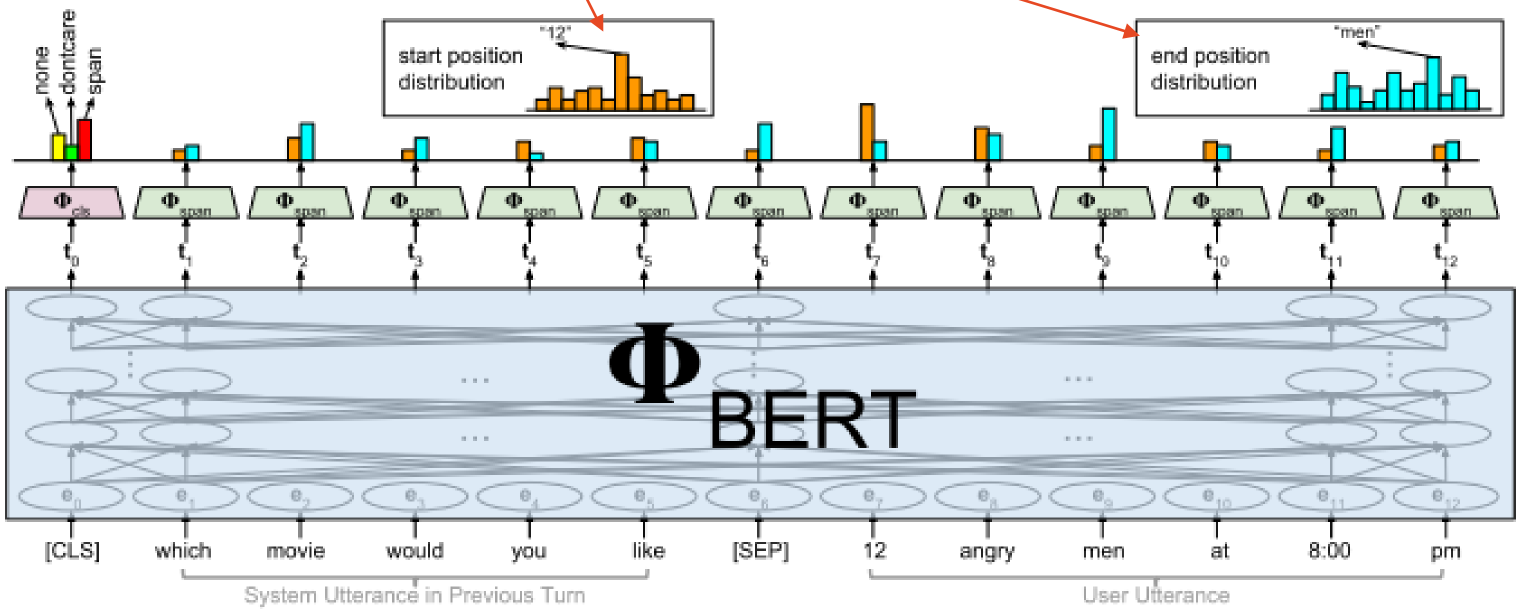
(Goel et al., 2019)
http://arxiv.org/abs/1907.00883

BERT & Span Selection

a.k.a. Span Tagging
(~question answering/reading comprehension)

