NPFL099 Statistical Dialogue Systems

7. Dialogue Policy (2) + Language Generation

http://ufal.cz/npfl099

Ondřej Dušek & Vojtěch Hudeček

10. 11. 2020

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

Recap from last time: Reinforcement Learning

- RL = find a **policy** that maximizes long-term reward
  - MDP representation: agent in an environment
  - taking **actions**, moving across **states**, getting **rewards**
- optimization approaches:
  - Monte Carlo – sample (a dialogue), then update
  - Temporal Difference – look ahead, refine estimates as you go
  - **actor** (optimize policy directly) vs. **critic** (indirectly via state/action values)
- Q-networks – optimizing indirectly (critic) via $\boldsymbol{Q}$ **= action-value function**
  - Q = expected return of taking action a in state s under policy π
  - greedy policy under Q: "choose what's best for next step according to Q"
  - if Q is optimal, its greedy policy is also optimal
- Deep Q Networks = just represent Q with a neural net
  - + a few tricks (experience replay, target freezing)

Policy Gradients

- Instead of value functions, train a **network to represent the policy**
  - allows better action sampling – according to actual stochastic policy
    - no need for ϵ-greedy (which is partially random, suboptimal)
- To optimize, we need a **performance metric**: $J(\theta) = V^{\pi_\theta}(s_0)$
  - expected return in starting state when following π_θ
  - we want to directly optimize this using gradient ascent
- **Policy Gradient Theorem**:
  - expresses $\nabla J(\theta)$ in terms of $\nabla \pi(a|s,\theta)$

$$\nabla J(\theta) \propto \underbrace{\sum_s \mu(s)}_{\mu(s)\text{ is state probability under }\pi\text{ – this is the same as expected value }E_\pi} \sum_a Q^\pi(s,a) \nabla \pi(a|s,\theta) = E_\pi\left[\sum_a Q^\pi(s,a) \nabla \pi(a|s,\theta)\right]$$

(Sutton & Barto, 2018; p. 324ff)

REINFORCE: Monte Carlo Policy Gradients

- direct search for policy parameters by stochastic gradient ascent
  - looking to maximize performance $J(\boldsymbol{\theta}) = V^{\pi_\theta}(s_0)$
- choose learning rate α, initialize $\boldsymbol{\theta}$ arbitrarily
- loop forever:
  - generate an episode $s_0, a_0, r_1, \dots, s_{T-1}, a_{T-1}, r_T$, following $\pi(\cdot \mid \cdot, \boldsymbol{\theta})$
  - for each $t = 0,1 \dots T$: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \underbrace{\alpha \gamma^t R_t \nabla \ln \pi(a_t | s_t, \boldsymbol{\theta})}$

this will guarantee the right state distribution/frequency $\mu(s)$

returns $R_t = \sum_{i=t}^{T-1} \gamma^{i-t} r_{i+1}$

this is stochastic $\nabla J(\boldsymbol{\theta})$:

- from policy gradient theorem
- using single action sample a_t
- expressing Q^π as R_t (under E_π)
- using $\nabla \ln x = \frac{\nabla x}{x}$

variant – **advantage** instead of returns: discounting a **baseline** $b(s)$ (predicted by any model) $A_t = R_t - b(s_t)$ instead of R_t gives better performance

$V(s)$ is actually a good $b(s)$

(Sutton & Barto, 2018; p. 327f)

Policy Gradients (Advantage) Actor-Critic

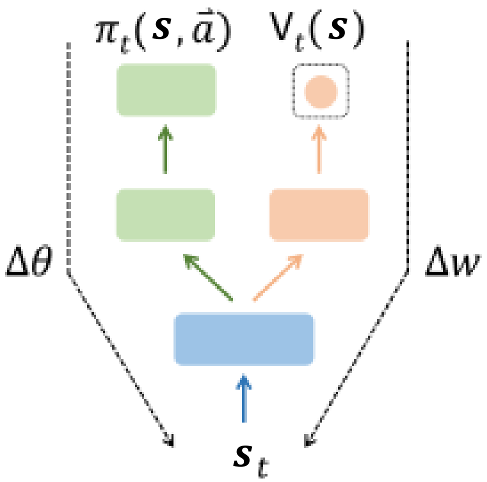
- REINFORCE + V approximation + TD estimates – better convergence
  - differentiable policy $\pi(a|s, \boldsymbol{\theta})$
  - differentiable state-value function parameterization $\hat{V}(s, \boldsymbol{w})$
  - two learning rates $\alpha^{\boldsymbol{\theta}}, \alpha^{\boldsymbol{w}}$
- loop forever:
  - set initial state s for the episode
  - for each step t of the episode:
    - sample action a from $\pi(\cdot\,|s, \boldsymbol{\theta})$, take a and observe reward r and new state s'
    - compute **advantage** $A \leftarrow r + \gamma\hat{V}(s', \boldsymbol{w}) - \hat{V}(s, \boldsymbol{w})$
    - update $\underbrace{\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\gamma^t A\nabla \ln \pi(a|s, \boldsymbol{\theta})}$, $\underbrace{\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha^{\boldsymbol{w}} \cdot A\nabla\hat{V}(s, \boldsymbol{w})}$
    - $s \leftarrow s'$

actor (policy update)

critic (value function update)

TD: update after each step

same as REINFORCE, except:
- we use $\hat{V}(s, \boldsymbol{w})$ as baseline
- r is used instead of R_t (TD instead of MC)



