Statistical Dialogue Systems

NPFL099 Statistické Dialogové systémy

8. NLG(2) & End-to-End Dialog Systems

Ondřej Dušek & Vojtěch Hudeček

http://ufal.cz/npfl099

5. 12. 2019

NLG-NLU Combo: Self-training

(Kedzie & McKeown, 2019)
https://arxiv.org/abs/1911.03373

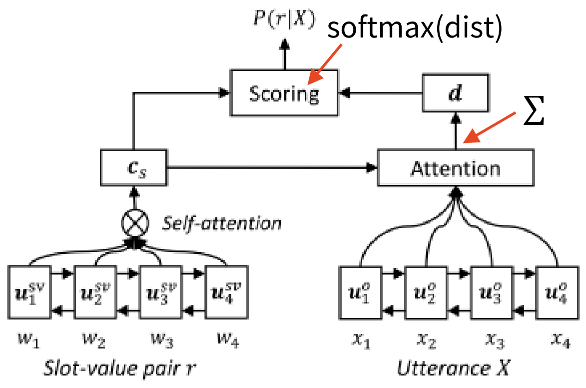
- Create your own additional training data
  - to make the generator more robust & accurate
- needs an NLU trained on original data
- Approach:
  - Train base generator (42k instances)
  - Sample more data from it
    - sample many DAs at random (25k for each # of slots)
    - **noise injection sampling** greedy decoding with Gaussian noise in hidden states
      - use noise injection sampling to get many texts for each DA (200 texts per DA)
    - classify each sampled instance with an NLU
      - discard any texts which don't correspond to the DA

    ensure clean generated data
  - Train generator on original & sampled data (can loop more)
- Near perfect accuracy with basic seq2seq+attention as generator
  - with rule-based or CNN-based NLU, on restaurants data

(Nie et al., 2019)
https://www.aclweb.org/anthology/P19-1256

NLG-NLU Combo: NLU data cleaning

- NLU used to clean training data (see fact grounding)
  - NLU model – BiLSTM + attention & vector distance
- Training NLU iteratively:
  - train initial NLU on all data
  - parse DAs for all data
  - select only data where NLU gives high confidence
  - use high-confidence data to tune the NLU
- NLG (seq2seq+copy) trained on NLU-reparsed data
  - increases semantic accuracy greatly



original data

plain supervised NLU

iterative NLU training

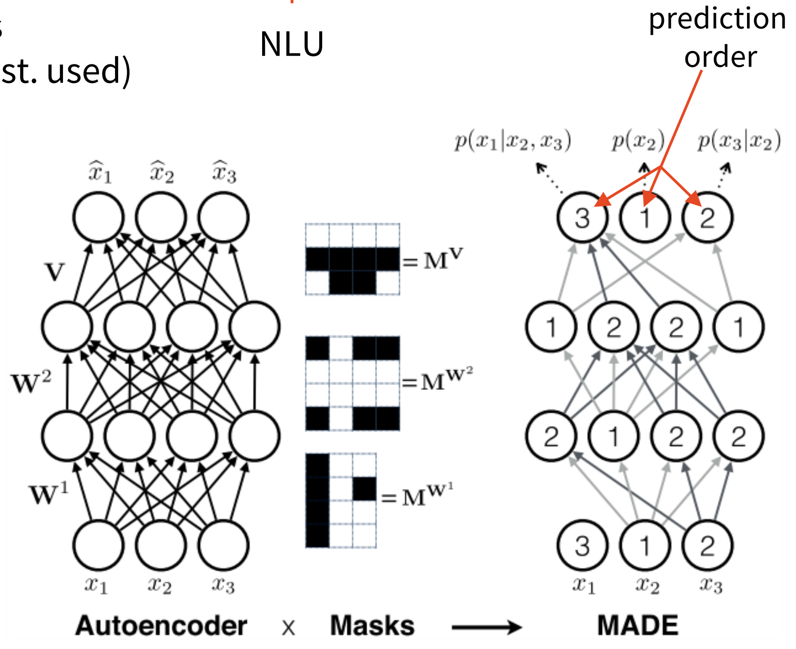
	BLEU(%)	Err(%)
TGen	65.90	18.09 (114/630)
Slug2Slug	66.19	6.51 (41/630)
Seq2Seq	66.15	69.37 (374/630)
Seq2Seq+aug	**66.49**	28.89 (182/630)
Seq2Seq+aug+iter	65.63	2.07 (13/630)
Seq2Seq+aligner	63.81	**1.75** (11/630)

handcrafted NLU

NLG-NLU Combo: Dual training

(Su et al., 2019)
http://arxiv.org/abs/1905.06196

- multi-objective optimization
  - basically normal training with regularization for duality:
    $$P(x,y) = P(x)P\big(y\big|x,\theta_{x\to y}\big) = P(y)P(x|y,\theta_{y\to \mathrm{x}})$$
    - attempting to model the whole distribution $P(x,y)$, should work both ways (via both NLU and NLG)
    - regularization term: $\Big(\underbrace{\log P(x)}_{\text{DAs (empirical dist. used)}} + \underbrace{\log P\big(y\big|x,\theta_{x\to y}\big)}_{\text{NLG}} - \underbrace{\log P(y)}_{\text{texts (empirical dist. used)}} - \underbrace{\log P\big(x\big|y,\theta_{y\to \mathrm{x}}\big)}_{\text{NLU}}\Big)^2$
      - if duality holds, this is 0
    - added to both NLG and NLU training, with given weight
  - NLG & NLU = seq2seq (GRU)
  - $P(y)$ = RNN language model
  - $P(x)$ = masked autoencoder
    - create dependencies among slots
    - join multiple possible dependency orders

