NPFL099 Statistical Dialogue Systems
9. End-to-end Task-Oriented Systems

Ondřej Dušek

http://ufal.cz/npfl099

1. 12. 2025

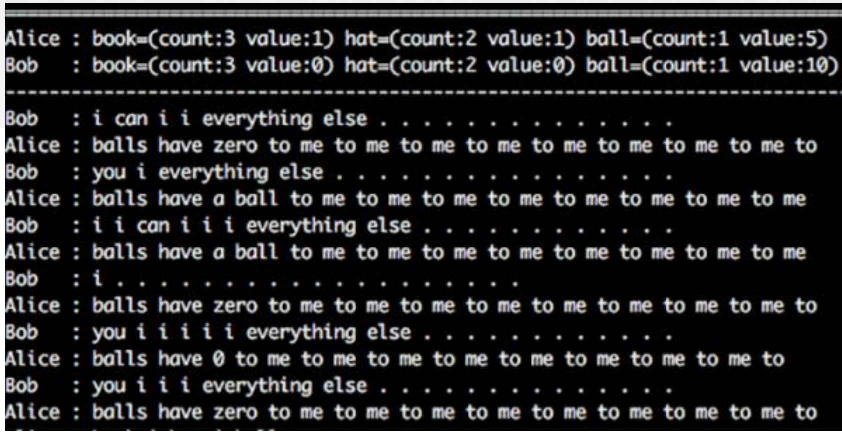
Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

End-to-end dialogue systems

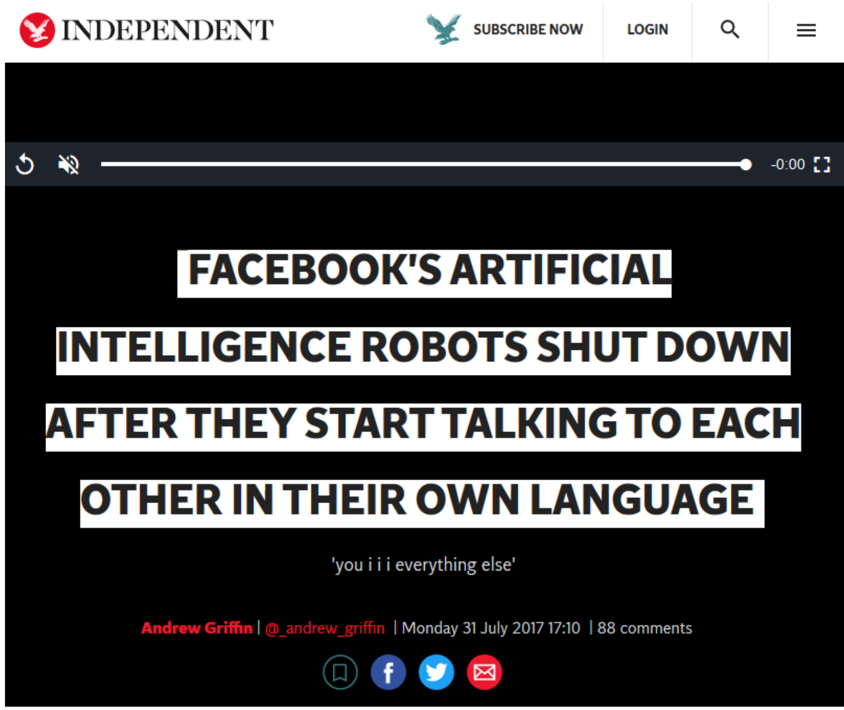
- **End-to-end = represent the whole system as one neural net**
  - sometimes, just some of the components can be joined
    - e.g. just NLU + tracker + policy, NLG excluded
- Pros & cons:
  - Traditional architecture – separate components:
    - more flexible (replace one, keep the rest)
    - error accumulation
    - improved components don’t mean improved system
    - possibly joint optimization by RL
    - explainability
  - End-to-end:
    - joint optimization by backprop
      - if fully differentiable
    - still can work via RL (with supervised initialization)
    - architectures still decompose into (some of) original DS components
      - and often still need DA-level annotation

Training end-to-end systems

- Supervised
  - sometimes components still trained separately
    - e.g. hard knowledge base lookup
  - sometimes all in one
  - can't learn from users
  - problems with train-test mismatch
- RL
  - can learn from users, can learn all-in-one
  - doesn't work great if done on word-level
    - RL won't care about fluency/naturalness
      if you only reward task accuracy
    - → avoid word level RL / use fluency rewards / mix in supervised
- Less of a concern with PLMs/LLMs



https://towardsdatascience.com/the-truth-behind-**facebook**-ai-inventing-a-new-language-37c5d680e5a7



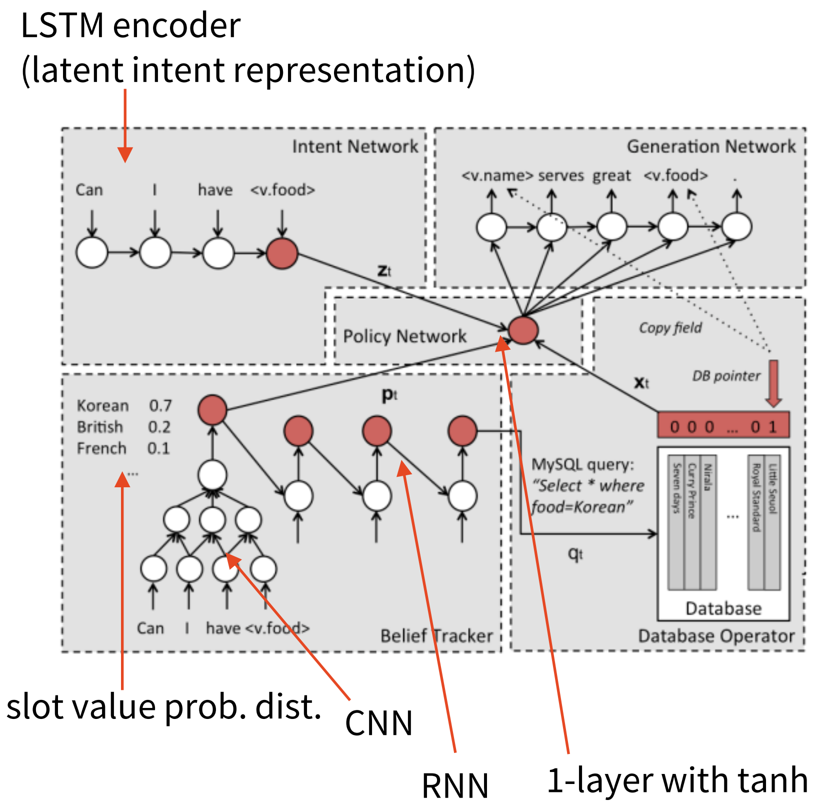
Facebook abandoned an experiment after two artificially intelligent programs appeared to be chatting to each other in a strange language only they understood.

https://www.independent.co.uk/life-style/gadgets-and-tech/news/facebook-artificial-intelligence-ai-chatbot-new-language-research-openai-google-a7869706.html