(Su et al., 2017)
http://arxiv.org/abs/1707.00130

ACER: Actor-Critic with Experience Replay

- off-policy actor-critic – using **experience replay** buffer
  - same approach as Q learning
  - since ER buffer has past experience with out-of-date policies (using "old" $\tilde{\theta}$), it's considered off-policy (behaviour policy $\pi_{\tilde{\theta}} \neq$ target policy π_θ)
    - sampling behaviour from $\pi_{\tilde{\theta}}$ is biased w. r. t. π_θ
    - correcting the bias – **importance sampling**: multiply by importance weight $\rho_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\tilde{\theta}}(a_t|s_t)}$
  - all updates are summed over batches & importance-sampled
    - new objective/performance metric: $\hat{E}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\tilde{\theta}}(a_t|s_t)}\hat{A}_t]$

using advantage instead of returns

batch average
over timesteps t

importance sampled

(Wang et al., 2017) http://arxiv.org/abs/1611.01224
(Su et al., 2017) http://arxiv.org/abs/1707.00130
(Weisz et al., 2018) http://arxiv.org/abs/1802.03753

TRACER: Trust-Region ACER

(Wang et al., 2017) http://arxiv.org/abs/1611.01224
(Su et al., 2017) http://arxiv.org/abs/1707.00130
(Weisz et al., 2018) http://arxiv.org/abs/1802.03753

- ACER may be unstable/slow to learn
  - prone to excessively large updates – need to set learning rates low
    - high learning rate = unstable, high variance
    - low learning rate = too slow
- → regularize: **limit KL-divergence change** between updated policy θ & average policy $\overline{\theta}$
  - $\overline{\theta}$ is a moving average of past policies: $\overline{\theta} \leftarrow \alpha\overline{\theta} + (1-\alpha)\theta$
  - modified policy gradient g is defined as:
    $\min_g \frac{1}{2} ||\nabla\theta - g||_2^2$ so that $\nabla KL[\pi_{\overline{\theta}}(s_t)||\pi_\theta(s_t)]^T g \leq \xi$
    (approx. increase in KL)
    - minimizing sum of squared differences (L2)
    - i.e. the closest you can get to the gradient, but don't increase KL between the average and new policy too much
    - quadratic programming, has closed-form solution

standard update (excessive)

trust region

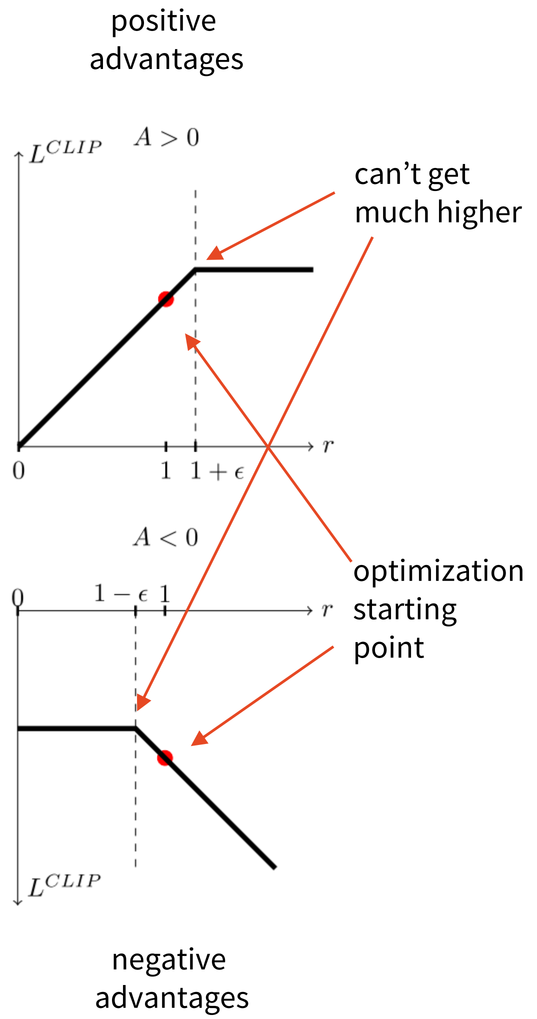
https://medium.com/@jonathan_hui/rl-trust-region-policy-optimization-trpo-explained-a6ee04eeeee9

Proximal Policy Optimization

(Schulman et al., 2017) http://arxiv.org/abs/1707.06347

- Changing the objective to be more like trust-region
  - without the need to adjust gradients & do the optimization
- Basically clipping the ACER objective
  - define $r_t(\theta) = \dfrac{\pi_\theta(a_t|s_t)}{\pi_{\tilde{\theta}}(a_t|s_t)}$ – ratio to old params
  - starting from $\hat{E}_t\left[\dfrac{\pi_\theta(a_t|s_t)}{\pi_{\tilde{\theta}}(a_t|s_t)}\hat{A}_t\right] = \hat{E}_t\left[r_t(\theta)\hat{A}_t\right]$ (see ACER)
  - using $\hat{E}_t\left[\min\left(\underbrace{r_t(\theta)\hat{A}_t}_{\text{original}}, \underbrace{\text{clip}[r_t(\theta)]_{1-\epsilon}^{1+\epsilon}\hat{A}_t}_{\text{clipped to stay close to 1}}\right)\right]$