(Chao & Lane, 2019)
http://arxiv.org/abs/1907.03040

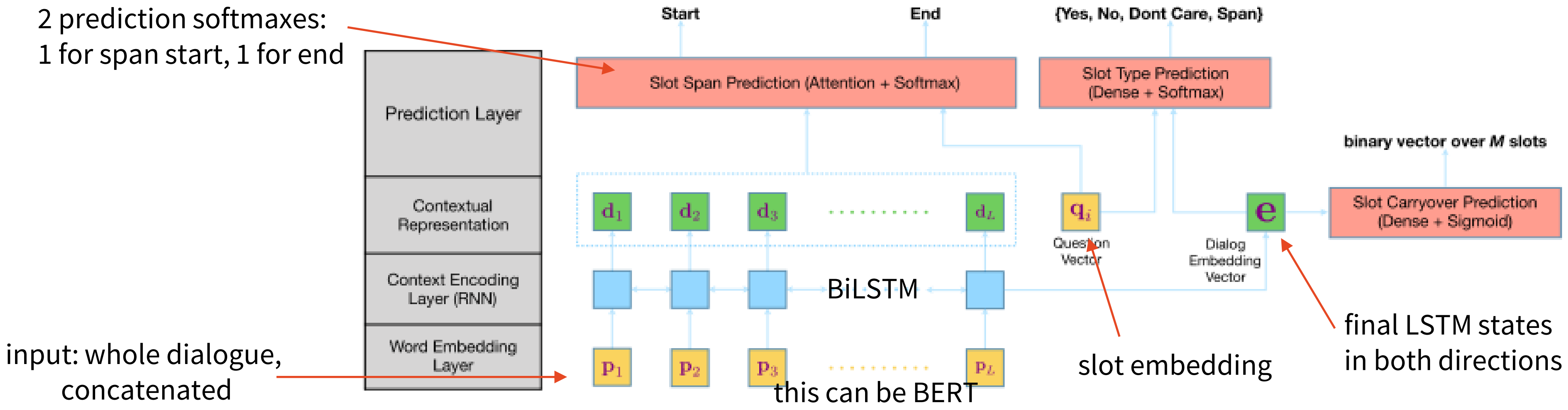
- BERT over previous system & current user utterance
- from 1st token's representation, get a **decision:** ***none*/*dontcare*/span**
  - per-slot (BERT is shared, but the final decision is slot-specific)
- span = need to find a concrete value as a span somewhere in the text
  - **predict start & end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



Span Selection with Modelled Update

(Gao et al., 2019)
https://www.aclweb.org/anthology/W19-5932/

- 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



Span Selection & Better Copying

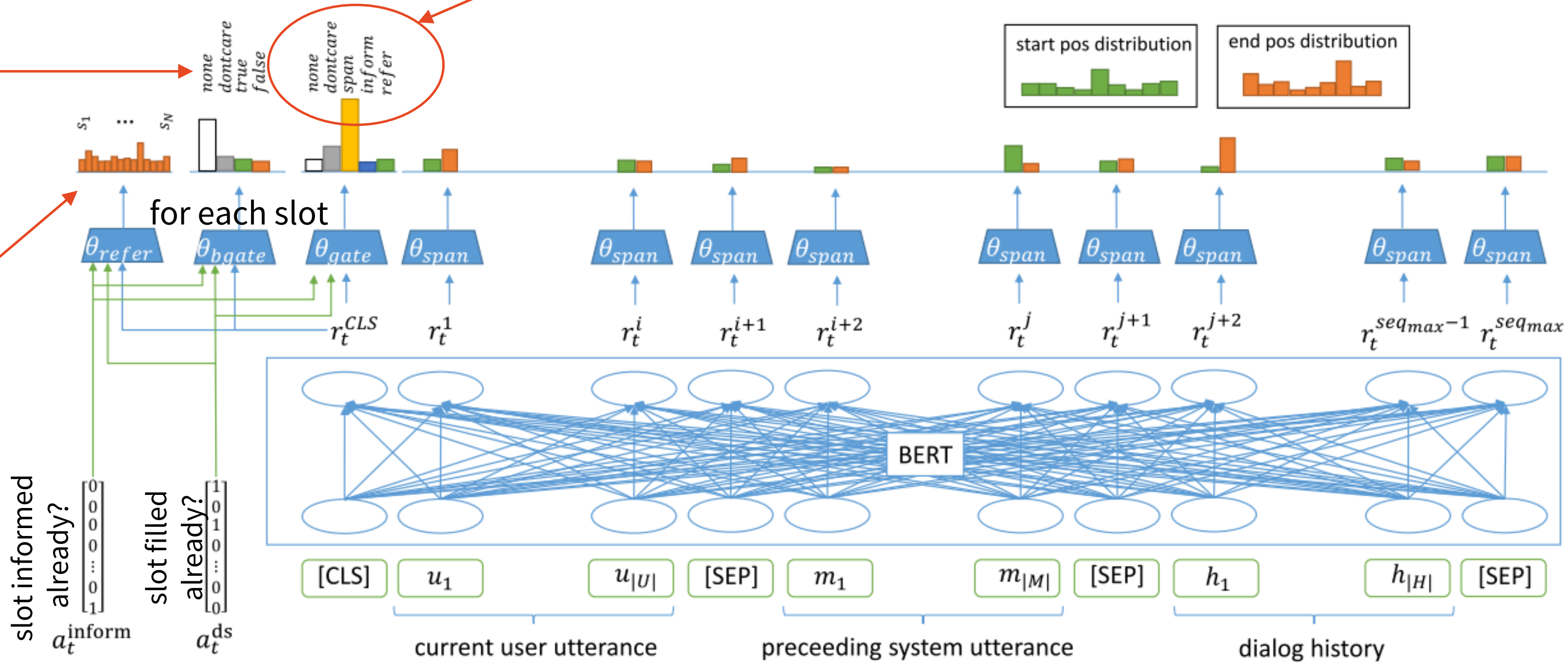
(Heck et al., 2020)
https://aclweb.org/anthology/2020.sigdial-1.4/