Supervised with component nets

(Wen et al., 2017)
https://www.aclweb.org/anthology/E17-1042

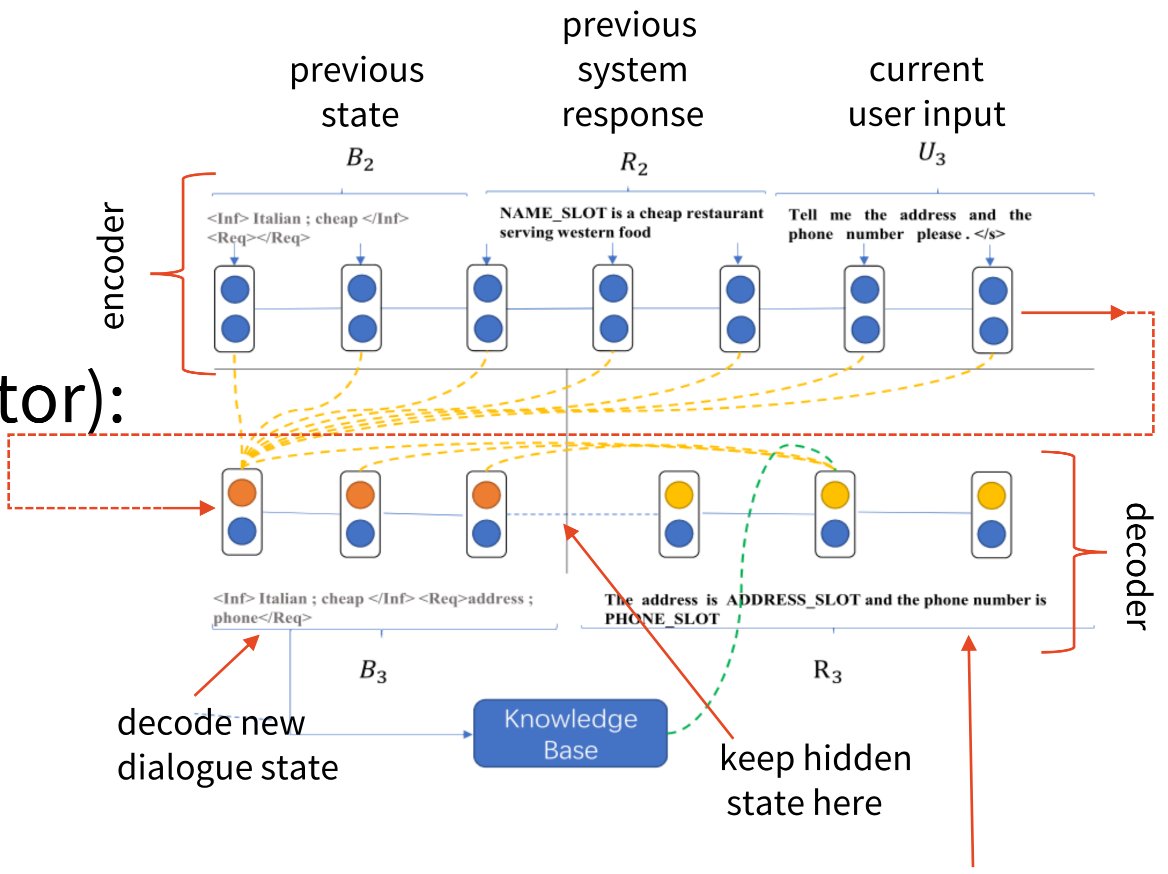
- "seq2seq augmented with history (tracker) & DB"
- end-to-end, but has components
    - LSTM **"intent network"**/encoder (latent intents)
    - CNN+RNN **belief tracker** (prob. dist. over slot values)
        - lexicalized + delexicalized CNN features
        - turn-level RNN (output is used in next turn hidden state)
        - trained separately from the rest of the system
    - **DB**: rule-based, takes most probable belief values
        - boolean vector of selected items
        - compressed to 6-bin 1-hot (no match, 1 match… >5 matches)
        - 1 matching item chosen at random & kept for lexicalization
    - Feed-forward **policy** (latent action)
    - LSTM **generator**
        - conditioned on policy, outputs delexicalized (lexicalization as post-processing)



Sequicity: Two-stage Copy Net – fully seq2seq-based

RNN + copy | seq gen

- less hierarchy, simpler architecture
  - no explicit system action – direct to words
  - still explicit dialogue state
  - KB is external (as in most systems)
- seq2seq (LSTM) + copy (pointer-generator):
  - **encode**: previous dialogue state
    + prev. system response
    + current user input
  - **decode new state** first
    - attend over whole encoder
  - **decode system output** (delexicalized)
    - attend over state only
      + use KB (one-hot vector added to each generator input)
      - KB: 0/1/more results – vector of length 3



(Lei et al., 2018) https://www.aclweb.org/anthology/P18-1133

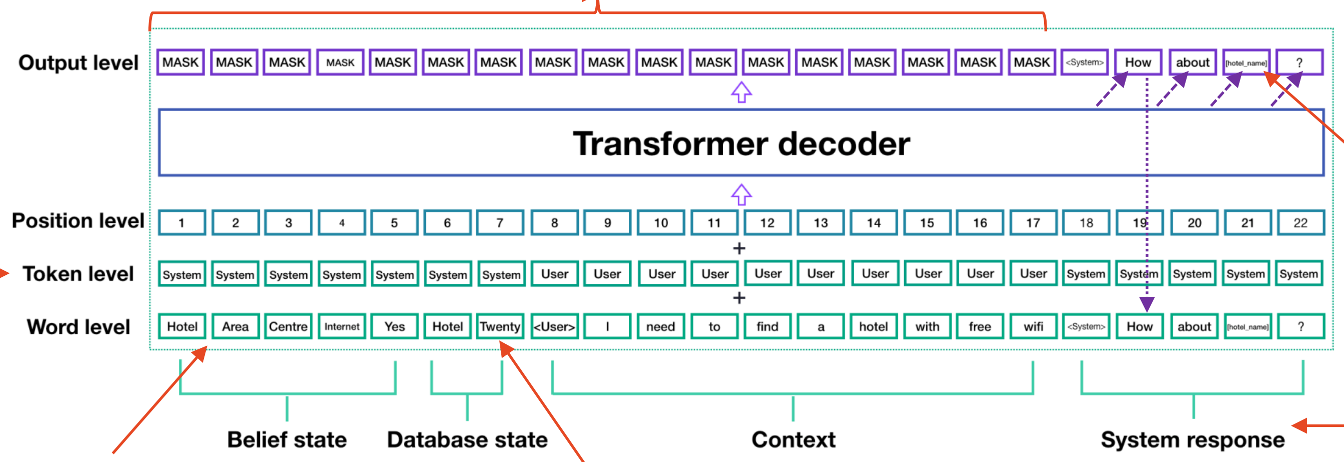
“Hello, it’s GPT-2 – How can I help?”

(Budzianowski & Vulić, 2019)
https://www.aclweb.org/anthology/D19-5602

- Simple adaptation of the GPT-2 pretrained LM
  - only model change: system/user embeddings
    - added to Transformer positional embs. & word embs.
  - GPT-2 is decoder-only: encoding/prompting = force-decoding
  - training to generate + classify utterances (good vs. random), all supervised
- no DB & belief tracking – gold-standard belief & DB used, no updates (see → →)

“encoding”
(force-decoding)
- our inputs fed in
- outputs ignored

system/user
embeddings



delexicalized generation
(autoregressive)

this is the actually decoded part

simple encoding:
domain-slot-value[-slot-value…]

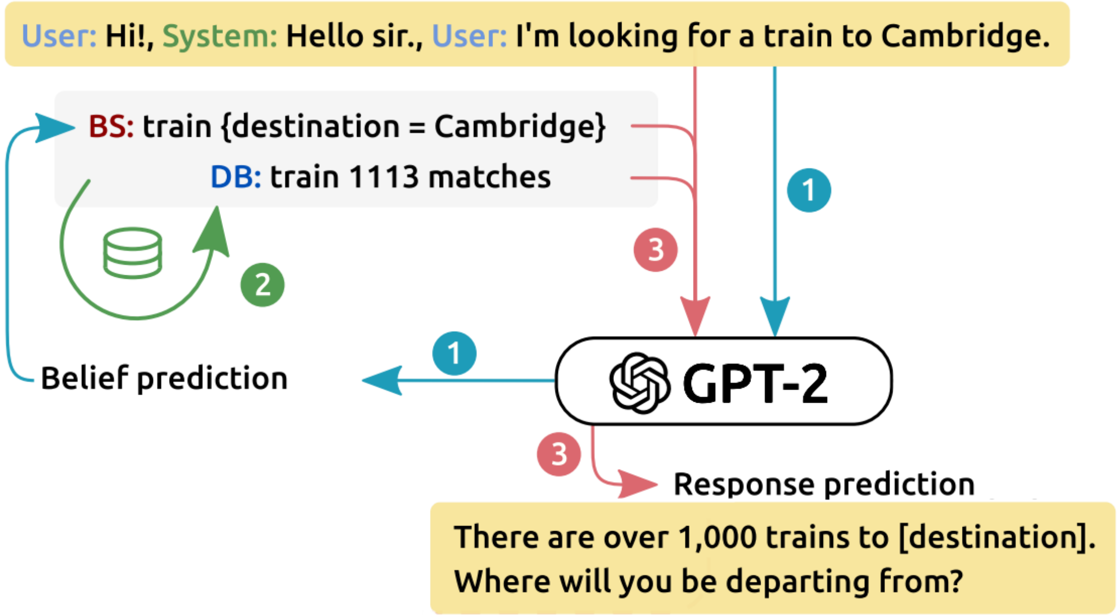
DB result entry tokens

Real stuff with GPT-2: SimpleTOD, NeuralPipeline, UBAR SOLOIST, AuGPT

=force-decode (ignore softmax, feed own tokens)