minimum – lower bound on the unclipped objective

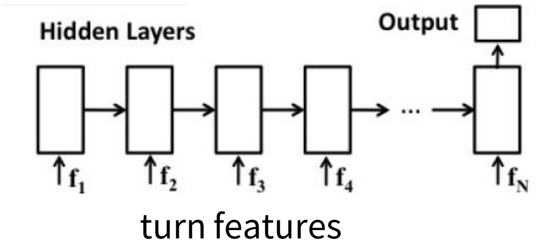


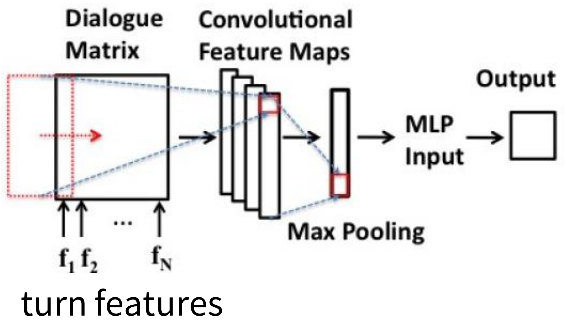
Rewards in RL

- Reward function is critical for successful learning
- Handcrafting is not ideal
  - domain knowledge typically needed to detect dialogue success
  - need simulated or paid users,
    can’t learn from users without knowing their task
  - paid users often fail to follow pre-set goals
- Having users provide feedback is costly & inconsistent
  - real users don’t have much incentive to be cooperative
- Learning/optimizing the rewards is desirable

Supervised dialogue quality estimation

- turn features → RNN/CNN → success/fail or return (multi-class/regression)
  - user & system DA (one-hot)
  - belief state (per-slot prob. distributions)
  - turn number
- trained from data collected by training a DM with a user simulator
  - using handcrafted rewards
  - success/failure & return known
  - acc. >93% on 18k dialogues, ~85-90% on 1k dialogues
    - binary RNN best (not too huge differences)
- used as reward estimator ≥ handcrafted
  - similar performance & doesn't need known goals
  - can learn from real users
  - still ultimately based on handcrafted rewards





(Su et al., 2015)
http://arxiv.org/abs/1508.03386

Turn-level Quality Estimation

(Schmitt & Ultes, 2015; Ultes et al., 2017; Ultes, 2019)
https://doi.org/10.1016/j.specom.2015.06.003
https://doi.org/10.21437/Interspeech.2017-1032
https://aclweb.org/anthology/W19-5902/

Interaction Quality

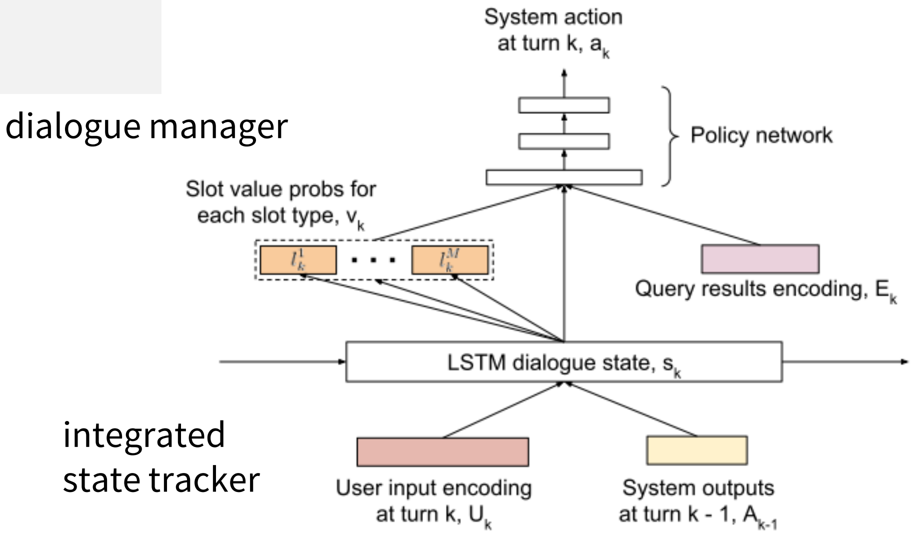
- turns annotated by experts (Likert 1-5)
- trained model (SVM/RNN)
  - very low-level features
  - mostly ASR-related
  - multi-class classification
- result is domain-independent
  - trained on a very small corpus (~200 dialogues)
  - same model applicable to different datasets
- can be used in a RL reward signal
  - works better than task success