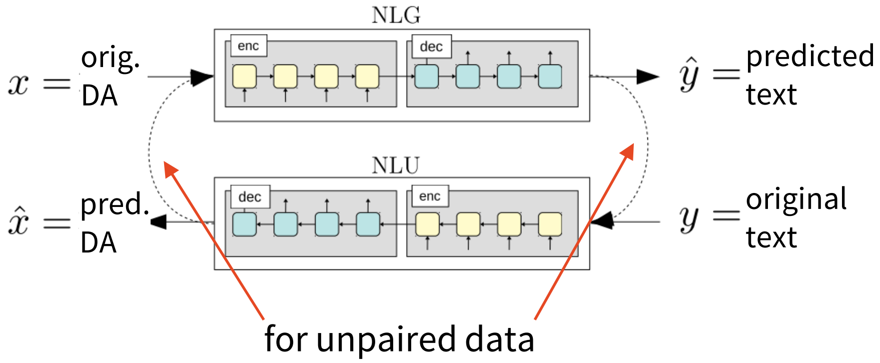


(Germain et al., 2015) http://proceedings.mlr.press/v37/germain15.pdf

NLG-NLU Combo: Semi-supervised

(Qader et al., 2019)
https://arxiv.org/abs/1910.03484

- learn from partially unpaired data
  - some DA-text pairs, some loose DAs, some loose texts
- similar to previous: symmetric models, joint optimization
  - $\text{loss} = \alpha \cdot \text{loss}_{\text{NLG}}^{\text{paired}} + \beta \cdot \text{loss}_{\text{NLG}}^{\text{unpaired}} + \gamma \cdot \text{loss}_{\text{NLU}}^{\text{paired}} + \delta \cdot \text{loss}_{\text{NLU}}^{\text{unpaired}}$
  - losses for paired data are as usual (MLE, seq2seq models)
  - unpaired case: models are connected, reconstruction loss
    - loss is difference from original text/DA when passing through the whole loop
    - greedy decoding
    - making it fully differentiable: Straight-Through Gumbel-Softmax
      - Gumbel-Softmax: approximate sampling from categorial token distributions
      - straight-through = real (hard) sampling for forward pass, smooth approximation for backward pass



Gumbel-Softmax

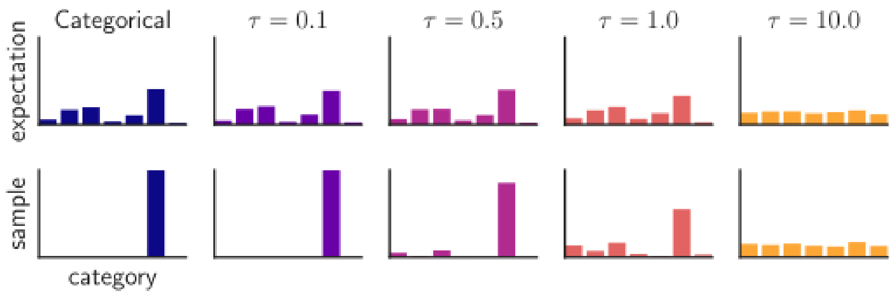
(Jang et al., 2017)
https://arxiv.org/abs/1611.01144

- "reparameterization trick for discrete distributions"
  - reparameterization: $z \sim \mathcal{N}(\mu, \sigma) \rightarrow z \sim \mu + \sigma \cdot \mathcal{N}(0,1)$ ← Normal noise
    - differentiating w. r. t. μ & σ still works, no hard sampling on that path
- Gumbel-max:
  - categorical distribution π with probabilities π_i
  - sampling from π: $z = \text{onehot}(\arg\max_i(\log \pi_i + g_i))$ ← Gumbel noise: $g_i = -\log(-\log(\text{Uniform}(0,1)))$
- Swap argmax for softmax with temperature τ:

$$y_i = \frac{\exp\left(\frac{\log(\pi_i) + g_i}{\tau}\right)}{\sum_{j=1}^{N} \exp\left(\frac{\log(\pi_j) + g_j}{\tau}\right)}$$

  - can differentiate w. r. t. π if $\tau > 0$

$\tau \rightarrow 0$: more like one-hot ← → $\tau \rightarrow \infty$: more like uniform



“Unsupervised” NLG

(Freitag & Roy, 2018)
http://aclweb.org/anthology/D18-1426

- treat an NLG system as a denoising autoencoder
  - “fill in missing/corrupted sentences”
  - DA is a “corrupted sentence” with just the values to generate
- preparing unlabeled data:
  - removing only frequent words (~assuming these are not slot values)
  - shuffling, but keeping original bigrams
  - adding more out-of-domain data (news)

name	type	food	family friendly
Loch Fyne	restaurant	Indian	yes

original	Loch Fyne is a family friendly restaurant providing Indian food .
remove random 60%	Fyne is restaurant food .
remove only words w_i with $N(w_i) > 100$	Loch Fyne family friendly Indian
+shuffle words	family friendly Indian Loch Fyne

this one is used

- model: standard seq2seq
- works better than supervised (lower BLEU, but better accuracy)
- only works for simple DAs
  - E2E restaurants: not even a real DA, just slots & values, overlap with text