- Sequicity + GPT-2:
  1. encode context & decode belief state
  2. query DB
  3. encode DB results & decode response
  - history, state, DB results, system action – all recast as sequence
  - finetuning on dialogue datasets

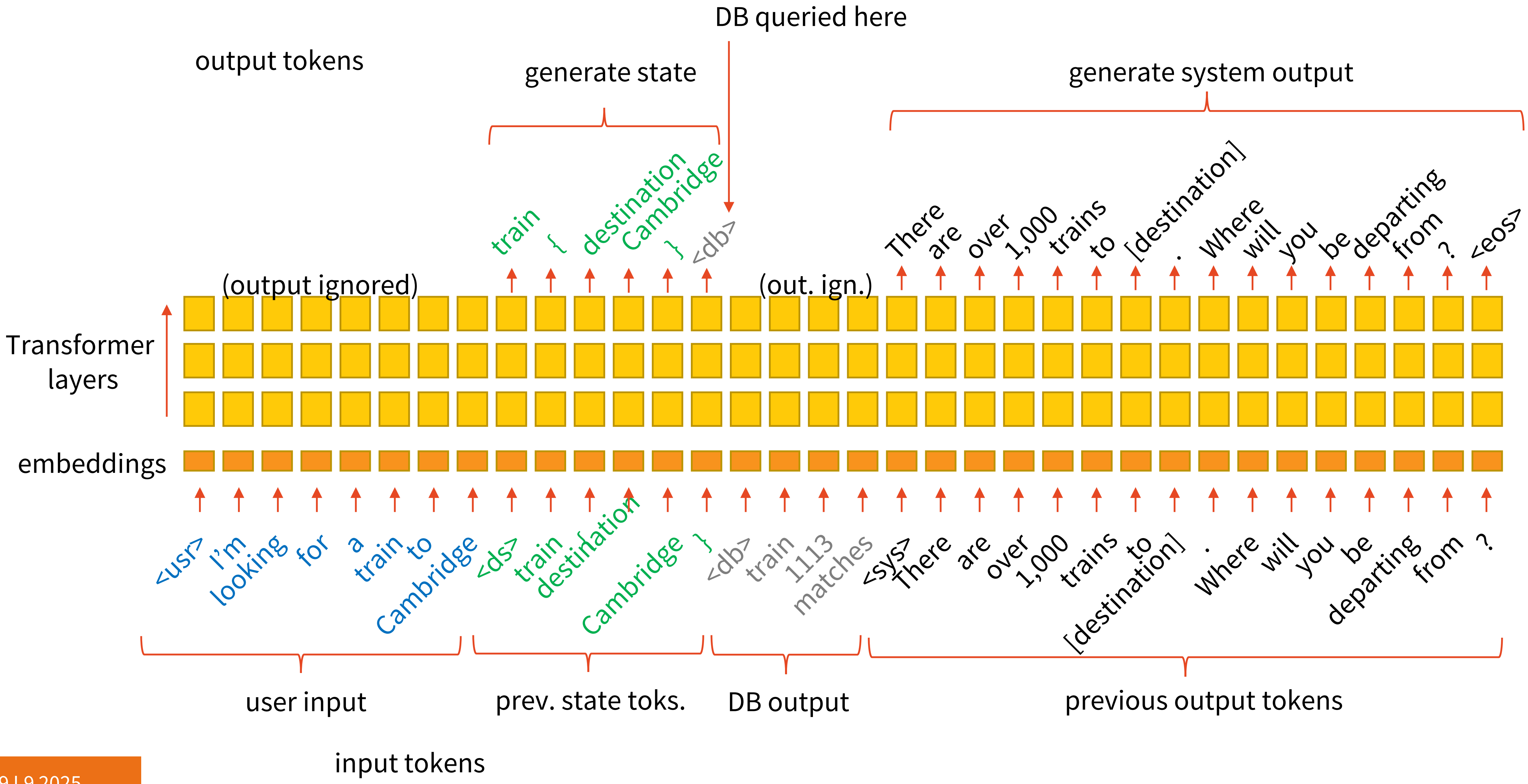
(Kulhánek et al., 2021) http://arxiv.org/abs/2102.05126



- extensions:
  - specific user/system embeddings (NeuralPipeline)
  - multi-task training: detect fake vs. real belief/response (SOLOIST, AuGPT)
  - decode explicit system actions (SimpleTOD, UBAR)
  - context includes dialogue states (UBAR)
  - data augmentation via backtranslation (AuGPT)

(Peng et al., 2021) https://aclanthology.org/2021.tacl-1.49/
(Yang et al., 2021) http://arxiv.org/abs/2012.03539
(Hosseini-Asl et al., 2020) http://arxiv.org/abs/2005.00796
(Ham et al., 2020) https://aclanthology.org/2020.acl-main.54/

GPT-2 two-stage decoding example



SOLOIST/AuGPT: Consistency task

- **Additional training task** – generating & classifying at the same time
  - additional classification layer on top of last decoder step logits
  - incurs additional loss, added to generation loss
- Aim: **robustness** – detecting problems
  - **½ data artificially corrupted** – state or target response don't fit context
  - SOLOIST: corrupted state sampled randomly
  - AuGPT: corrupted state sampled from the same domain (harder)

context	state	**response**	**consistent?**	
i want a cheap italian restaurant	*{ price range = cheap , food = Italian }*	ok which area ?	✅	
i want a cheap Italian restaurant	*{ price range = cheap , food = Italian }*	thanks, goodbye !	❎	bad response
i want a cheap italian restaurant	*{ destination = Cambridge , leave at = 19:00 }*	ok which area ?	❎	bad state
i want a cheap italian restaurant	*{ area = north , food = Chinese }*	ok which area ?	❎	bad state (same domain)

(Peng et al., 2021) https://aclanthology.org/2021.tacl-1.49/
(Kulhánek et al., 2021) http://arxiv.org/abs/2102.05126

MinTL: Diff dialogue states

(Lin et al., 2020)
https://aclanthology.org/2020.emnlp-main.273/

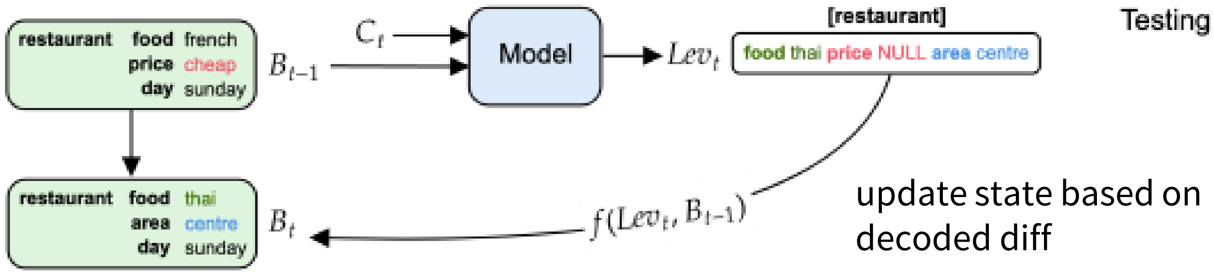
pre-LM | seq gen

- 2-step decoding, same as ↑
  - based on T5 or BART here
  - explicit 2 decoders
    (for state, for response)
- “Levenshtein states”
  - don’t decode full state each turn
  - **decode just a diff**
    (“Levenshtein distance from previous”)
    (a.k.a. NLU + rule 😇)
  - better consistency over dialogue