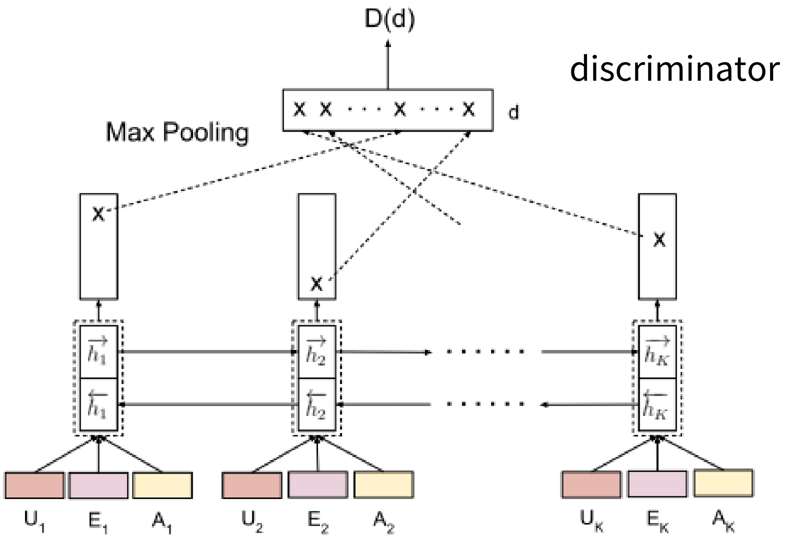
		Parameter	Description
current turn	*Exchange level*	ASRRecognitionStatus	ASR status: *success*, *no match*, *no input*
		ASRConfidence	confidence of top ASR results
		RePrompt?	is the system question the same as in the previous turn?
		ActivityType	general type of system action: *statement*, *question*
		Confirmation?	is system action confirm?
whole dialogue	*Dialogue level*	MeanASRConfidence	mean ASR confidence if ASR is success
		#Exchanges	number of exchanges (turns)
		#ASRSuccess	count of ASR status is success
		%ASRSuccess	rate of ASR status is success
		#ASRRejections	count of ASR status is reject
		%ASRRejections	rate of ASR status is reject
last 3 turns	*Window level*	{Mean}ASRConfidence	mean ASR confidence if ASR is success
		{#}ASRSuccess	count of ASR is success
		{#}ASRRejections	count of ASR status is reject
		{#}RePrompts	count of times RePromt? is true
		{#}SystemQuestions	count of ActivityType is question

“reject” = ASR output doesn’t match in-domain LM

Reward as discriminator

- no predefined rewards, learn from data
  - known success, but learned reward for it
  - success = match user slot values
    & provide all requested information
- discriminator: LSTM + max-pooling
  - classify 1/0 successful (from dataset) vs. simulated over whole dialogue
- dialogue manager
  - LSTM tracker & feed-forward policy in a single model
- supervised pretraining + GAN-style training
  - supervised reward learning = “inverse RL”
  - DM: REINFORCE with rewards from discriminator
  - discriminator: sample with current DM
    & train to classify successful vs. simulated





(Liu & Lane, 2018) http://arxiv.org/abs/1805.11762

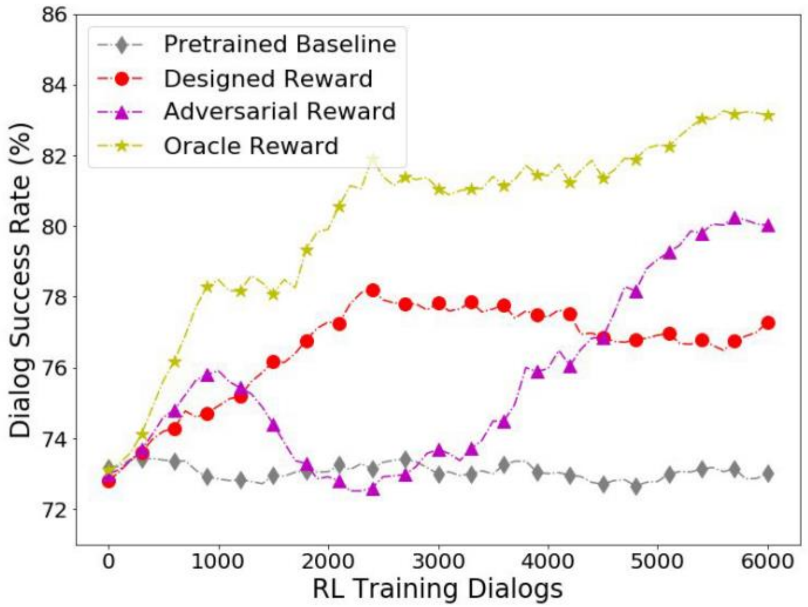
Reward as discriminator

- comparing rewards

does not copy the actual dialogue success

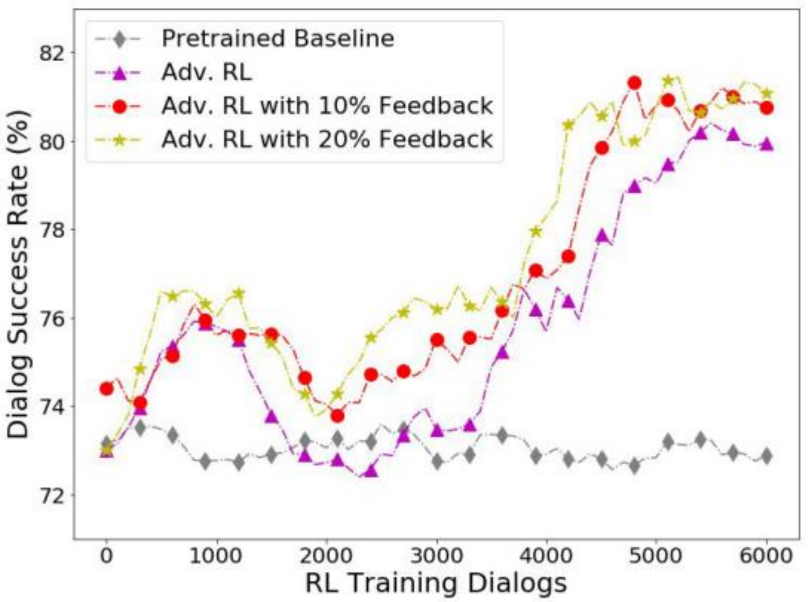
known goal only
  - **oracle** = 1/0 successful/failed
  - **designed** = +1 for each correct slot, +1 for each informed request (with correct slots)

also unknown
  - **pretrained** = without the GAN training
  - **adversarial** = full setup with GAN training
  - adversarial better than handcrafted