- "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:
none/*dontcare*/span/inform/refer

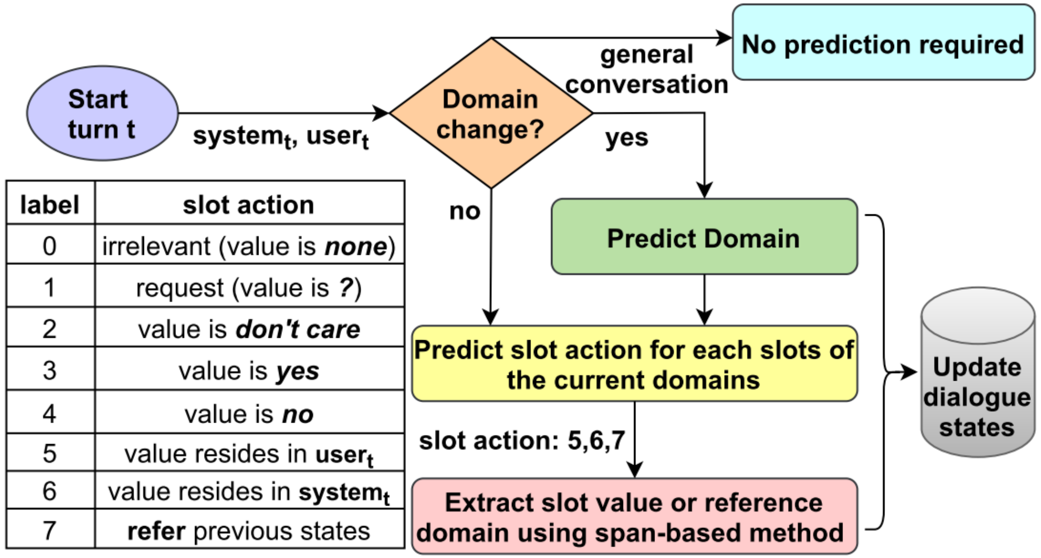
boolean slots are handled separately (classification)

coreference – distribution over slots to copy from



Multi-domain Span Selection

- encode domain & slot names w. static pretrained word-embeddings (GloVe)
  - adding **new unseen domains & slots** is easy (no retraining)
- otherwise similar as previous, BERT-based:
  - decide if domain changed (BERT: yes/no/chitchat)
  - if yes, detect new domain(s) (BERT + GloVe: 1/0 for domain candidate)
  - for each domain, find values (BERT + GloVe span selection)



label	slot action
0	irrelevant (value is ***none***)
1	request (value is ***?***)
2	value is ***don't care***
3	value is ***yes***
4	value is ***no***
5	value resides in **user$_t$**
6	value resides in **system$_t$**
7	**refer** previous states