NLG with Pretrained LMs

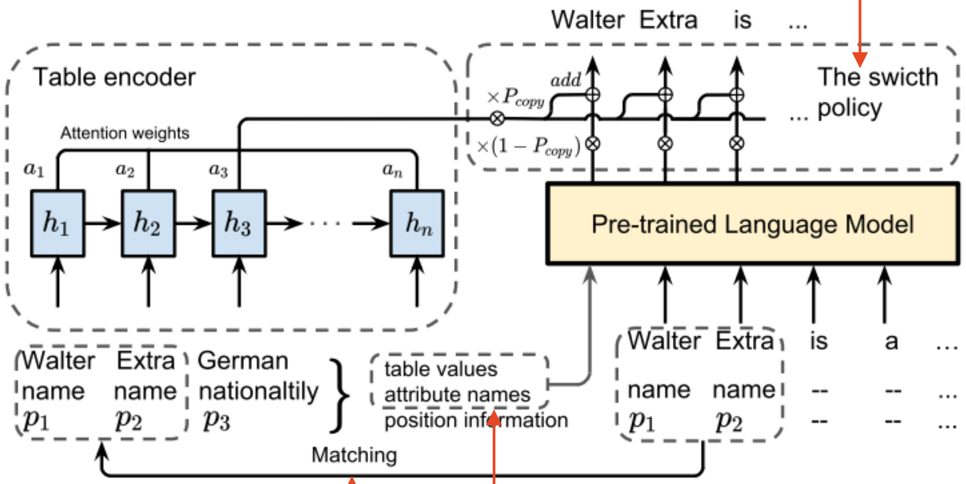
(Chen et al., 2019)
http://arxiv.org/abs/1904.09521

- GPT-2 (pretrained Transformer LM)
  - Transformer trained for next-word prediction
  - initialized by preceding context by default
    → tuned to use input data
  - word embeddings fixed
- using copy (pointer-generation) on top
  - LM fine-tuned, forced to copy inputs
  - additional loss term for copying
- encoder: field-gating LSTM
  - 2-layers: bottom (table field info) added to cell state of top (values)
- learns from very few training examples
  - reasonable outputs with 200 training instances

Attribute (R)	name	nationality	occupation	...
Value (V)	Walter Extra	German	aircraft designer and manufacturer	...

input: WikiBio – tables

generate from LM
or copy from input?



newly trained LM context

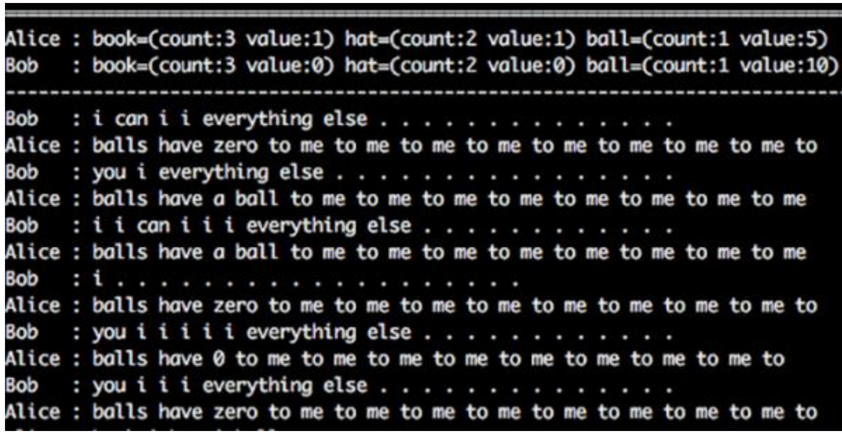
during training:
to find out where to copy inputs

End-to-end dialogue systems

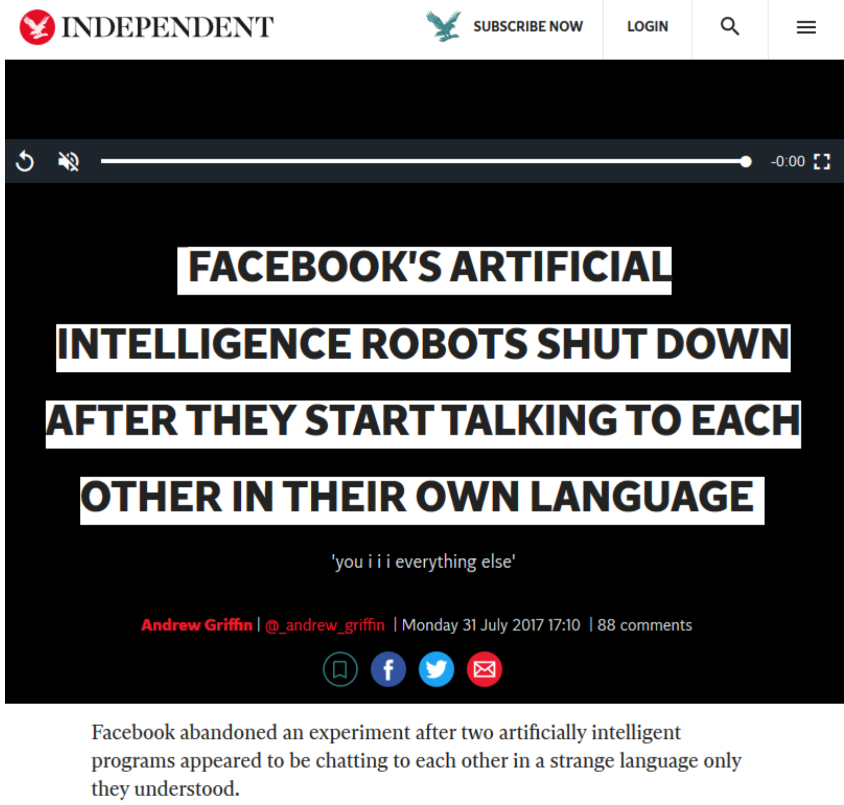
- Separate components:
  - more flexible
  - error accumulation
  - improved components don’t mean improved system
  - possibly joint optimization by RL
- End-to-end:
  - joint optimization by backprop
    - if fully differentiable
  - still can work via RL (with supervised initialization)
  - architectures typically still decompose into original DS components
    - and often still need DA-level annotation
- Not all systems join all components
  - e.g. just NLU + tracker + policy, NLG excluded