(results on DSTC2 data)

- can also learn from partial user feedback
  - counters disadvantage for dialogues different from previous policy
  - use discriminator if feedback is not available
  - further slight improvement



(Liu & Lane, 2018) http://arxiv.org/abs/1805.11762

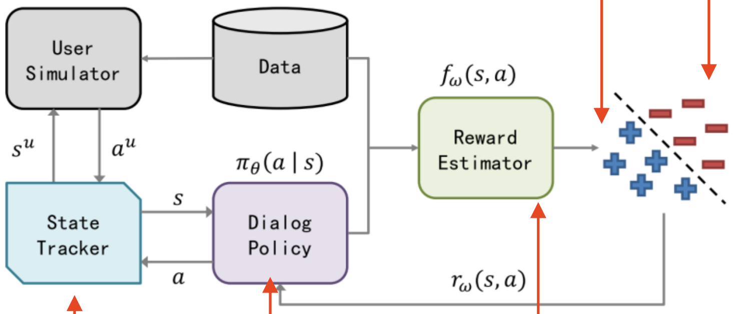
Turn-level adversarial rewards

- discriminator: policy vs. human-human
  - irrespective of success → can be done on turn level
- policy π & reward estimator f are feed-forward
  - ReLU, 1 hidden layer
- still the same process:
  - pretrain both π & f using supervised learning
  - sample dialogs using π
  - update f to distinguish sampled vs. human-human
  - update π using rewards provided by f
- using proximal policy optimization to update π
- using 2 different user simulators
  - provides more diversity

2 simulators:
- agenda/rules
- seq2seq

human-human
from data

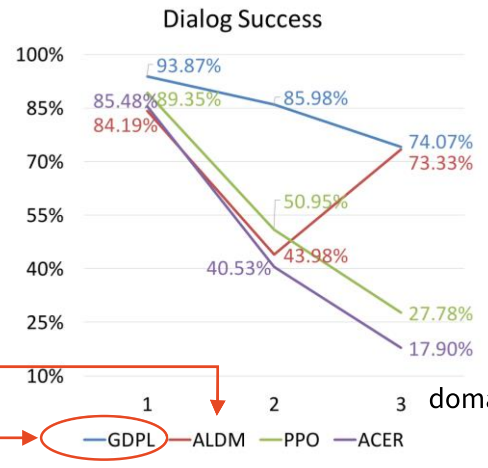
generated



rule-based

feed-forward

feed-forward



domains

(previous slide)

(this model)

(Takanobu et al., 2019) http://arxiv.org/abs/1908.10719

Alternating supervised & RL

- we can do better than just supervised pretraining
- alternate regularly
  - start with supervised more frequently
    - alleviate sparse rewards, but don't completely avoid exploring
  - later do more RL
    - but don't forget what you learned by supervised learning
- options:
  - schedule supervised every N updates
  - same + increase N gradually
  - use supervised after RL does poorly (worse than baseline)
    - baseline = moving average over history + $\lambda \cdot$ std. error of the average
    - agent is less likely to be worse than baseline in later stages of learning

(Xiong et al., 2018)
http://arxiv.org/abs/1806.06187

Natural Language Generation

- conversion of system action semantics → text (in our case)
- NLG output is well-defined, but input is not:
  - DAs
  - any other semantic formalism
  - database tables
  - raw data streams
  - user model
  - dialogue history

can be any kind of knowledge representation

e.g. “user wants short answers”

e.g. for referring expressions, avoiding repetition

- general NLG objective:

  given input & communication goal,

  create accurate + natural, well-formed, human-like text

- additional NLG desired properties:
  - variation
  - simplicity
  - adaptability

NLG Subtasks (textbook pipeline)

- Inputs
- ↓ **Content/text/document planning**
  - content selection according to communication goal
  - basic structuring & ordering
- Content plan
- ↓ **Sentence planning/microplanning**
  - aggregation (facts → sentences)
  - lexical choice
  - referring expressions
- Sentence plan
- ↓ **Surface realization**
  - linearization according to grammar
  - word order, morphology
- Text

deciding **what to say**

deciding **how to say it**

typically handled by dialogue manager in dialogue systems

organizing content into sentences & merging simple sentences

e.g. *restaurant* vs. *it*

this is needed for NLG in dialogue systems

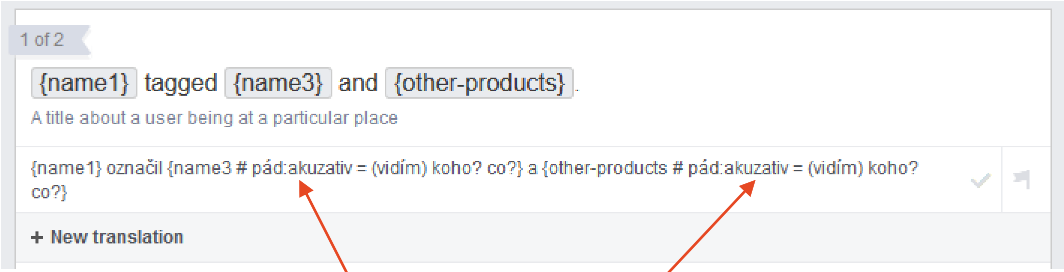
NLG Basic Approaches

- **canned text**
  - most trivial – completely hand-written prompts, no variation
  - doesn’t scale (good for DTMF phone systems)
- **templates**
  - “fill in blanks” approach
  - simple, but much more expressive – covers most common domains nicely
  - can scale if done right, still laborious
  - most production dialogue systems
- **grammars & rules**
  - grammars: mostly older research systems, realization
  - rules: mostly content & sentence planning
- **machine learning**
  - modern research systems
  - pre-neural attempts often combined with rules/grammar
  - NNs made it work much better