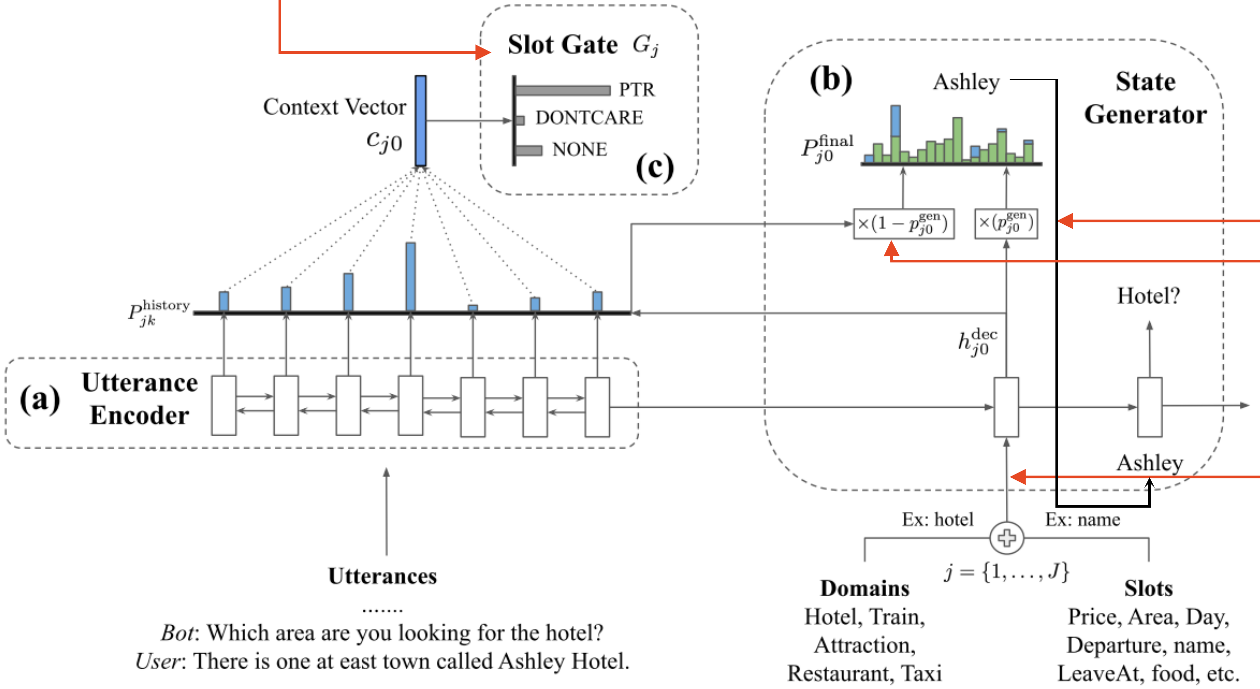
(Dey & Desarkar, 2021)
https://aclanthology.org/2021.sigdial-1.23

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

takes concatenated dialogue history



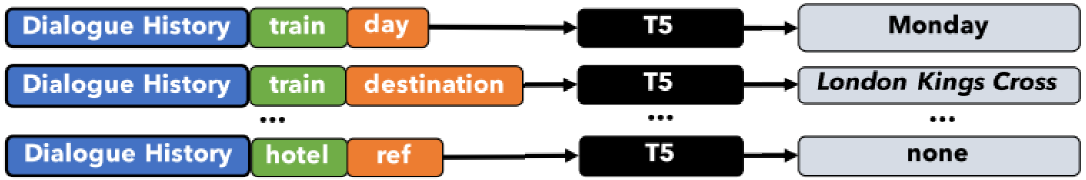
pointer-generator net (see NLU lecture): can **generate** tokens from vocabulary or **copy** tokens from attention

specific start token for each slot (& domain)

