NPFL099 Statistical Dialogue Systems

9. End-to-end Systems

http://ufal.cz/npfl099

Ondřej Dušek & Vojtěch Hudeček

1. 12. 2020
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

• **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
Supervised with component nets

• 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

<table>
<thead>
<tr>
<th>Encoder</th>
<th>Tracker</th>
<th>Decoder</th>
<th>Match(%)</th>
<th>Success(%)</th>
<th>T5-BLEU</th>
<th>T1-BLEU</th>
</tr>
</thead>
</table>
| Baseline
  | lstm    | lstm    | -        | -          | 0.1650   | 0.1718   |
| Variant
  | lstm    | turn recurrence | lstm      | -        | 0.1813   | 0.1861   |
| lstm    | rnn-cnn, w/o req. | lstm      | 89.70    | 30.60     | 0.1769   | 0.1799   |
| cnn     | rnn-cnn | lstm    | 88.82    | 58.52     | 0.2354   | 0.2429   |

Full model w/ different decoding strategy

<table>
<thead>
<tr>
<th>Encoder</th>
<th>Tracker</th>
<th>Decoder</th>
<th>Match(%)</th>
<th>Success(%)</th>
<th>T5-BLEU</th>
<th>T1-BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>lstm</td>
<td>rnn-cnn</td>
<td>lstm</td>
<td>86.34</td>
<td>75.16</td>
<td>0.2184</td>
<td>0.2313</td>
</tr>
<tr>
<td>lstm</td>
<td>rnn-cnn</td>
<td>+ weighted</td>
<td>86.04</td>
<td>78.40</td>
<td>0.2222</td>
<td>0.2280</td>
</tr>
<tr>
<td>lstm</td>
<td>rnn-cnn</td>
<td>+ att.</td>
<td>90.88</td>
<td>80.02</td>
<td>0.2286</td>
<td>0.2388</td>
</tr>
<tr>
<td>lstm</td>
<td>rnn-cnn</td>
<td>+ att. + weighted</td>
<td>90.88</td>
<td>83.82</td>
<td>0.2304</td>
<td>0.2369</td>
</tr>
</tbody>
</table>

- Average on top 5 candidate outputs
- BLEU for best output
- Match + answered all requested slots
- Returned correct restaurant

-base seq2seq
-HRED (hierarchical seq2seq)

(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
Reinforcement Learning: Recurrent Q-Networks

• NLU + state tracking + DM
  • NLG still kept separate
  • actions are either system DAs or updates to state (DB hypothesis)
  • forced to alternate action types by masking
  • rewards from DB for narrowing down selection

• Models a Q-network as a LSTM
  • or rather LSTM underlying multiple MLPs
    • LSTM maintains internal state representation
  • 1 MLP for system DAs
  • 1 MLP per slot (action=select value X)

(Zhao & Eskenazi, 2016)
http://arxiv.org/abs/1606.02560
Dual RL optimization: agent & user simulator

- end-to-end agent & end-to-end simulator
  - pretrains both with supervised & tunes with RL against each other

**Agent network**

**policy**: 1-layer feed-forward + softmax over actions

**explicit belief state**: 1-layer feed-forward + softmax per slot

**LSTM state** tracker (implicit state)

**User simulator network**

**belief**: tracking currently requested values (using current action)

**goal** predefined: list of slot values to provide & slots to request, constant for dialogue, binary vector

**tracker**, same as agent

**template NLG**, same as agent

**pointer** to k-th KB result (produced as output of tracker, moved when user requests alternatives)

**NLG**: simple templates

KB query is one of the actions, this manages the query results

**Agent network**

**BiLSTM encoding**

**User simulator network**

**BiLSTM encoding**

Dual RL optimization: agent & user simulator

- incremental rewards based on % of completed user goal
  - used by both agent & system
- REINFORCE/Advantage Actor-Critic
- iteratively training agent & user simulator
  - fixing one and training the other for 100 dialogues, then swapping
- joint RL training is better than training just the agent