Template-based NLG

- Most common in dialogue systems
  - especially commercial systems
- Simple, straightforward, reliable
  - custom-tailored for the domain
  - complete control of the generated content
- Lacks generality and variation
  - difficult to maintain, expensive to scale up
- Can be enhanced with rules
  - e.g. articles, inflection of the filled-in phrases
  - template coverage/selection rules, e.g.:
    - select most concrete template
    - cover input with as few templates as possible
    - random variation

(Facebook, 2015)



(Facebook, 2019)

inflection rules

```
'iconfirm(to_stop={to_stop})&iconfirm(from_stop={from_stop})':
    "Alright, from {from_stop} to {to_stop},",

'iconfirm(to_stop={to_stop})&iconfirm(arrival_time_rel="{arrival_time_rel}")':
    "Alright, to {to_stop} in {arrival_time_rel},",

'iconfirm(arrival_time="{arrival_time}")':
    "You want to be there at {arrival_time},",

'iconfirm(arrival_time_rel="{arrival_time_rel}")':
    "You want to get there in {arrival_time_rel},",
```

(Alex public transport information rules)
https://github.com/UFAL-DSG/alex

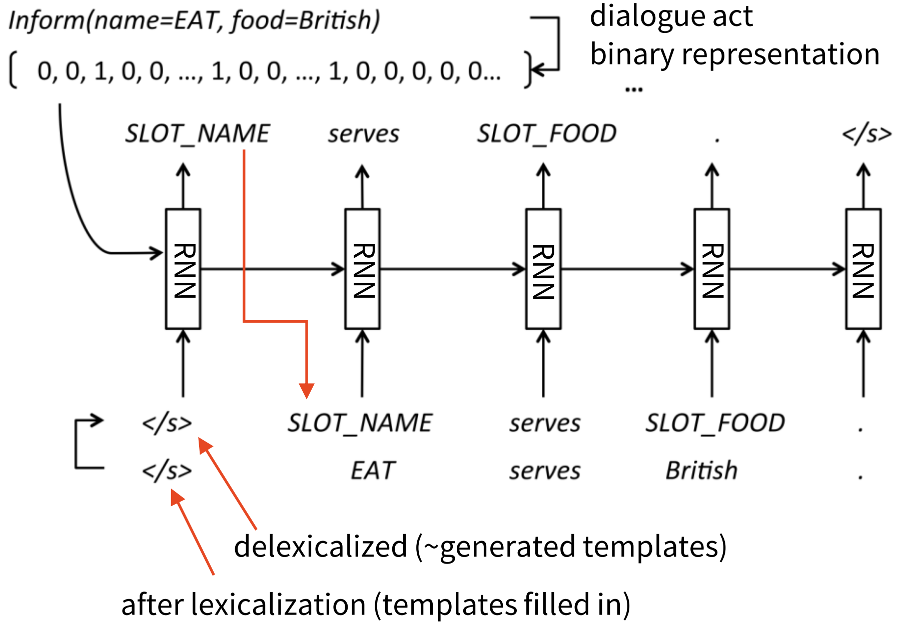
Neural End-to-End NLG: RNNLG

(Wen et al, 2015; 2016)
http://aclweb.org/anthology/D15-1199
http://arxiv.org/abs/1603.01232

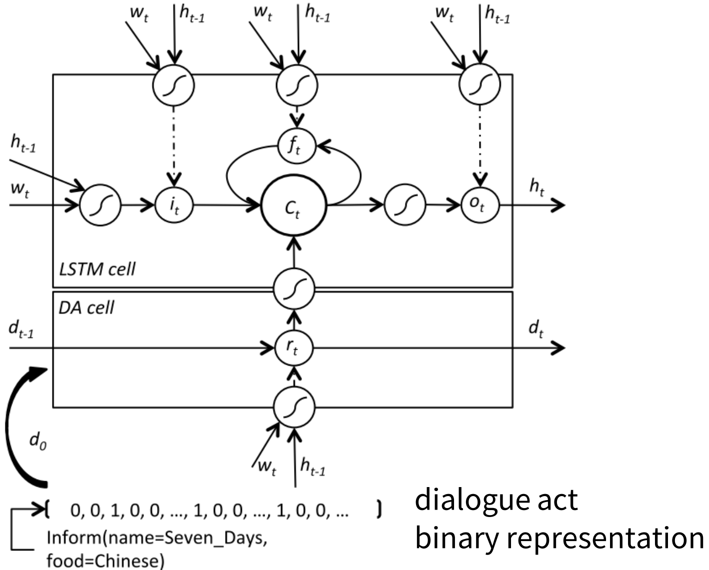
- Unlike previous, doesn't need alignments
  - no need to know which word/phrase corresponds to which slot

`name [Loch Fyne], eatType[restaurant], food[Japanese], price[cheap], familyFriendly[yes]`

Loch Fyne is a kid-friendly restaurant serving cheap Japanese food.