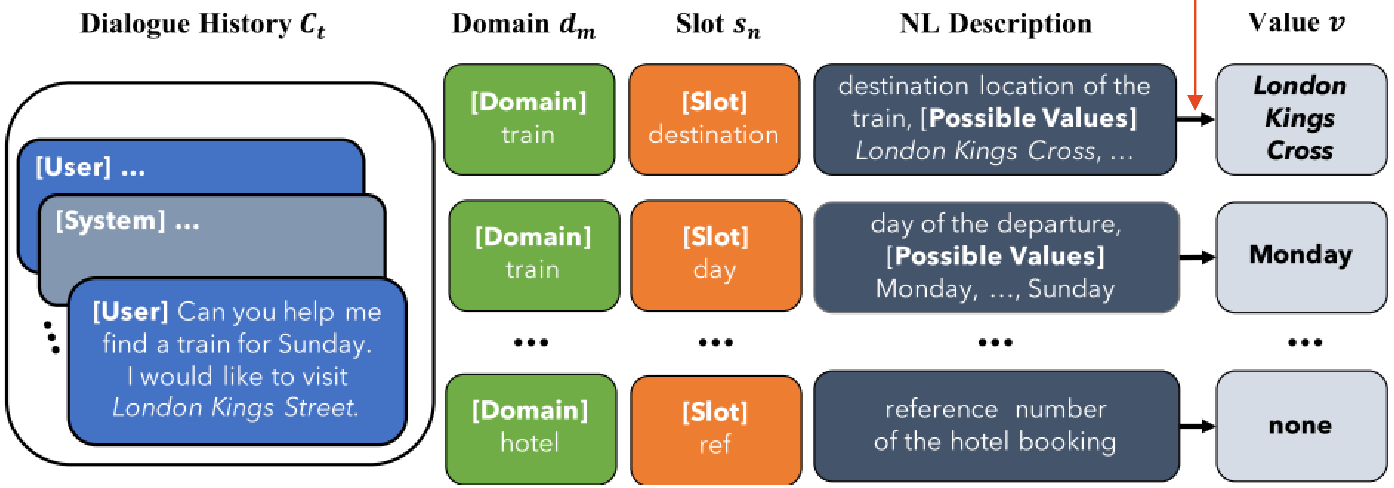
Generator + Pretrained LMs

(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



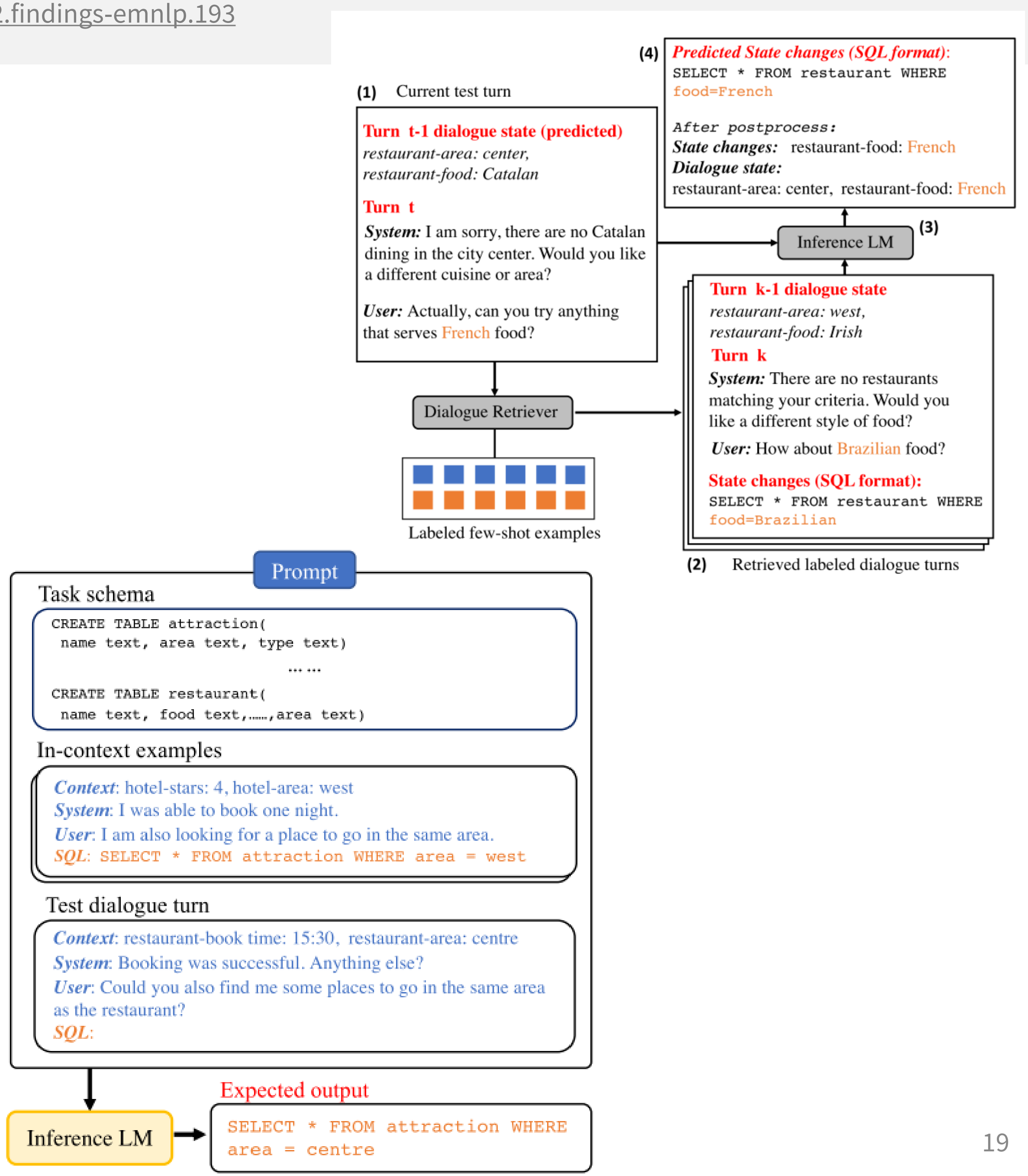
passed through T5



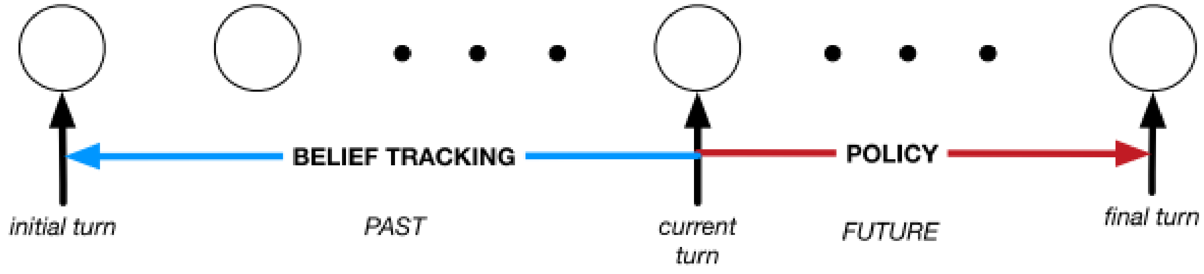
LLM Prompting

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

- Prompt LLM to produce state
  - this work: GPT-Neo, CodeGen, GPT-3
- Needs context
  - DB schema shown in SQL
  - Dialogue context: prev. state + 1 turn
  - Retrieved few-shot examples
    - SBERT similarity
- Needs framing
  - State changes ~ SQL
- Works well in few-shot settings
  - Needs less data for retrieval (~1-5%/100-500 dialogues works already)



Action Selection / Policy



(from Milica Gašić's slides)