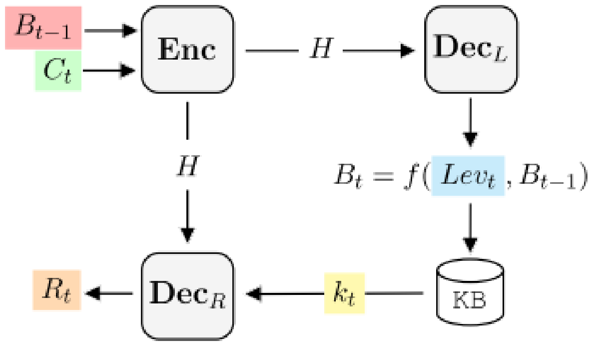
obtain diffs from state annotation

encode previous state & context

decode diffs



update state based on decoded diff



[hotel] stars 5 area centre day sunday [restaurant] food thai area centre day sunday name bangkok city <EOB> Can you help me book a 5 star hotel near the restaurant on the same day? <EOU>For how many people? <EOR>10 people <EOU>

<SOB>[hotel] people 10 <EOB>

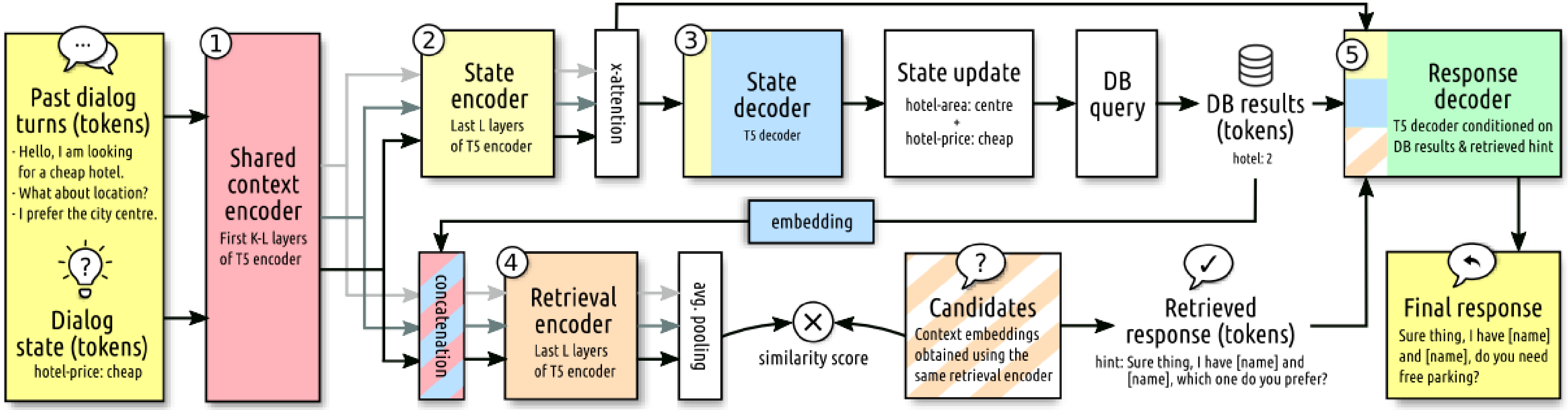
<KB2> sorry, there are no matches. would you like to try another part of town? <EOR>

DB queried based on updated state
response decoder starting token = # of DB results

Retrieval-augmented generation

(Nekvinda & Dušek, 2022)
https://aclanthology.org/2022.sigdial-1.29

- Same previous, but use examples for inspiration
  - retrieve similar example from training data & pass it to response decoder as a "**hint**"
  - α-blending: with prob. α, replace hint with true response to promote copying
- Example retrieval based on system action annotation
  - positive examples: same action, negative: different actions
- Joint model for example retrieval & state + response decoding
  - T5 with 2 decoders (state vs. response) + duplicate last 2 encoder layers for retrieval



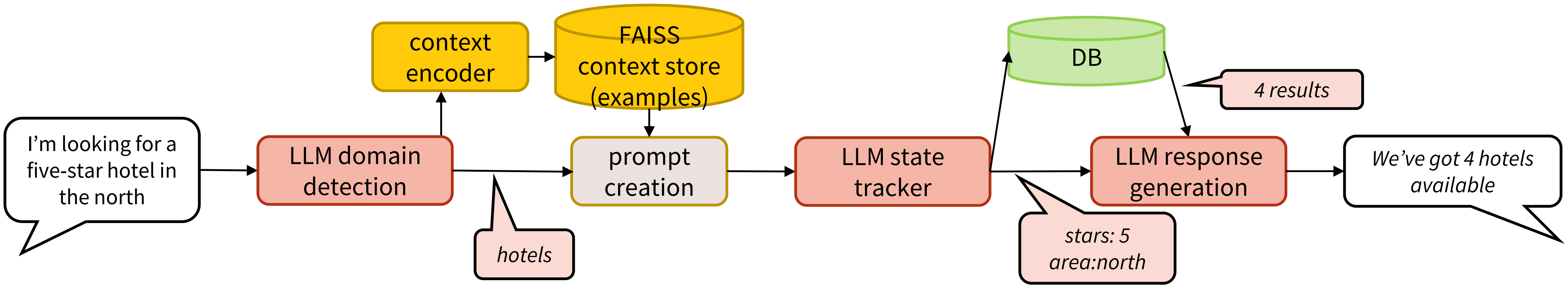
LLM-based dialogue

(Hudeček & Dušek, 2023)
https://aclanthology.org/2023.sigdial-1.21

- "Sequicity but with LLM prompting"
  - same idea: **context** → **state** → **DB** → **response**
    - state tracking & response generation done with LLMs
  - additional LLM step needed: domain detection
    - tracking & response prompts use domain descriptions
  - not entirely "end-to-end" – same LLM, multiple runs
- Zero-shot/few-shot (opt. ~10 ex./domain + retrieval)
- Works, but worse than finetuning (esp. on state tracking)
  - not that bad with better LLMs & if debugged properly 😇

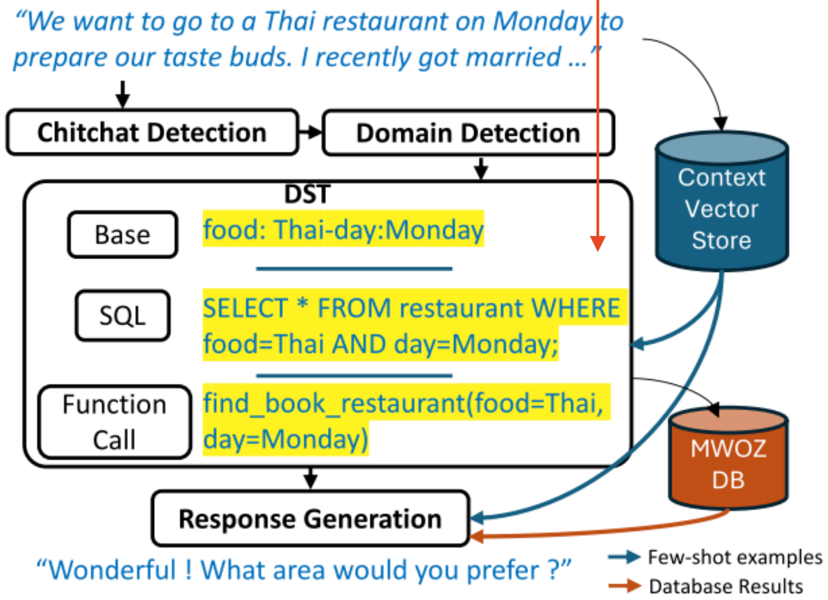
instruction	Definition: Capture values from a conversation about hotels. Capture pairs "entity:value" separated by colon and no spaces in between. Separate the "entity:value" pairs by hyphens. Values that should be captured are:
domain description	- "pricerange": the price of the hotel - "area": the location of the hotel …
examples	--- Example 1 --- … ---
dial. history	Assistant: "Hello, how can I help you?" …
user input	Customer: "I am looking for a five-star hotel in the north"