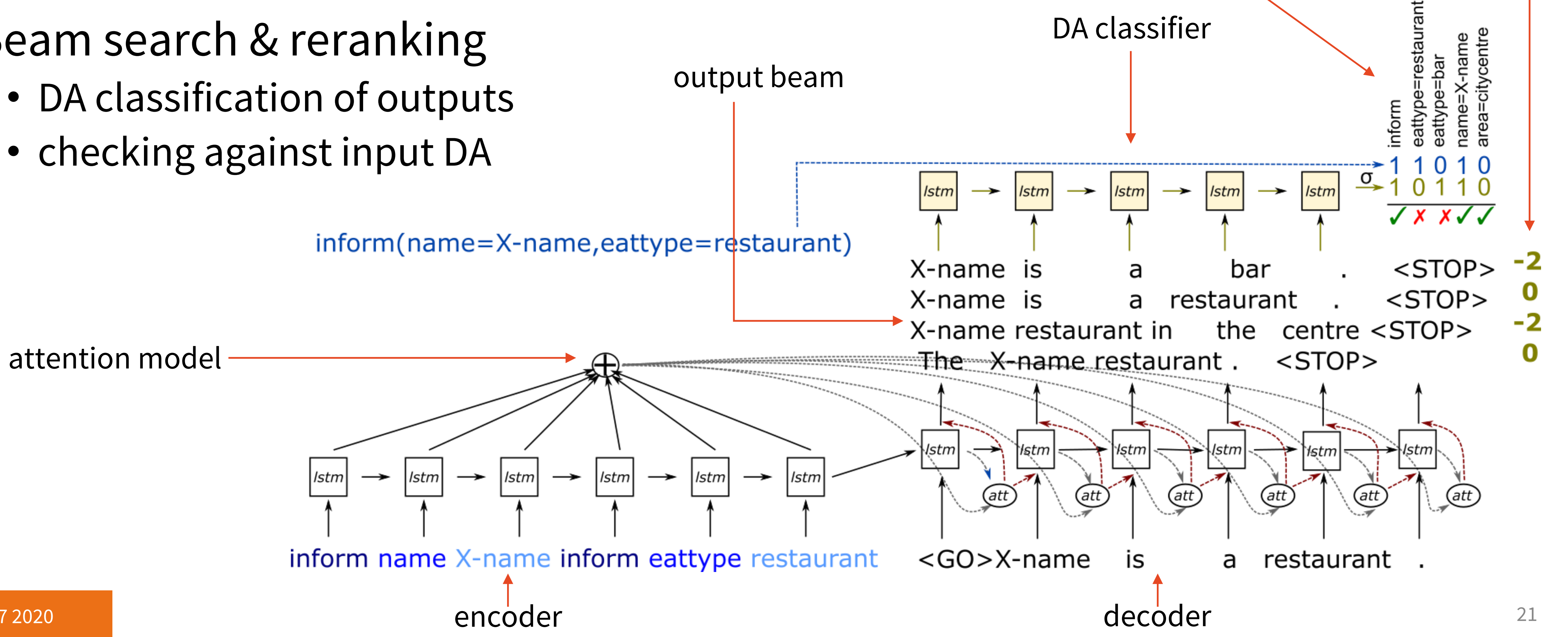
- Using RNNs, generating word-by-word
  - neural language models conditioned on DA
  - generating delexicalized texts
- input DA represented as binary vector
- Enhanced LSTM cells (SC-LSTM)
  - special part of the cell (gate) to control slot mentions



Seq2seq NLG (TGen)

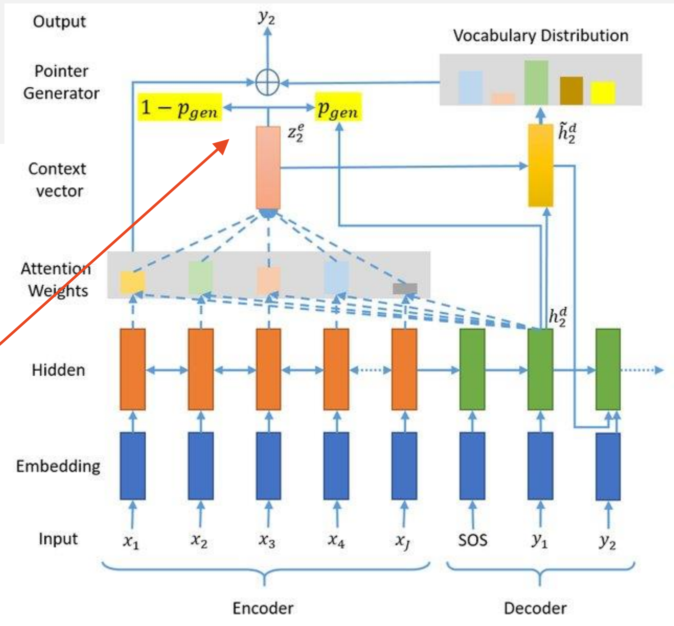
(Dušek & Jurčíček, 2016)
https://aclweb.org/anthology/P16-2008

- Standard seq2seq with attention
  - encoder – triples <DA type, slot, value>
  - decodes words (possibly delexicalized)
- Beam search & reranking
  - DA classification of outputs
  - checking against input DA