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

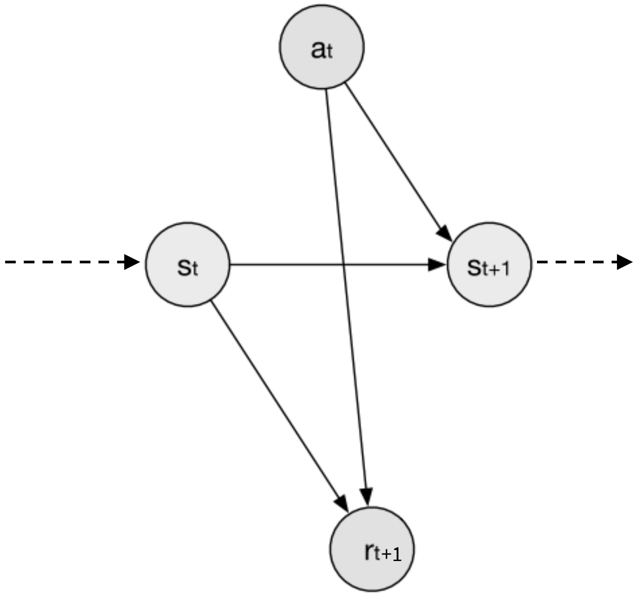
Why Reinforcement Learning

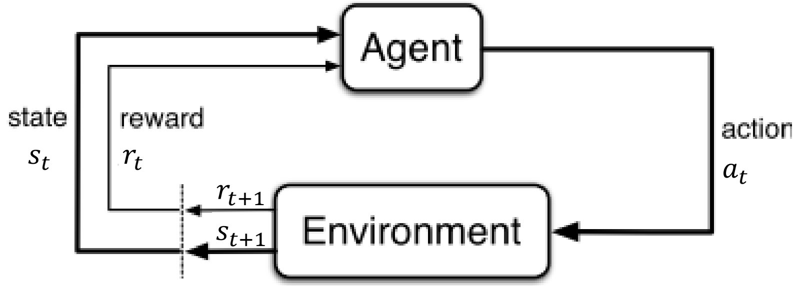
- **Action selection ~ classification** → use supervised learning?
  - set of possible actions is known
  - belief state should provide all necessary features
- Yes, but…
  - You'd **need** sufficiently large **human-human data** – hard to get
    - human-machine would just mimic the original system
  - 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** $s_t \in \mathcal{S}$ (~ dialogue state)
    - takes **actions** $a_t \in \mathcal{A}$ (~ 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

(from Milica Gašić's slides)

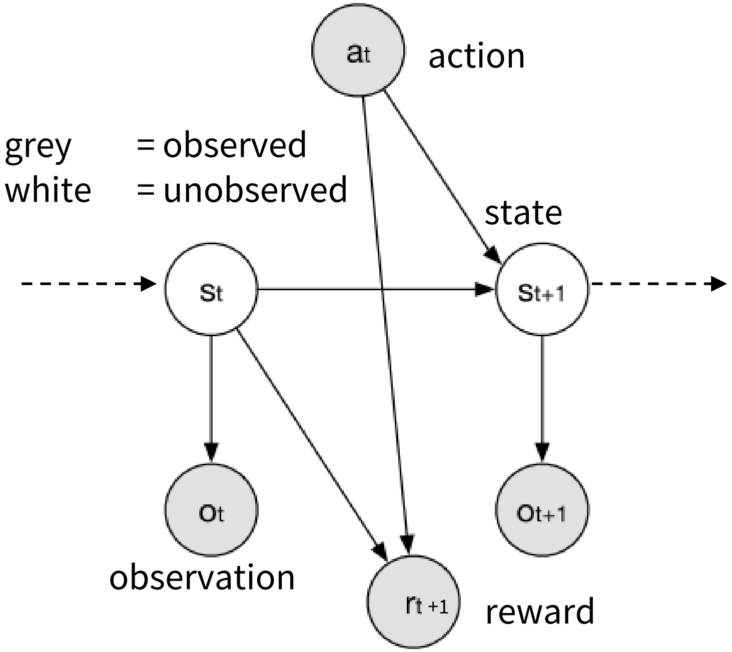




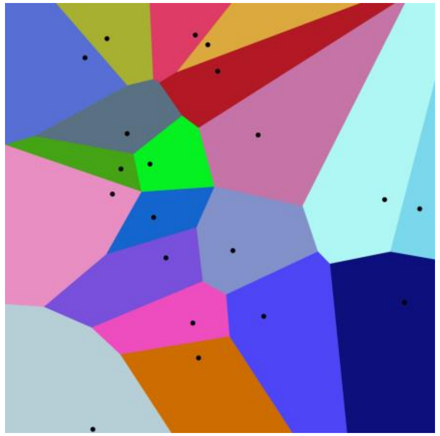
(Sutton & Barto, 2018)

Partially-observable MDPs

(from Milica Gašić's slides)