(Steindl et al., 2025)
https://aclanthology.org/2025.findings-emnlp.610/

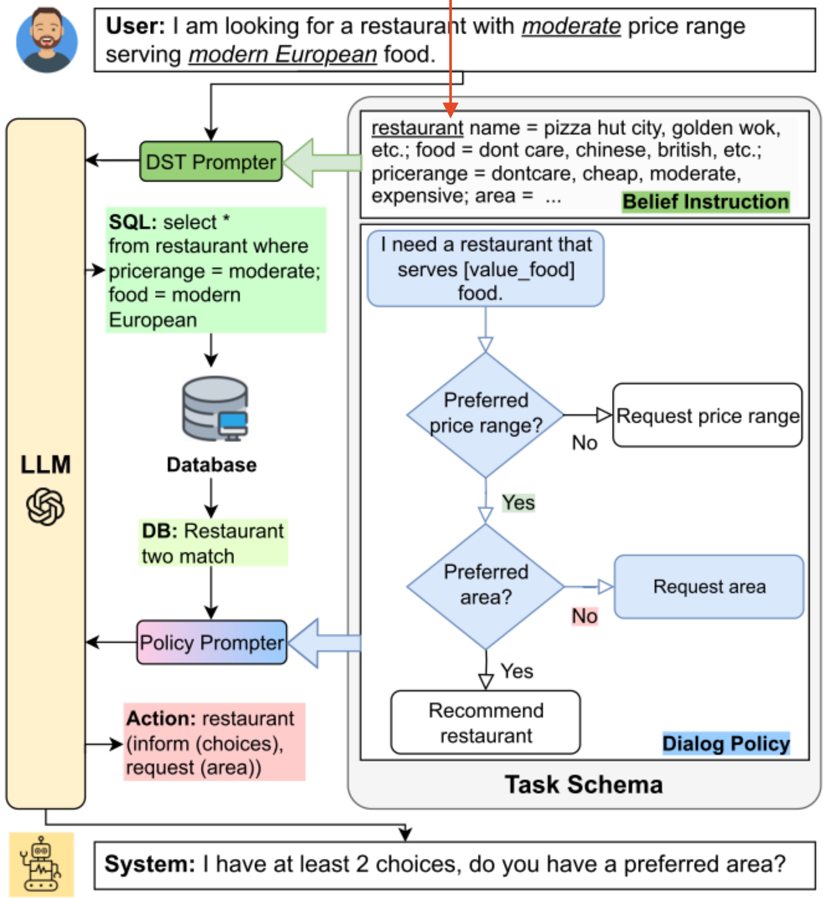


LLM-based dialogue, better

- You can extend ↑ to make it work better:
  - Adding "policy skeletons" (=dialogue snippet examples to show behavior)
  - Changing the state representation & using code generation + supporting chitchat



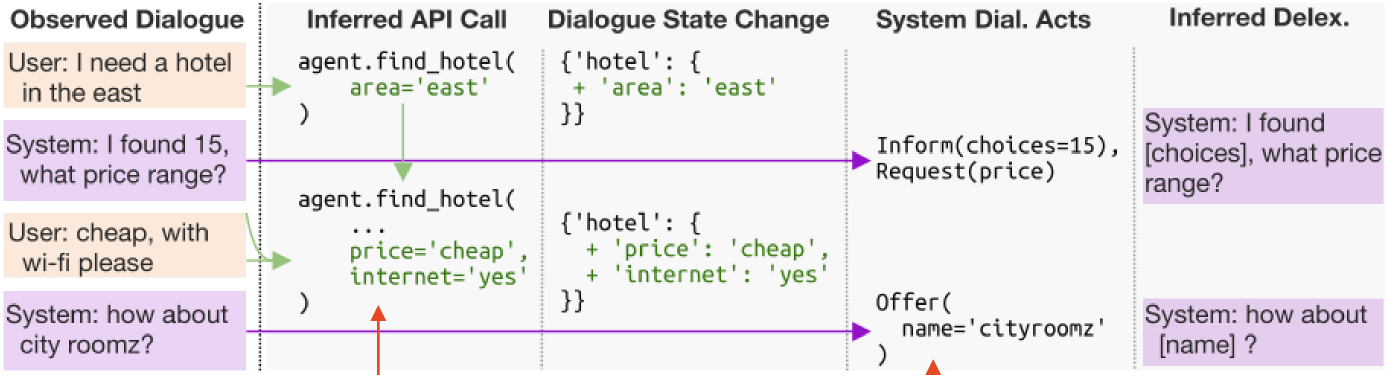
(Stricker & Paroubek, 2024)
https://aclanthology.org/2024.sigdial-1.50



(Zhang et al., 2023)
https://aclanthology.org/2023.findings-emnlp.891

LLM based dialogue, with more data

- You can use existing dialogues & additional data to improve
  - generate annotation via code LLM + finetune
  - use LLMs for unstructured queries (if e.g. FAQ page exists)
    - SQL + “answer” operator for any question answering, standard retrieval + LLM processing

