NPFL099 Statistical Dialogue Systems

9. End-to-end Task-Oriented Systems

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

http://ufal.cz/npfl099

28. 11. 2023
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
Supervised with component nets

- “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)

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

- Partially handcrafted, designed for little training data:
  - With Alexa-type assistants in mind

- **Utterance representations:**
  - Bag-of-words binary vector
  - Average of word embeddings

- **Entity extraction & tracking**
  - Domain-specific NER
  - Handcrafted tracking
  - Returns **action mask**
    - Permitted actions in this step (e.g. can’t place a phone call if we don’t know who to call yet)
    - Return (optional) handcrafted **context features** (various flags)

- **LSTM state tracker** (output retained for next turn)
  - I.e. no explicit state tracking, doesn’t need state tracking annotation

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

- feed-forward **policy** – produces probability distribution over actions
  - mask applied to outputs & renormalized → choosing action = output template
- handcrafted fill-in for entities
  - takes features from ent. extraction
  - ~learned part is fully delexicalized
- **actions** may trigger API calls
  - APIs can return feats for next step
- training – supervised & RL:
  - SL: beats a rule-based system with just 30 training dialogues
  - RL: REINFORCE with baseline
  - RL & SL can be interleaved
- extensions: better input than binary & averaged embeddings

(Shalyminov & Lee, 2018)  
https://arxiv.org/abs/1811.12148  
(Marek, 2019)  
http://arxiv.org/abs/1907.12162
Sequicity: Two-stage Copy Net – fully seq2seq-based

- 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](https://www.aclweb.org/anthology/P18-1133)
• 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 = “force-decoding”
    • pass input through all layers but ignore the softmax next-token prediction, feed our own tokens
  • 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

pre-LM | seq gen

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

NPFL099 L9 2023
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
4. finetuning on dialogue datasets

- 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)


BS: train {destination = Cambridge}
DB: train 1113 matches

Belief prediction

Response prediction

There are over 1,000 trains to [destination]. Where will you be departing from?
GPT-2 two-stage decoding example

Transformer layers

embeddings

(output ignored)

user input

prev. state toks.

DB output

previous output tokens

input tokens

DB queried here

generate state

generate system output

(output ignored)

There are over 1,000 trains to [destination]. Where will you be departing from?

I'm looking for a train to Cambridge.

prev. state toks.

DB output

prev. output tokens

embeddings

prev. state toks.
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!

<table>
<thead>
<tr>
<th>context</th>
<th>state</th>
<th>response</th>
<th>consistent?</th>
</tr>
</thead>
<tbody>
<tr>
<td>i want a cheap Italian restaurant { price range = cheap, food = Italian }</td>
<td>ok which area?</td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>i want a cheap Italian restaurant { price range = cheap, food = Italian }</td>
<td>thanks, goodbye!</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>i want a cheap Italian restaurant { destination = Cambridge, leave at = 19:00 }</td>
<td>ok which area?</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>i want a cheap Italian restaurant { area = north, food = Chinese }</td>
<td>ok which area?</td>
<td></td>
<td>✗</td>
</tr>
</tbody>
</table>

- new in AuGPT

- bad response
- bad state
- bad state (same domain)
MinTL: Diff dialogue states

- 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 time
- **just decode a diff**
  (“Levenshtein distance from previous”)
- better consistency over dialogue

**Diagram**

- **Encode previous state & context**
- **Obtain diffs from state annotation**
- **Update state based on decoded diff**

**Example**

- **DB queried based on updated state**
- **Response decoder starting token = # of DB results**

---

(Lin et al., 2020)

Retrieval-augmented generation

- Same idea as previous, but use examples for inspiration
  - retrieve similar example from training data & pass it to response decoder as a “hint”
  - $\alpha$-blending: with prob. $\alpha$, 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

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

(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
  - Zero-shot/few-shot
    - optionally ~10 ex./domain in context store (FAISS)
  - Works, but worse than finetuning (esp. on state tracking)

---

**Example 1**

Assistant: “Hello, how can I help you?”

Customer: “I am looking for a five-star hotel in the north”
Few-shot dialogue generation

(Zhao & Eskenazi, 2018) [link]

- **Domain transfer:**
  - source domain training dialogues
  - target domain “seed responses” with annotation

- **Encoding all into latent space**
  - keeping response & annotation encoding close
  - keeping context & response encoding close
  - decoder loss + matching loss

- **Encoder:** HRE (hierarchical RNN)
- **Decoder:** copy RNN (with sentinel)
  - “copy unless attention points to sentinel” (see Mem2Seq)
- **DB queries & results treated as responses/inputs**
  - DB & user part of environment

**Diagram:**
- Responses and annotations encoded into latent space
- Encoder: HRE (hierarchical RNN)
- Decoder: copy RNN (with sentinel)
- Training on source domain
- Training on target domain

---

![Diagram](http://aclweb.org/anthology/W18-5001)
• Making system actions latent, learning them implicitly
• Like a VAE, but **discrete latent space** here \((M \ k\)-way variables)  
  • using Gumbel-Softmax trick for backpropagation  
  • using Full ELBO (KL vs. prior network)  
  or “Lite ELBO” (KL vs. uniform \(1/k\))

• 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

• ignores DB & belief tracking  
  • takes gold annotation from data (assumes external model for this)
LAVA: Latent Actions with VAE pretraining

• kinda combination of two previous
• **discrete latent space** for actions
• 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

#### 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

---


Memory networks

- not a full dialogue model, just ranker of candidate replies
- no explicit modules
- based on attention over history
  - sum of bag-of-words embeddings
    - added features (user/system, turn no.)
  - weighted match against last user input (dot + softmax)
  - linear transformation to produce next-level input
- last input matched (dot + softmax) against a pool of possible responses

single step of the loop

linear transform

\[ o = R \sum p_i m_i \]

matrix product (a.k.a. attention)

\[ p_i = \text{softmax}(q^T m_i) \]

sum of BoW embeddings

whole dialogue history (except last user input)

last user input

response candidates

loop a few times

multiple steps

(Sukhbaatar et al., 2015) http://arxiv.org/abs/1503.08895
(Bordes et al., 2017) http://arxiv.org/abs/1605.07683
Mem2Seq: Memory nets + pointer-generator = soft DB lookups directly in the model

- “standard” MemNN encoder:
  - special memory:
    - token-level dialogue history (whole history concatenated, no hierarchy)
    - with added turn numbers & user/system flags
    - DB tuples (sums of subject-relation-object)
    - “sentinel” (special token)
  - decoder: MemNN over GRU
    - GRU state is MemNN initial query
    - last level attention is copy pointer
    - if copy pointer points at sentinel, generate from vocabulary
      - copies whenever it can
    - vocabulary distribution comes from 1st level of memory + GRU state
      - linear transform + softmax

(Madotto et al., 2018) [https://www.aclweb.org/anthology/P18-1136]
Attention weights at individual word generation steps

Note: some DB entries were omitted for readability.
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
  • extensions: retrieval-augmented, LLM prompting

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

No labs today
See you next week

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