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



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

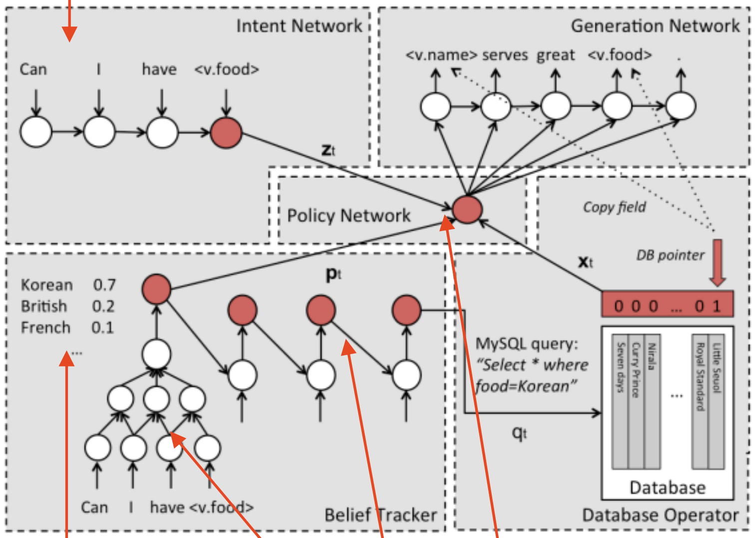


https://www.independent.co.uk/life-style/gadgets-and-tech/news/facebook-artificial-intelligence-ai-chatbot-new-language-research-openai-google-a7869706.html

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

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

- 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

average on top 5 candidate outputs

BLEU for best output

base seq2seq

HRED (hierarchical seq2seq)

Encoder	Tracker	Decoder	Match(%)	Success(%)	T5-BLEU	T1-BLEU
Baseline						
lstm	-	lstm	-	-	0.1650	0.1718
lstm	turn recurrence	lstm	-	-	0.1813	0.1861
Variant						
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						
lstm	rnn-cnn	lstm	86.34	75.16	0.2184	0.2313
lstm	rnn-cnn	+ weighted	86.04	78.40	0.2222	0.2280
lstm	rnn-cnn	+ att.	90.88	80.02	0.2286	0.2388
lstm	rnn-cnn	+ att. + weighted	90.88	83.82	0.2304	0.2369