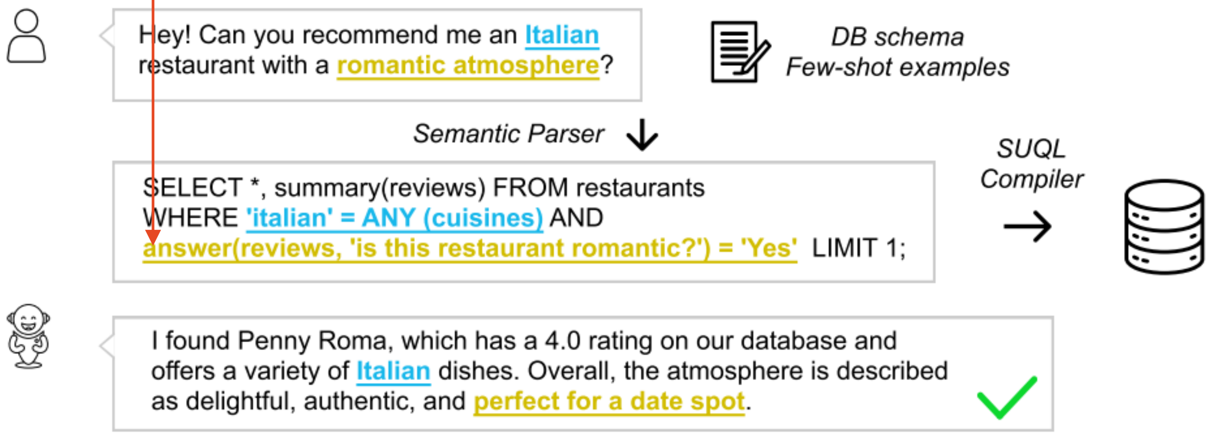


prompt LLM to predict dialogue state as API call

prompt LLM to extract system dialogue acts

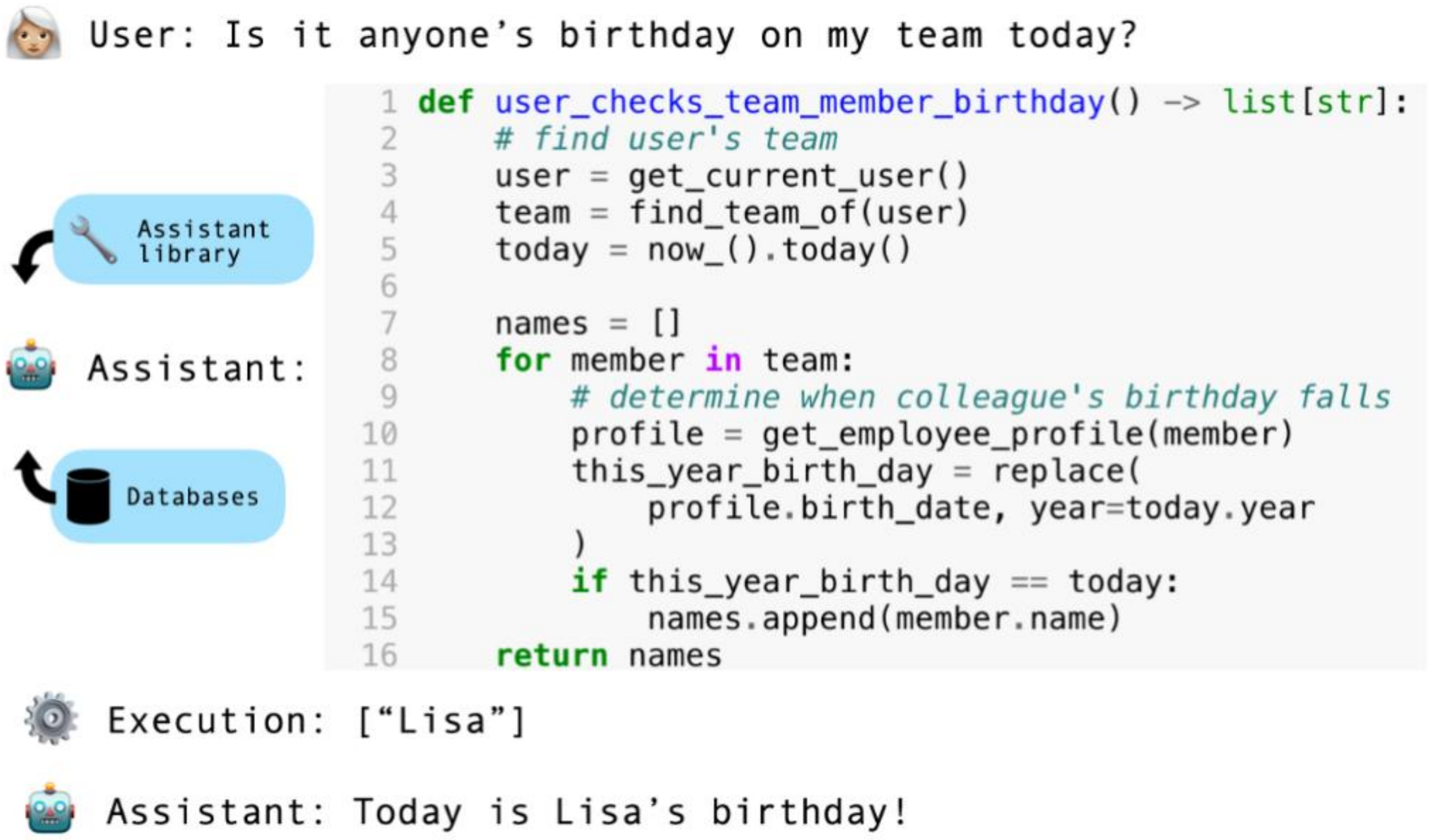
(King & Flanigan, 2024)
http://arxiv.org/abs/2404.15219

(Liu et al., 2024)
https://aclanthology.org/2024.findings-naacl.283



LLM based dialogue, beyond slots

- LLMs asked to reason with given API functions
- 1 question needs more than 1 API call
- LLMs generate code, executed in a simulated environment
- So far very experimental, only reasoning LLMs work

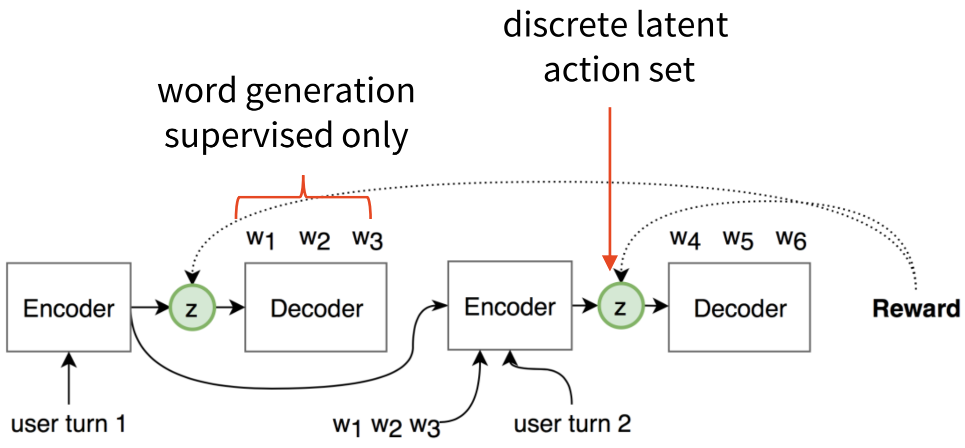


(Coca et al., 2025) https://aclanthology.org/2025.acl-long.1234

Latent Action RL

(Zhao et al., 2019)
https://www.aclweb.org/anthology/N19-1123

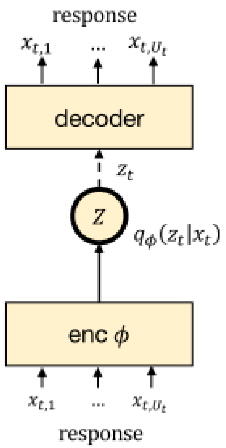
- Make system actions latent, learn them implicitly w/o annotation
- Like a VAE, but **discrete latent space** here (M k-way variables)
  - using Gumbel-Softmax trick for backpropagation
- RL over latent actions, not words
  - avoids producing disfluent language
  - **corpus-based RL** – "faking it " on supervised data
    - generate outputs, but use original contexts from a dialogue from training data
    - success & RL updates based on generated responses
  - interleaves with supervised to learn word generation
- Ignores DB & belief tracking
  - takes gold annotation from data (assumes external model for this)

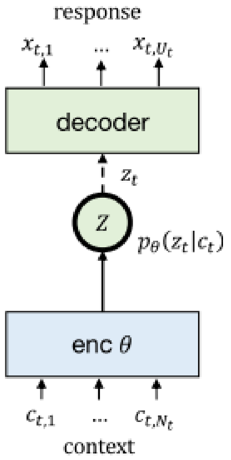


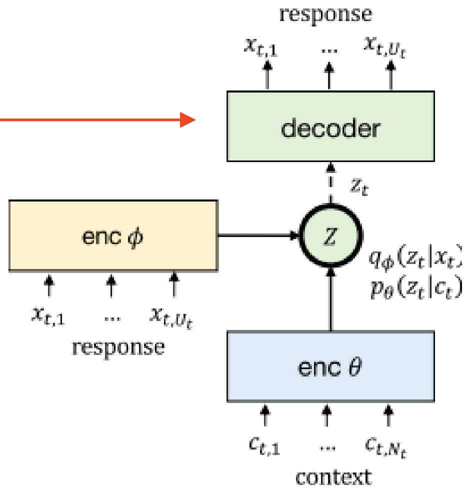
LAVA: Latent Actions with VAE pretraining

(Lubis et al., 2020)
https://aclanthology.org/2020.coling-main.41/

- Also **discrete latent space** for **actions**
  - inputs & responses should be close in latent space
- Multi-step training scenario:
  1) **autoencode** responses into latent space
  2) **supervised** training for response generation via the latent space
  3) **RL** over the latent actions
     - same "fake RL" as previous
- Options to join autoencoding & response generation
  a) KL loss – don't go too far from autoencoding in latent space
  b) multi-task training (go back to autoencoding once in a while)
- Again, assumes gold state & DB