(Liu & Lane, 2017)
http://arxiv.org/abs/1709.06136
Imitation Learning from Expert Users

- system very similar to previous
  - but only optimizing the system
  - with humans, or simulator
- supervised pretraining
- 2nd step: hybrid SL/RL: **imitation learning** with expert users
  - if the system makes a mistake, user provides correct action & fixed belief
    - needs expert users, laborious – or a good simulator
    - data collected in this way can be used further SL rounds
  - more guidance than RL, but system learns from its own policy
    - no mismatch between training data & policy used by system
- finally: RL with normal user feedback
  - success 0/1 at the end of the dialogue

(Liu et al., 2018) http://arxiv.org/abs/1804.06512
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](https://www.aclweb.org/anthology/P18-1133)
**Sequicity: training + more supervision**

- **training:** supervised – word-level cross-entropy
- **RL fine-tuning with turn-level rewards**
  - prime the system to decode user-requested slot placeholders
- **variant – more supervision**
  - use the same approach to decode explicit NLU output & system action

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

**Sequicity + explicit state**

(Shu et al., 2019) [https://www.aclweb.org/anthology/W19-5922/](https://www.aclweb.org/anthology/W19-5922/)

- the same context encoder as Sequicity
- state decoder:
  - individual slots decoded separately
  - **prevents decoding invalid states**
  - the same decoder run for each slot
- informable:
  - decode values, seq2seq way
- requestable:
  - classify 0/1 if user requested
- response generation:
  - 1st step – classify which slots to include
  - then seq2seq delexicalized generation

![Diagram](image-url)
• 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

• Again, no DB & belief tracking
  • using gold-standard belief & DB, no way of updating belief
Real stuff with GPT-2: SOLOIST, SimpleTOD, NeuralPipeline

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

• small differences/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)
    • one more decoding step

(Peng et al., 2020)
(Hosseini-Asl et al., 2020)
(Ham et al., 2020)

https://www.aclweb.org/anthology/2020.acl-main.54
Structured Fusion Nets: End-to-end on top of individual modules

- 1st step: optimize separate NLU/DM/NLG modules
- 2nd step: optimize end-to-end network over the outputs of modules
Structured Fusion Nets

(Mehri et al., 2019)
https://www.aclweb.org/anthology/W19-5921/

- high-level module on top of NLU/DM/NLG modules works better than just joining, even with joint optimization
- modules can be fine-tuned (end-to-end differentiable)
  - this helps in either case (with modules only or high-level network)
  - multi-task learning doesn’t help more (alternating fine-tuning with module-specific tasks)
- RL: only high-level
  - this way the base generator maintains fluency
  - BLEU OK & success much higher

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>Inform (%)</th>
<th>Success (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised Learning</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Seq2Seq (Budzianowski et al., 2018)</td>
<td>18.80</td>
<td>71.29%</td>
<td>60.29%</td>
</tr>
<tr>
<td>Seq2Seq w/ Attention (Budzianowski et al., 2018)</td>
<td>18.90</td>
<td>71.33%</td>
<td>60.96%</td>
</tr>
<tr>
<td>Seq2Seq (Ours)</td>
<td>20.78</td>
<td>64.40%</td>
<td>54.50%</td>
</tr>
<tr>
<td>Seq2Seq w/ Attention (ours)</td>
<td>20.36</td>
<td>66.50%</td>
<td>59.50%</td>
</tr>
<tr>
<td>Naive Fusion (Zero-Shot)</td>
<td>7.55</td>
<td>70.30%</td>
<td>36.10%</td>
</tr>
<tr>
<td>Naive Fusion (Fine-tuned Modules)</td>
<td>16.39</td>
<td>66.50%</td>
<td>59.50%</td>
</tr>
<tr>
<td>Multitasking</td>
<td>17.51</td>
<td>71.50%</td>
<td>57.30%</td>
</tr>
<tr>
<td>Structured Fusion (Frozen Modules)</td>
<td>17.53</td>
<td>65.80%</td>
<td>51.30%</td>
</tr>
<tr>
<td>Structured Fusion (Fine-tuned Modules)</td>
<td>18.51</td>
<td>77.30%</td>
<td>64.30%</td>
</tr>
<tr>
<td>Structured Fusion (Multitasked Modules)</td>
<td>16.70</td>
<td>80.40%</td>
<td>63.60%</td>
</tr>
<tr>
<td>Reinforcement Learning</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structured Fusion (Frozen Modules) + RL</td>
<td>16.34</td>
<td>82.70%</td>
<td>72.10%</td>
</tr>
</tbody>
</table>