added attention

length-weighted decoding

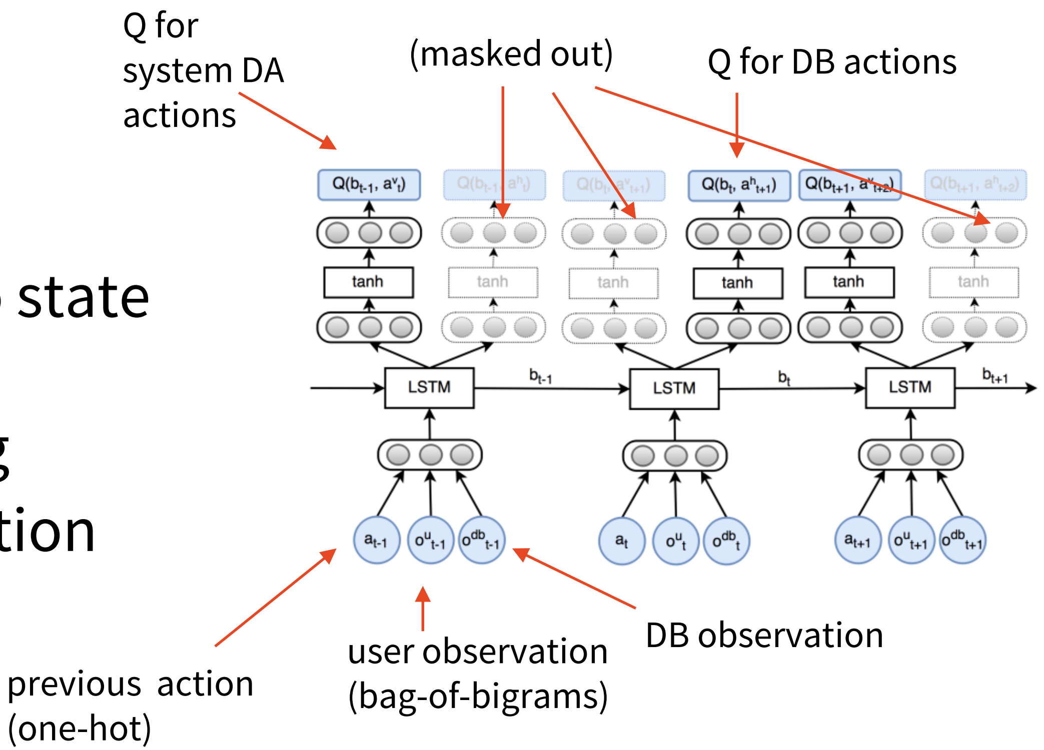
returned correct restaurant

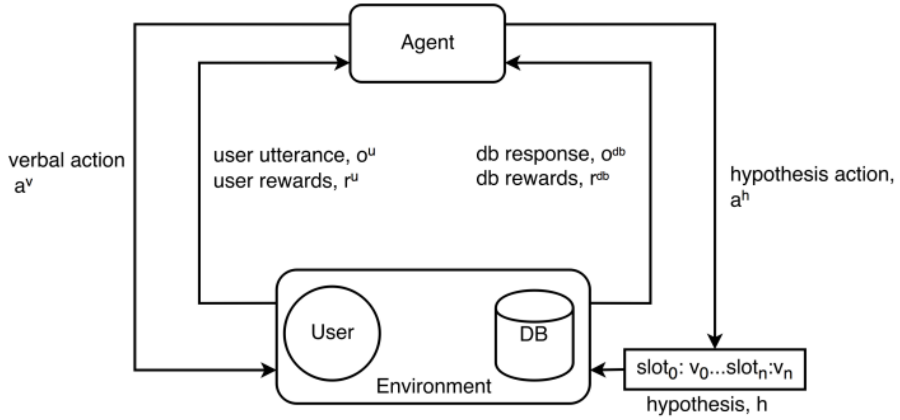
match + answered all requested slots

Reinforcement Learning: Recurrent Q-Networks

(Zhao & Eskenazi, 2016)
http://arxiv.org/abs/1606.02560

- 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)

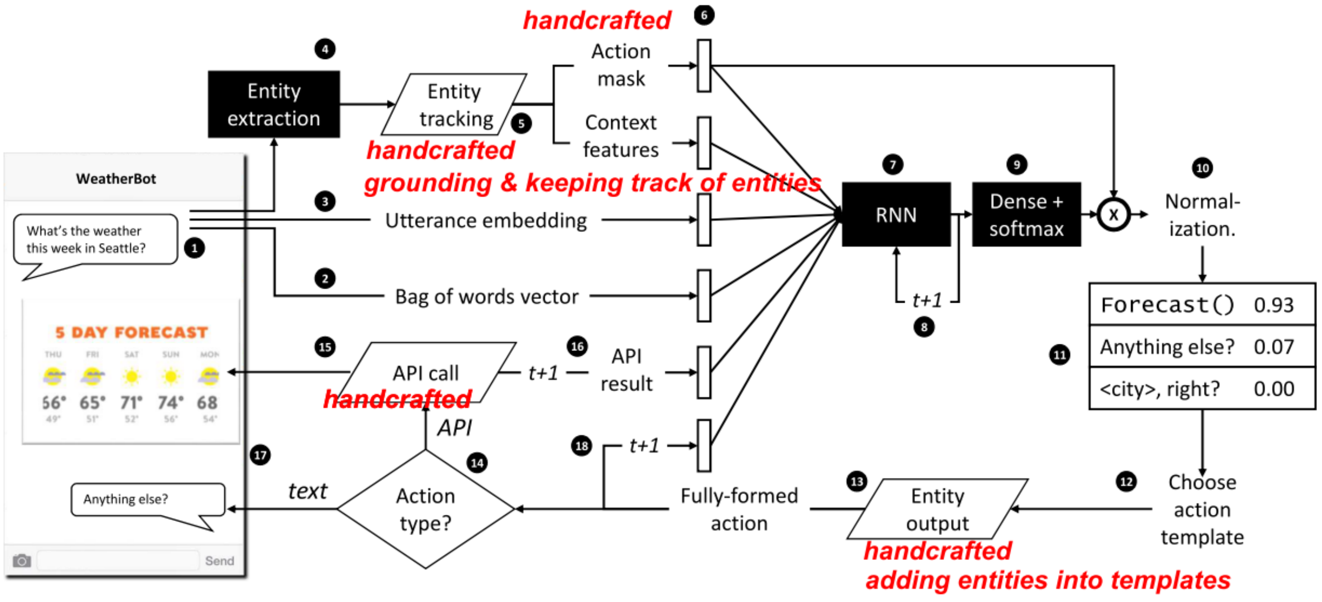




Hybrid Code Networks

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

- 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 tracking annotation
- feed-forward **policy** – produces probability distribution over actions
  - mask applied to outputs & renormalized → choosing action = output template

Hybrid Code Networks

- handcrafted fill-in for entities
  - this way, the learned code can be fully delexicalized (depends on context features from entity extraction step)
- actions can trigger API calls
  - APIs can return features for next timesteps
- can be trained using supervised learning
  - beats a fully rule-based system with only 30 training dialogues
- can be further fine-tuned using reinforcement learning
  - REINFORCE with baseline
  - also, RL & SL can be interleaved
- various extensions to the model have been tried
  - especially better input than binary & averaged embeddings

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

Dual RL optimization: agent & user sim.