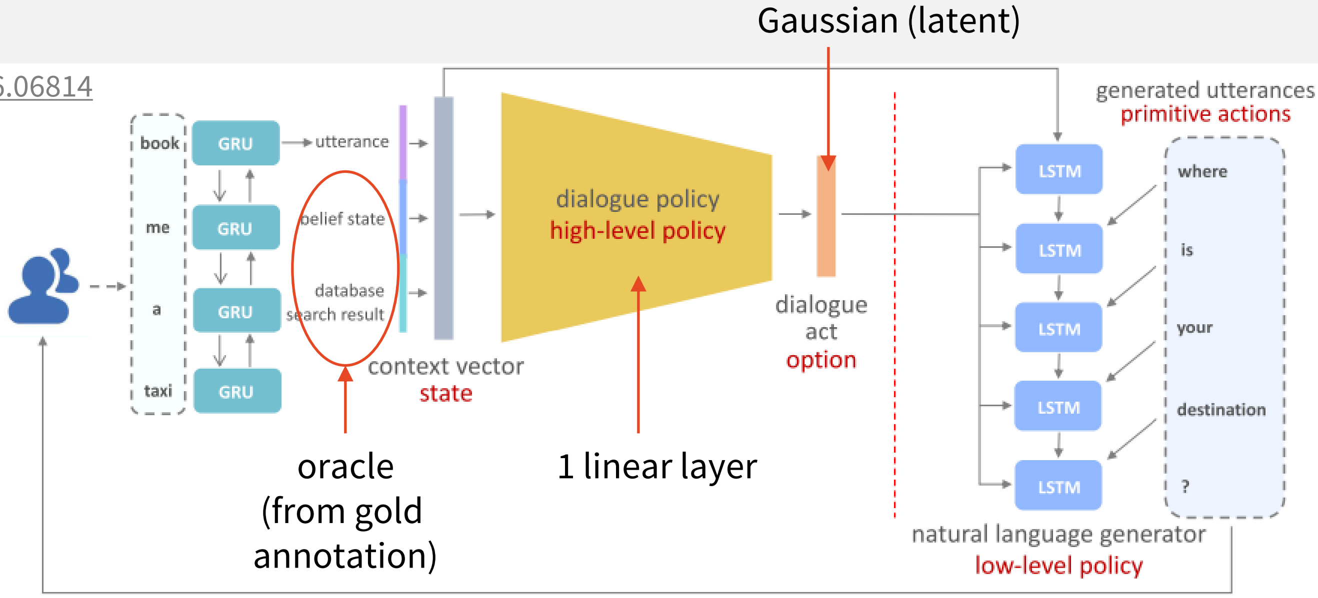






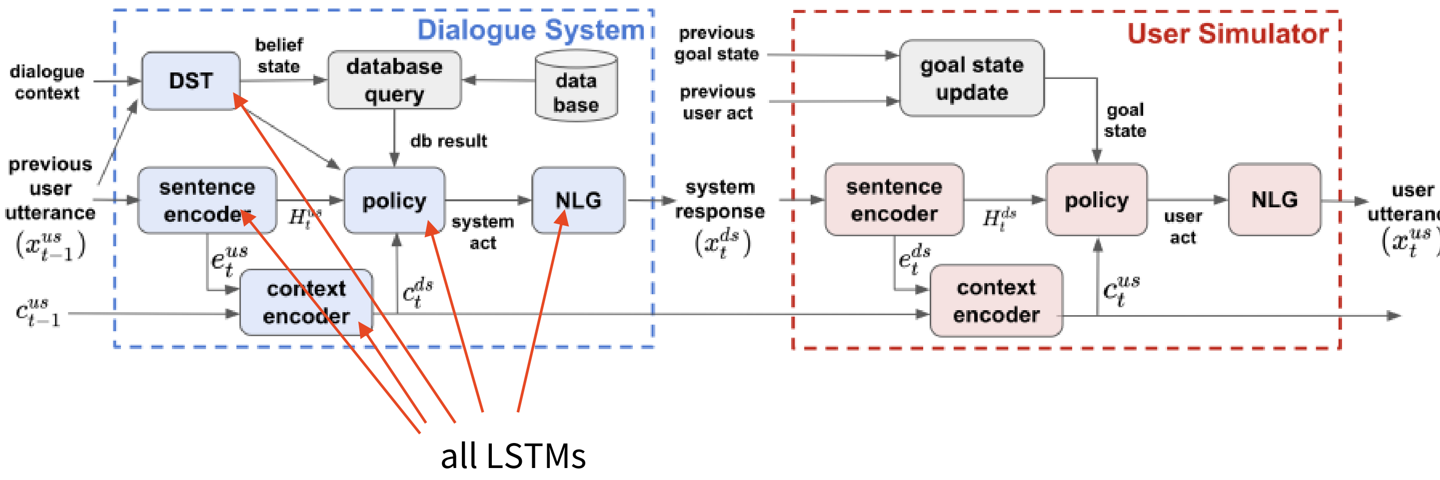
Better RL: HDNO & JOUST

(Wang et al., 2021) http://arxiv.org/abs/2006.06814



- **HDNO**: 2-level hierarchical RL
  - top level: (latent) actions
    bottom level: words
  - LM rewards on word level (for fluency)
  - separate updates on both levels
    (avoid aiming at a moving target)
  - "fake" corpus-based RL (as previous)
- **JOUST**: real RL with a user simulator
  - system & sim. share architecture
    - joint context encoder
  - system: additional state tracker
  - interaction on utterance level
  - supervised pretraining

