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

http://ufal.cz/npfl099
Ondřej Dušek, Vojtěch Hudeček & Tomáš Nekvinda
29. 11. 2021
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 doesn’t care about fluency/naturalness
    - either avoid word-level, or mix with 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)
- MLP policy (feed-forward)
- LSTM generator
  - conditioned on policy output, delexicalized
- DB: rule-based, takes most probable belief values
  - creates boolean vector of selected items
  - vector compressed to 6-bin 1-hot (no match, 1 match… >5 matches) on input to policy
  - 1 matching item selected at random & kept for lexicalization after generation

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

• belief tracker trained separately
• rest trained by cross-entropy on generator outputs
• data: CamRest676, collected by crowdsourcing/Wizard-of-Oz
  • workers take turns to be user & system, always just add 1 turn
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: Fully seq2seq-based model

- less hierarchy, simpler architecture
  - no explicit system action – direct to words
  - still explicit dialogue state
  - KB is external (as in most systems)

- seq2seq + 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 pretrained LM
  • system/user embeddings
    • added to Transformer positional embs. & word embs.
  • training to generate as well as classify utterances (good vs. random)
    • all supervised

• no DB & belief tracking (yet, see →)
  • using gold-standard belief & DB, no way of updating belief

pre-LM | seq gen

• Simple adaptation of the GPT pretrained LM
  • system/user embeddings
    • added to Transformer positional embs. & word embs.
  • training to generate as well as classify utterances (good vs. random)
    • all supervised

• no DB & belief tracking (yet, see →)
  • using gold-standard belief & DB, no way of updating belief

decoded part

delexicalized generation

decoded part

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

DB result entry tokens
Real stuff with GPT-2: SOLOIST, SimpleTOD, NeuralPipeline, UBAR

• basically Sequicity over GPT-2: decode belief state, consult DB, decode response
  • history, state, DB results/system action – all recast as sequence
  • finetuning on dialogue datasets

• extensions:
  • specific user/system embeddings (NP)
  • additional training (SOLOIST)
    • not just word-level generation (as GPT-2 default)
    • contrastive objective: detecting fake belief/fake response from real ones
  • explicit system actions (SimpleTOD, UBAR)
    • one more decoding step
    • Context includes dialogue states (UBAR)

(Ham et al., 2020)  https://www.aclweb.org/anthology/2020.acl-main.54
(Yang et al., 2021)  http://arxiv.org/abs/2012.03539
AuGPT: our take on this

- similar to Soloist:
  - “GPT-2 based Sequicity”
  - 1. encode context & user utterance
  - 2. decode belief state
  - 3. query DB
  - 4. encode results
  - 5. decode response
  - consistency auxiliary task

- for robustness & diversity:
  - input data augmentation via backtranslation
  - unlikelihood training (penalize repeated tokens)
  - nucleus sampling (cover ≥ 0.9 probability)

NB: “encode” with GPT-2 means **force-decode**
(ignore the softmax, feed your own tokens)

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

(more auxiliary tasks, not really useful)
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

<table>
<thead>
<tr>
<th>DecL</th>
</tr>
</thead>
<tbody>
<tr>
<td>DecR</td>
</tr>
</tbody>
</table>

DB queried based on updated state

response decoder starting token = # of DB results

---

(Lin et al., 2020)

Few-shot dialogue generation

(Zhao & Eskenazi, 2018) http://aclweb.org/anthology/W18-5001

- **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
Latent Action RL

- 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
  - “fake RL” based 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

(Lubis et al., 2020)
https://aclanthology.org/2020.coling-main.41/
• Hierarchical RL
  • top level: over system actions
  • bottom: over words
• system actions are latent
  • Gaussian distribution
• word-level RL with LM rewards
  • pretrained LSTM LM to provide scores
  • to avoid low fluency due to word-level RL
• REINFORCE with supervised pretraining
  • separate updates on both levels (so you’re not aiming at a moving target)
  • “fake RL” on data (same as previous)
• again, assumes gold state & DB

(Wang et al., 2021)
JOUST: system & simulator joint RL

- System & user simulator with similar architecture
  - both seq2seq-based
  - joint context encoder
  - system: state tracker
    - private context encoder (user shouldn’t see it)
  - user: preset goals & tracking
  - interaction via utterances

- RL over actions
  - REINFORCE, supervised pretraining
  - dialog-level or turn-level rewards
    - turn-level for each reasonable action, e.g. requesting new slot, providing entity etc.

- Domain transfer – new domains / domain combinations from fewer (~100-300) dialogs
Soft DB Lookups

- incorporating NLU/tracker uncertainty into DB results
- making the system fully differentiable
  - but less interpretable
- DB output = distribution over all items
  - plain MLE estimation: 
    \[ p(\text{row } i) = \prod_{\text{slots } j} \begin{cases} \frac{p(v=j)}{\text{# of } v's \text{ in table}} & \text{if } j \text{ specified & in table} \\ 1/\# \text{ rows (uniform)} & \text{otherwise} \end{cases} \]
  - not trained, based directly on tracker
- NLU(trackers – per-slot GRUs + softmaxes
  - input: counts of n-grams
- policy = GRU + softmax
- trained by RL
  - shown to outperform hard DB on a movie domain

(Dinghra et al., 2017)
https://www.aclweb.org/anthology/P17-1045

RNN classif

NPFL099 L9 2021
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 Bag-of-Words embeddings

whole dialogue history (except last user input)

last user input

response candidates

loop a few times

FC + att classif

(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

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

• “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

• encoder (word level)
  • state embedding (1st decoder GRU input)
  • standard MemNN (see previous slide)

• decoder (word level)
  • vocab softmax generated from 1st memory hop
    \[ P_{vocab}(\hat{y}_t) = \text{softmax}(W_1[h_t, o^K]) \]
  • pointer softmax is last memory level attention
    \[ P_{ptr} = p^K_t \]
  • only if \( P_{ptr} \) points at sentinel, \( P_{vocab} \) is used
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 + (sometimes) NLG
  - networks may decompose to components + need dialogue state annotation
  - joint training by backprop (if differentiable)
- Hybrid Code Nets – partially handcrafted, but end-to-end
- Sequicity – seq2seq & 2-step decoding: dialogue state, then response
- GPT-2-based systems – same idea, just with pretrained LMs
- Discrete latent action space – learning w/o action annotation
- Soft DB lookups – making the whole system differentiable
- RL optimization
  - without NLG (over actions) or hierarchical
  - “fake RL” on training data (no simulator needed)
  - JOUST: joint system-simulator training
Thanks

Contact us:
https://ufaldsg.slack.com/
{odusek,hudecek}@ufal.mff.cuni.cz
Skype/Meet/Zoom (by agreement)

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