MultiWOZ (multi-domain data)

% dialogues where appropriate entity was provided

% dialogues where system also provided all requested slots
• 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
  • on par with Structured Fusion Nets (slightly higher success, lower BLEU)
• again, ignores DB & belief tracking

(Zhao et al., 2019)
https://www.aclweb.org/anthology/N19-1123
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: \( p(\text{row } i) = \prod_{\text{slots } j} \frac{p(v=j)}{\# \text{ of } v's \text{ in table}} \) if \( j \) specified & in table
  - \( 1/\# \text{ rows (uniform)} \) otherwise
- 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
Key-value Retrieval Nets

- using attention to model DB access
- LSTM encoder, no specific tracker/NLU
- DB in a “key-value” format
  - subject-relation-object (subject-property-value)
  - dinner-time-8pm
  - key = subject + relation
    value = subject_relation
    - i.e. delexicalized values
- generator: seq2seq with 2 attentions
  - over inputs (as usual)
  - over keys in the DB – increases generator output probs. of DB values
    - doesn’t change probs. of regular vocabulary
- supervised training, better than seq2seq/copy

(eric et al., 2017)
https://www.aclweb.org/anthology/W17-5506
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)

\[ o = R \sum p_i m_i \]

(matrix product (a.k.a. attention))

```latex
\bar{v}_{ij} = \text{softmax}(q^T m_i)
```

(last user input)

(add features)

```latex
\text{linear transform}
```

(bag-of-words embeddings)

(last user input against a pool of possible responses)

(response candidates)

(last input matched)

(with attention)
• “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
Mem2Seq visualization

attention weights at individual word generation steps

generated gold: the closest parking garage is civic center garage located 4 miles away at 270 altaire walk
generated gold: the closest parking garage is civic center garage at 270 altaire walk 4 miles away through the directions

subject-relation-object (this gets embedded)

values (these get output)

DB

Memory Context

sentinel "don't copy, generate"
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
Few-shot & Latent Actions

- Latent discrete encoder-decoder
  - discrete VAE for dialogue turns
  - discrete Variational Skip Thought
    - predicting next turn
  - trained jointly

- Full model:
  - LAED to predict next action
  - DI-VAE for user input representation
  - HRED with ELMo
  - KVRET-like DB representation
    - DB is treated as part of context
  - decoder: same as previous
    - copy with sentinel
  - uses NER/entity linking instead of handcrafted annotations

(Zhao et al., 2018) [http://aclweb.org/anthology/P18-1101](http://aclweb.org/anthology/P18-1101)

Summary

• End-to-end = single network for NLU/tracker + DM + (sometimes) NLG
  • networks often decompose to components + need dialogue state annotation
  • joint training by backprop (if differentiable)
  • RL – interleaved with supervised, without NLG (over actions)

• Hybrid Code Nets: partially handcrafted, but end-to-end
• Sequicity: seq2seq-based & decoding dialogue state
• GPT-2-based: same idea, just with pretrained LMs
• Soft DB lookups – making the whole system differentiable
  • “transparent” (directly based on tracker)
  • attention/memory nets (multi-hop attention)

• Few-shot: lot of autoencoding
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