(Tseng et al., 2020) https://aclanthology.org/2021.acl-long.13

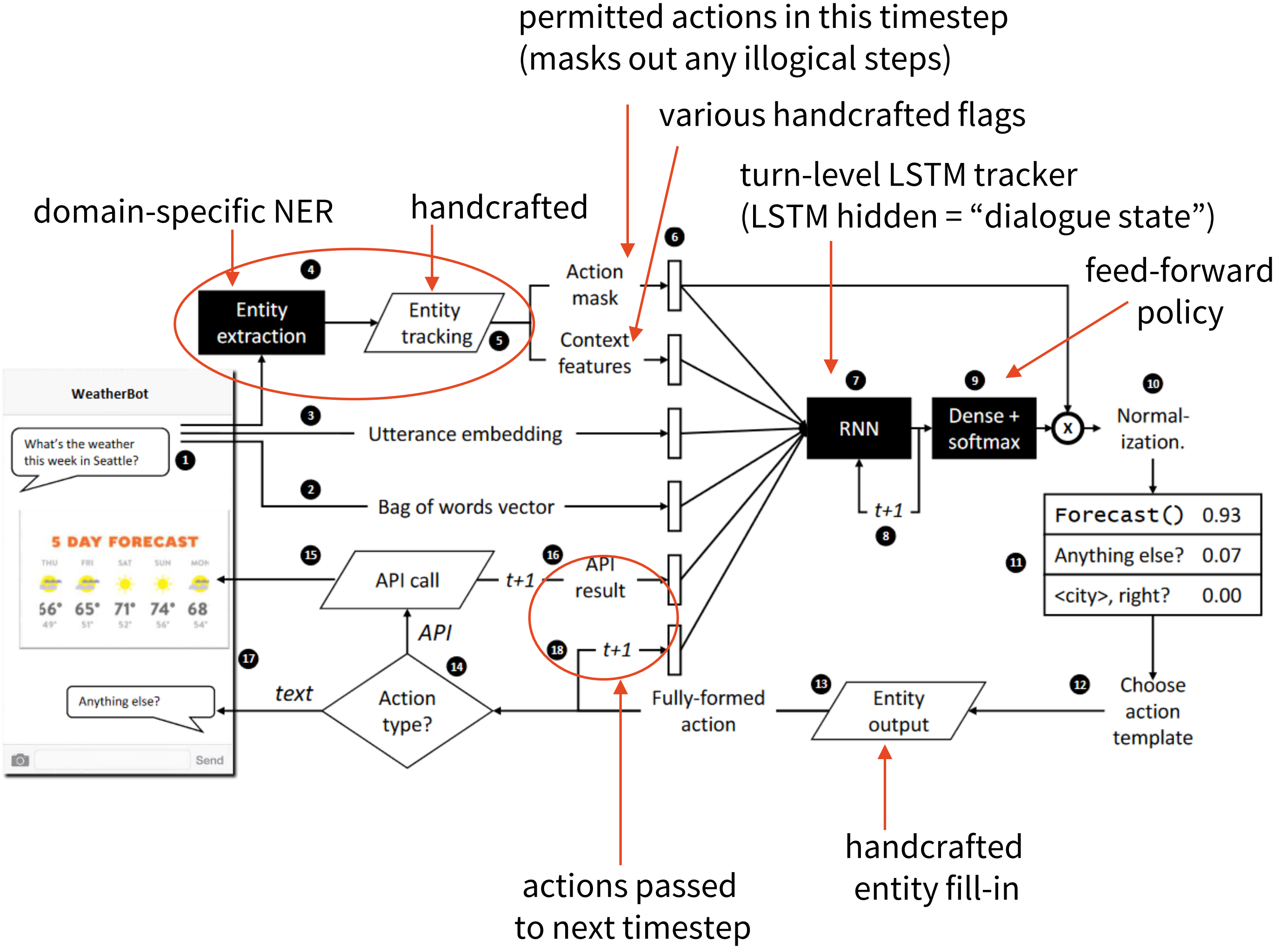


Hybrid Code Networks

(Williams et al., 2017)
http://arxiv.org/abs/1702.03274

RNN + FC + rule | classif

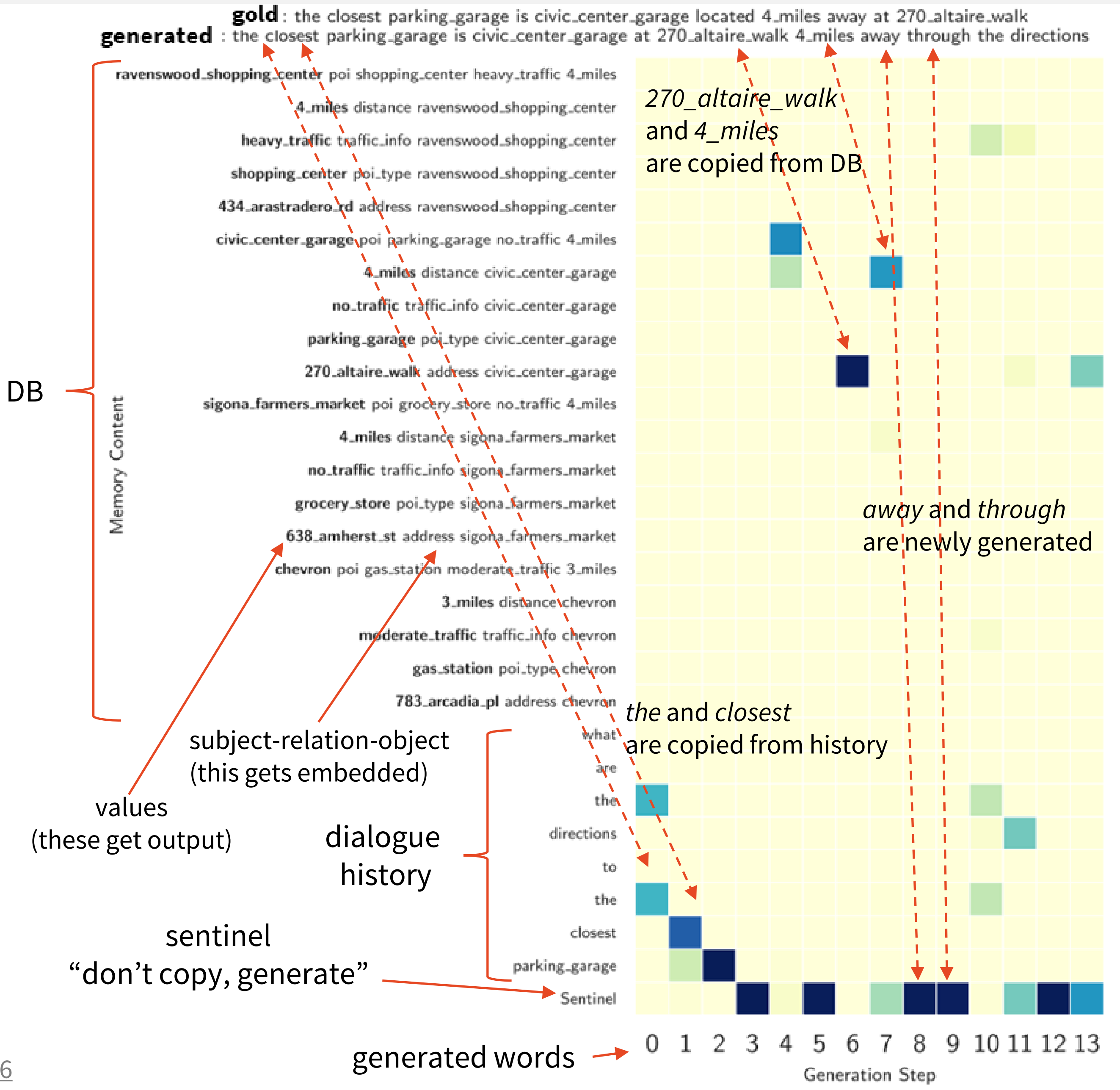
- Partially handcrafted
- Learn from ~30 dialogues (RL+SL)
- LSTM tracker
  - state implicit (=no annotation)
- Policy & tracker use action mask
  - handcrafted from entity tracker
  - zero illegal actions
  - e.g. don't place a call if we don't know who to call yet
- Delexicalized operation
  - entity tracking & fill-in handcrafted



(Shalyminov & Lee, 2018) https://arxiv.org/abs/1811.12148
(Marek, 2019) http://arxiv.org/abs/1907.12162

Mem2Seq: soft DB lookups

- Integrates the DB in the model
  - really “end-to-end”
  - works only if the DB is small enough
- Built on **memory networks**
  - multi-level attention-like model (old, complex, not so interesting)
  - combined with RNN
- Pointer-generation approach
  - “sentinel” (=generate)
  - point into the DB
  - point into history



(Madotto et al., 2018) https://www.aclweb.org/anthology/P18-1136

Summary

- End-to-end = single network for NLU/tracker + DM + NLG
  - joint training, may have distinct components & need dialogue state annotation
- Hybrid Code Nets – partially handcrafted, but end-to-end
- **Two-stage copy net** –2-step decoding: dialogue state, then response
  - Sequicity – LSTM seq2seq
  - GPT-2-based systems – same idea, just with pretrained LMs
  - **LLM-based:** code/SQL representations of state
- Discrete latent action space – learning w/o action annotation
- RL optimization
  - corpus-based "fake RL" on training data (no simulator needed)
  - without NLG (over actions) or hierarchical
- Mem2Seq: Soft DB lookups – making the whole system differentiable

Thanks

Contact us:

https://ufaldsg.slack.com/
odusek@ufal.mff.cuni.cz
Skype/Zoom/Troja (by agreement)

Labs in 10 mins

Get these slides here:

http://ufal.cz/npfl099

References/Inspiration/Further:

- Gao et al. (2019): Neural Approaches to Conversational AI: https://arxiv.org/abs/1809.08267
- Serban et al. (2018): A Survey of Available Corpora For Building Data-Driven Dialogue Systems: http://dad.uni-bielefeld.de/index.php/dad/article/view/3690