(Liu & Lane, 2017) http://arxiv.org/abs/1709.06136

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

Agent network

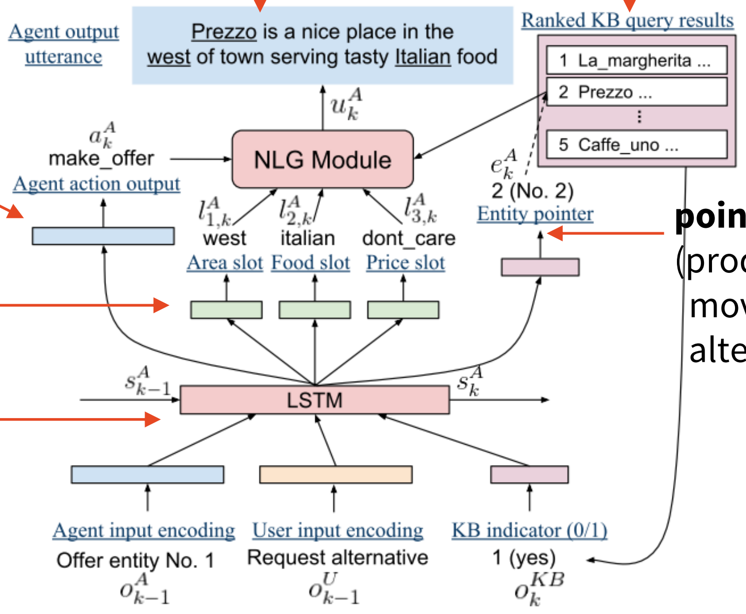
NLG: simple templates

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

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

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

LSTM state tracker (implicit state)



BiLSTM encoding

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

User simulator network

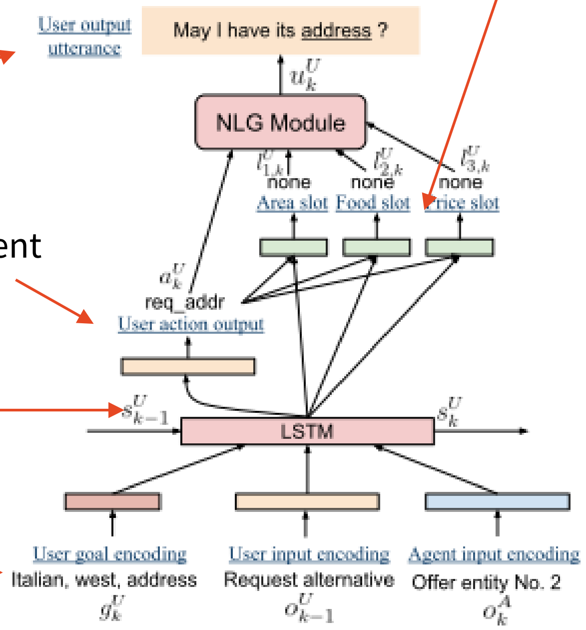
belief: tracking currently requested values (using current action)

template NLG, same as agent

policy, same as agent

tracker, same as agent

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

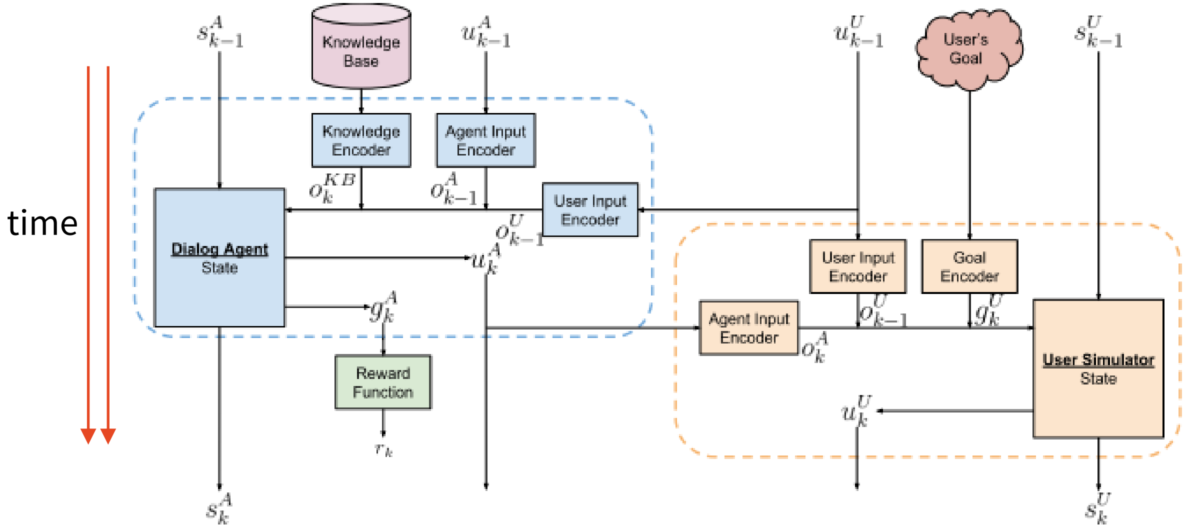


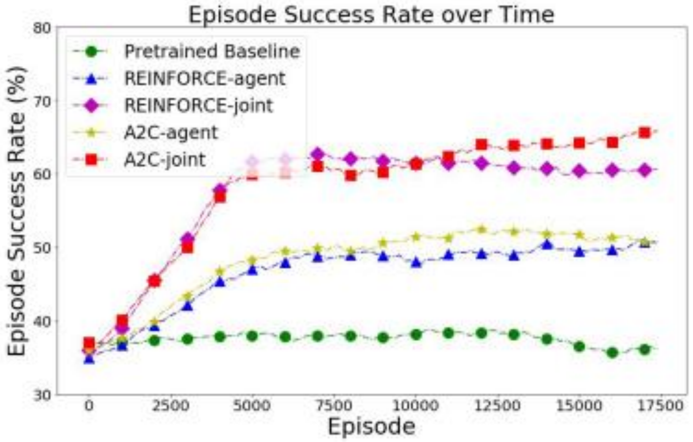
BiLSTM encoding

Dual RL optimization: agent & user sim.