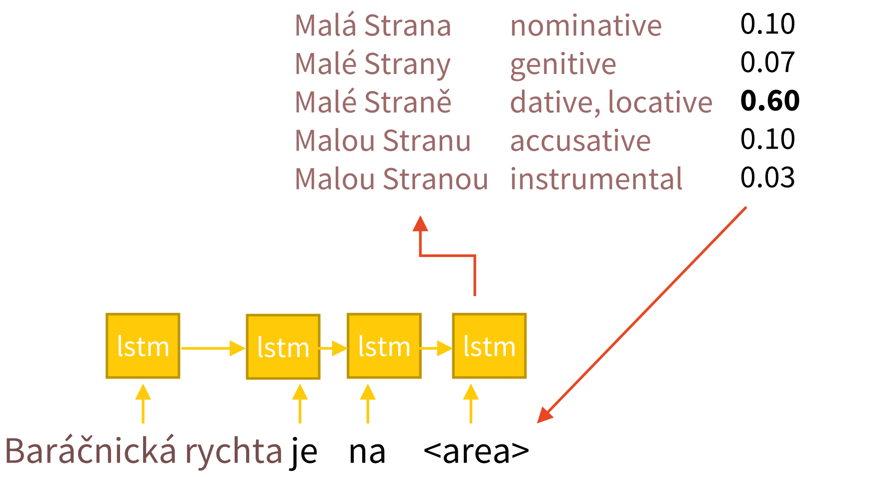


Delexicalization vs. Copy/Pointer net

- Most models still use it
  - preprocess/postprocess step – names to <placeholders>
  - generator works with template-like stuff
- Alternative – **copy mechanisms** (see NLU)
  - generate or point & copy from input
  - does away with the pre/postprocessing
- Czech & other languages with rich morphology
  - basic delexicalization or copy don't work
    - nouns need to be inflected (unlike English, where they only have 1 form)
  - basically another step needed: **inflection model**
    - one option: RNN LM



inform(name=Baráčnická rychta, area=Malá Strana)



Baráčnická rychta is in Malá Strana

(Shi et al., 2018) http://arxiv.org/abs/1812.02303
(Dušek & Jurčíček, 2019) https://arxiv.org/abs/1910.05298

Ensembling

- "two heads are better than one" – use more models & aggregate
  - common practice in neural models elsewhere in NLP
- base version: same model, different random initializations
- getting diverse predictions: use different models
  - different architectures – e.g. CNN vs. LSTM encoder
  - different data – **diverse ensembling**
    - cluster training data & train different models on different portions
  - clustering & training can be done jointly:
    - assign into groups randomly/train k models for 1 iteration
    - check prob. of each training instance under each model
    - reassign to model that predicts it with highest probability

iterate until assignments converge

(Juraska et al., 2018) http://arxiv.org/abs/1805.06553
(Gehrmann et al., 2018) https://www.aclweb.org/anthology/W18-6505

Ensembling

- combine predictions from multiple models:
  - just use the model that's best on development data
    - won't give diverse outputs, but may give better quality
  - compose n-best list from predictions of all models
    - n-best lists are more diverse
    - assuming reranking (e.g. checking against input DA)
  - vote on the next word at each step / average predicted word distributions
    - & force-decode chosen word with all models
    - this is rather slow
    - might not even work:
      - each model may expect different sentence structures, combination can be incoherent

Problems with neural NLG

(Dušek et al., 2019)
http://arxiv.org/abs/1901.07931

- Checking the semantics
  - neural models tend to forget / hallucinate (make up irrelevant stuff)
  - reranking works currently best to mitigate this, but it's not perfect
- Delexicalization needed (at least some slots)
  - otherwise the data would be too sparse
  - alternative: copy mechanisms
- Diversity & complexity of outputs
  - still can't match humans by far
  - needs specific tricks to improve this
    - vanilla seq2seq models tend to produce repetitive outputs
- Still more hassle than writing up templates 😏

open sets, verbatim on the output
(e.g., restaurant/area names)

(Puzikov & Gurevych, 2018)
https://www.aclweb.org/anthology/W18-6557

Summary

- Policy optimization
  - optimizing directly (**Policy Gradient Theorem**)
  - REINFORCE = Monte Carlo policy gradients
  - advantage =return – baseline
  - policy gradients actor-critic = REINFORCE + TD + state value estimates
  - ACER (actor-critic with experience replay) + extensions
- RL **rewards**: critical for good performance & can be (partially) learned
- **NLG**: system DA → text
  - templates work pretty well
  - **seq2seq** & similar = best data-driven
    - problems: hallucination, not enough diversity
    - fixes: reranking, delexicalization/copy nets, ensembling

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

No labs today (project questions?)
Topic deadline – today!

No class next week (holiday)

24 November: rest of NLG
+ hints on your experiments

References/Inspiration/Further:

- Matiisen (2015): Demystifying Deep Reinforcement Learning: https://neuro.cs.ut.ee/demystifying-deep-reinforcement-learning/
- Karpathy (2016): Deep Reinforcement Learning – Pong From Pixels: http://karpathy.github.io/2016/05/31/rl/
- David Silver's course on RL (UCL): http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html
- Sutton & Barto (2018): Reinforcement Learning: An Introduction (2nd ed.): http://incompleteideas.net/book/the-book.html
- Milan Straka's course on RL (Charles University): http://ufal.mff.cuni.cz/courses/npfl122/
- 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