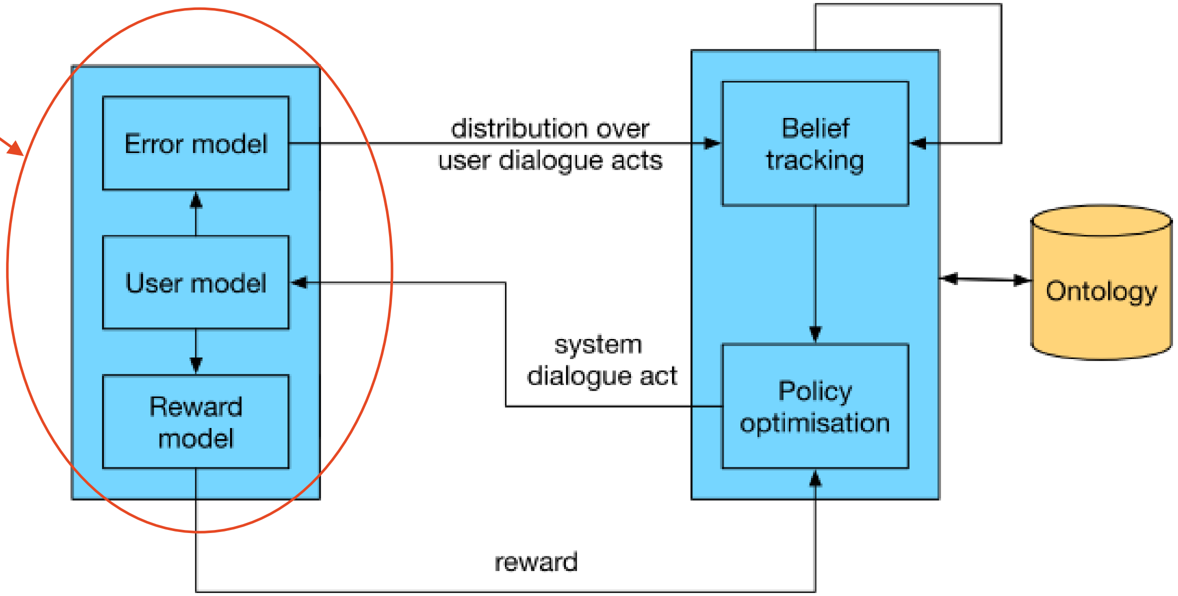
- POMDPs – **belief** states instead of dialogue states
  - true states ("what the user wants") are not observable
  - observations ("what the system hears") depend on states
  - belief – probability distribution over states
  - can be viewed as **MDPs with continuous-space states**
    - just represent 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



https://en.wikipedia.org/wiki/Voronoi_diagram

Simulated Users

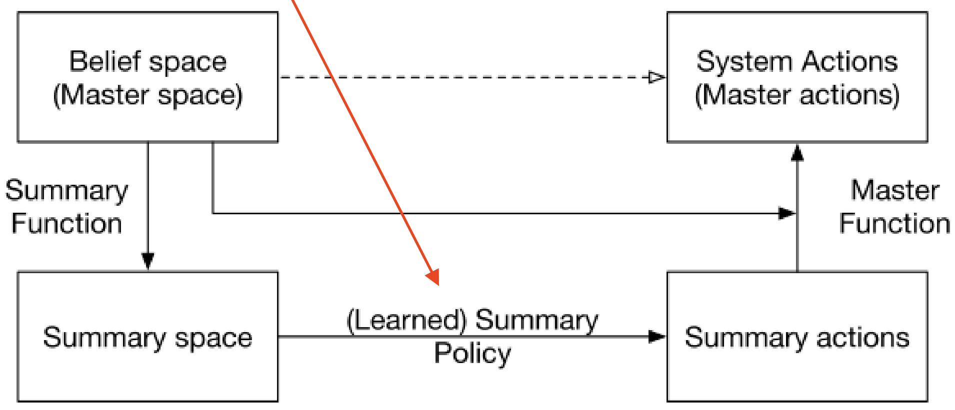
- Static datasets aren't enough for RL
  - data might not reflect our newly learned behaviour
- RL needs a lot of data, more than real people would handle
  - 1k-100k's dialogues used for training, depending on method
- solution: **user simulation**
  - basically another DS/DM
  - (typically) working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



(from Milica Gašić's slides)

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

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

accumulated long-term reward (from turn t onwards)

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

$\gamma \in [0,1]$ = **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 → maximize **expected return**

$$\mathbb{E}[R_t | \pi, s_0]$$

expected R_t if we start from state s_0 and follow policy π

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
Skype/Meet/Zoom/Troja (by agreement)

Labs in 10 minutes
Finetuning GPT-2

Next Tue 10:40
rest of Dialogue Policy

Get these slides here:

http://ufal.cz/npfl099

References/Inspiration/Further:

- Filip Jurčíček'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