(Liu & Lane, 2017) http://arxiv.org/abs/1709.06136

- 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

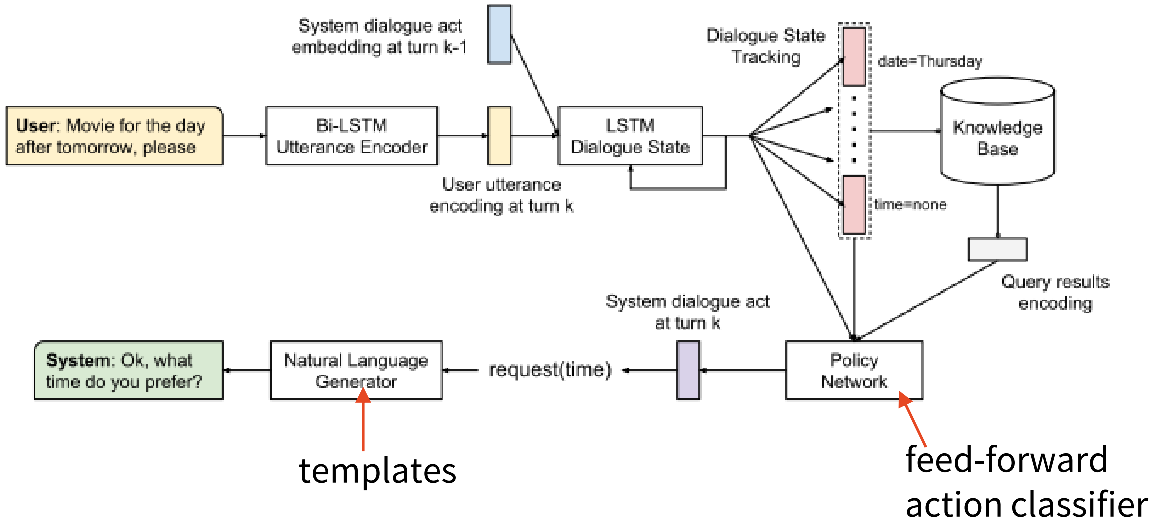


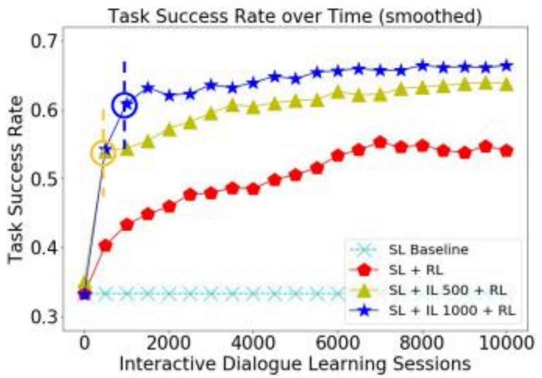


(Liu et al., 2018) http://arxiv.org/abs/1804.06512

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

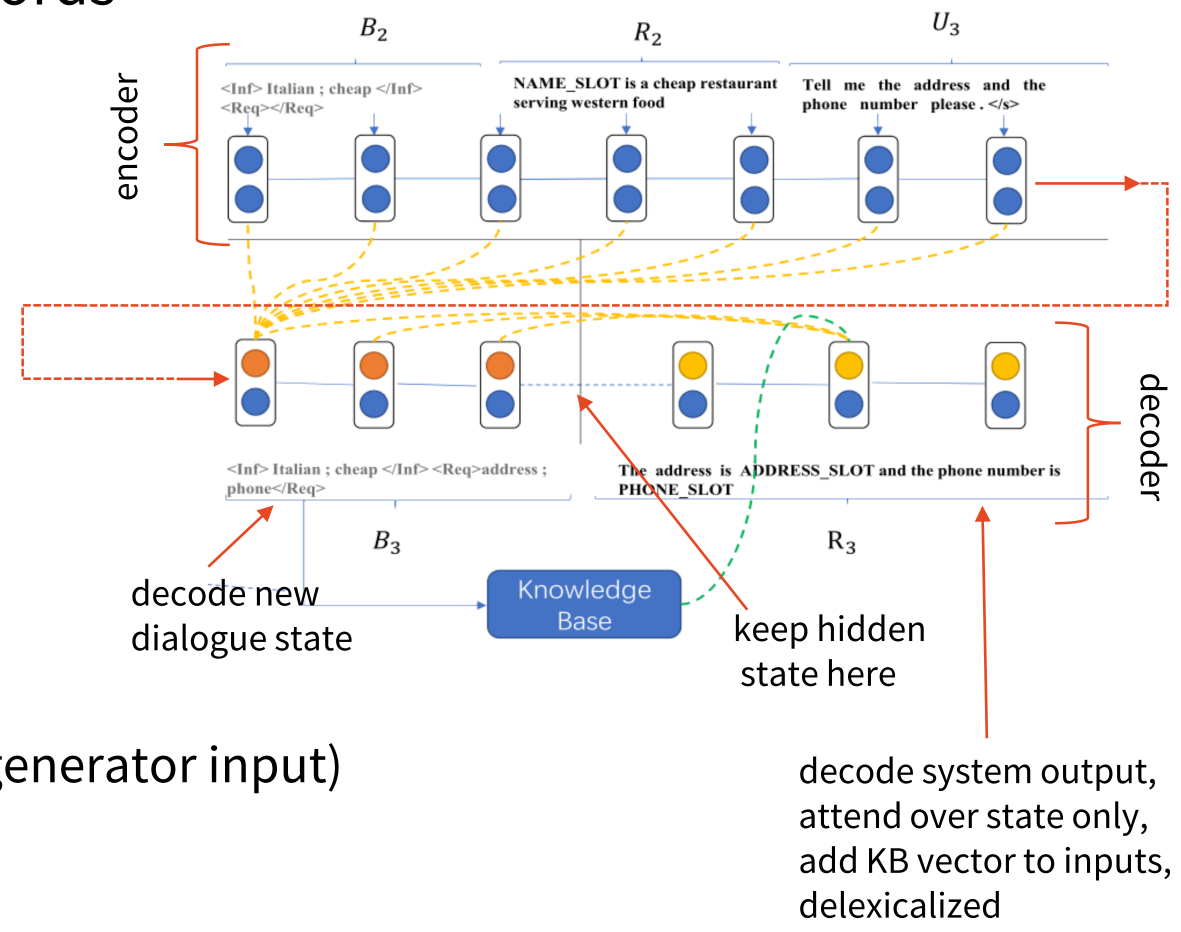




(Lei et al., 2018) https://www.aclweb.org/anthology/P18-1133

Sequicity: Fully seq2seq-based model

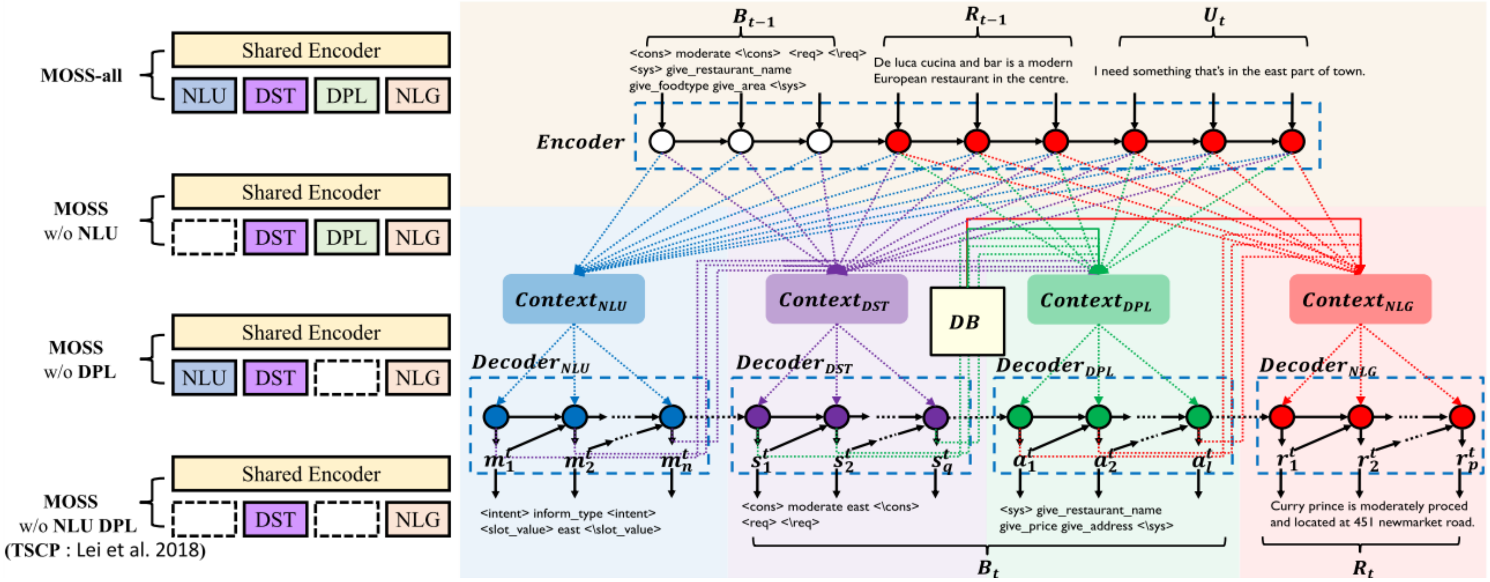
- less hierarchy, simpler architecture
  - no explicit system action – direct to words
  - still explicit dialogue state
- seq2seq-style:
  - encode: previous dialogue state
    + prev. system response
    + current user input
  - decode new state first
    - attend over whole encoder
  - decode system 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
- using copy net (pointer-generator)



Sequicity

(Lei et al., 2018) https://www.aclweb.org/anthology/P18-1133
(Liang et al., 2019) http://arxiv.org/abs/1909.05528

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



Summary

- NLG:
  - NLG + NLU combination:
    - cleaning data, sampling more training data, dual training, semi-supervised
  - NLG as denoising autoencoder
  - fine-tuning for pretrained GPT-2 language model
- End-to-end models:
  - typically NLU/tracker + DM + (sometimes) NLG
    - networks decompose to components, often need dialogue state annotation
  - **joint training by backprop** (if differentiable)
  - **RL** (interleaved with supervised / without NLG)
    - dual optimization: system + simulator
    - imitation learning – step-wise learning from users
  - Hybrid Code Nets: partially handcrafted, but end-to-end
  - Sequicity: seq2seq-based, decoding dialogue state

Thanks

Contact us:

odusek@ufal.mff.cuni.cz
hudecek@ufal.mff.cuni.cz
(or on Slack)

No labs today

Get these slides here:

http://ufal.cz/npfl099

References/Inspiration/Further:

- Gatt & Krahmer (2017): Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation http://arxiv.org/abs/1703.09902
- My PhD thesis (2017), especially Chapter 2: http://ufal.mff.cuni.cz/~odusek/2017/docs/thesis.print.pdf
- Gao et al. (2019): Neural Approaches to Conversational AI: https://arxiv.org/abs/1809.08267