Statistical Dialogue Systems

NPFL099 Statistické Dialogové systémy

7. Dialogue Policy (2) & Natural Language Generation

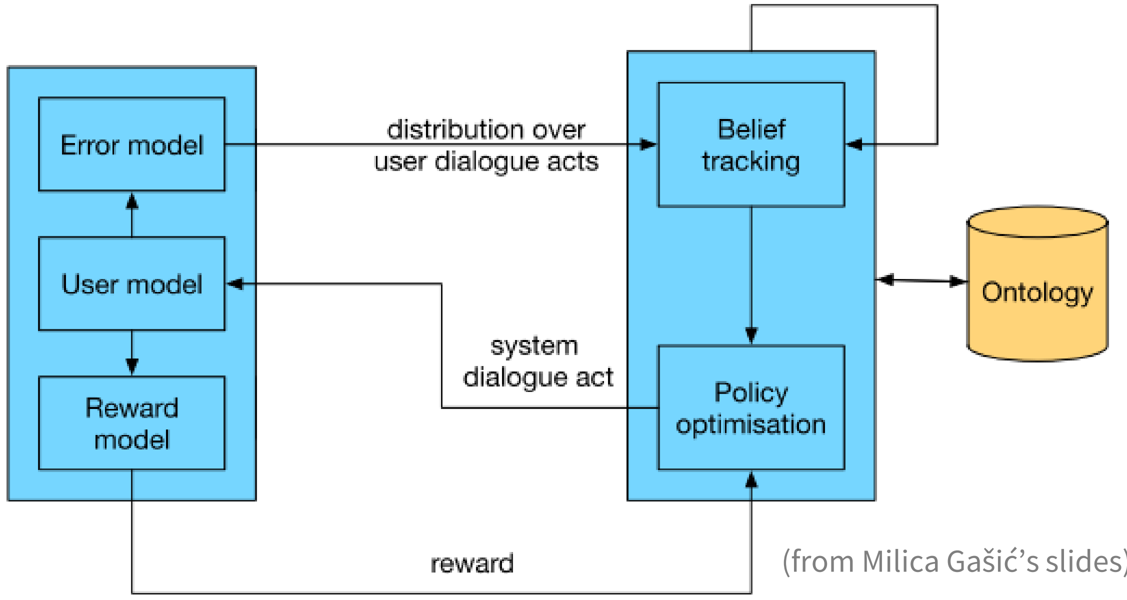
Ondřej Dušek & Vojtěch Hudeček

http://ufal.cz/npfl099

28. 11. 2019

Simulated Users

- We can’t really learn just from static datasets
  - on-policy algorithms don’t work
  - data might not reflect our newly learned behaviour
- RL needs a lot of data, more than real people would handle
  - 1k-100k’s dialogues used for training, depending on method
- solution: **user simulation**
  - basically another DS/DM
  - (typically) working on DA level
  - errors injected to simulate ASR/NLU
- approaches:
  - rule-based (frames/agenda)
  - n-grams
  - MLE policy from data
  - combination (best!)



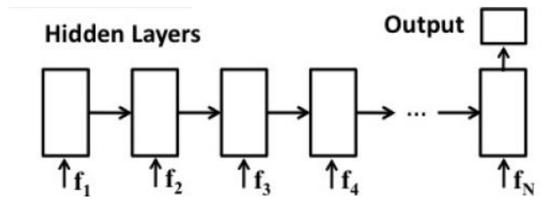
(from Milica Gašić’s slides)

Rewards in RL

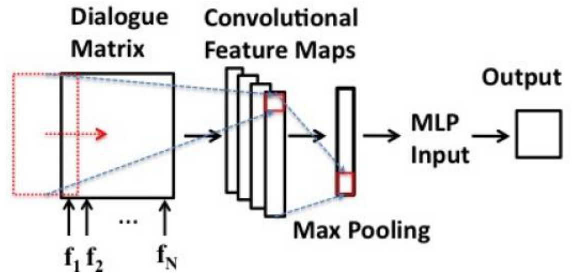
- Reward function is critical for successful learning
- Handcrafting is not ideal
  - domain knowledge typically needed to detect dialogue success
  - need simulated or paid users,
    can’t learn from users without knowing their task
  - paid users often fail to follow pre-set goals
- Having users provide 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



turn features



turn features

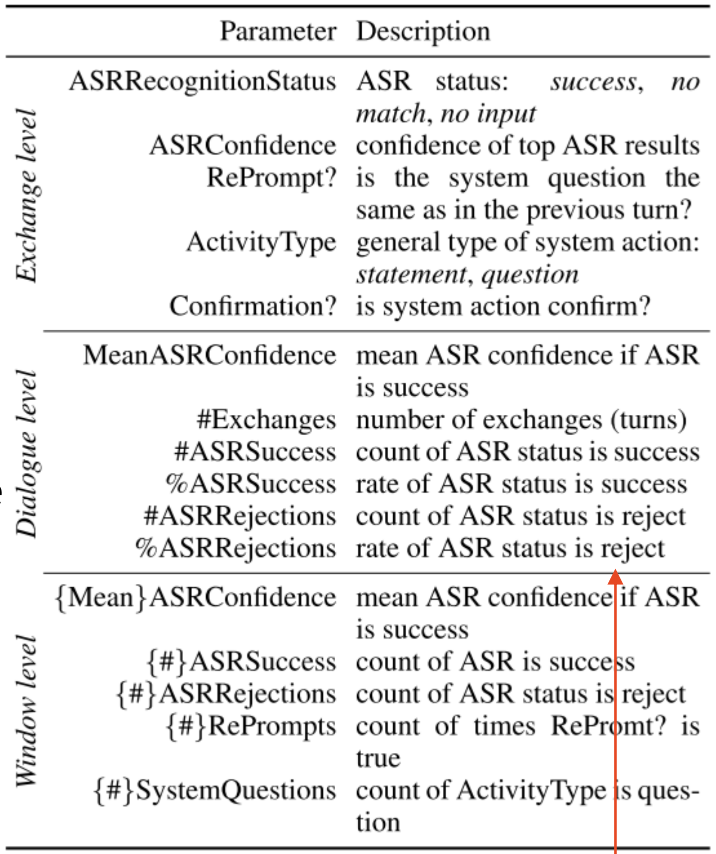
(Su et al., 2015)
http://arxiv.org/abs/1508.03386

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



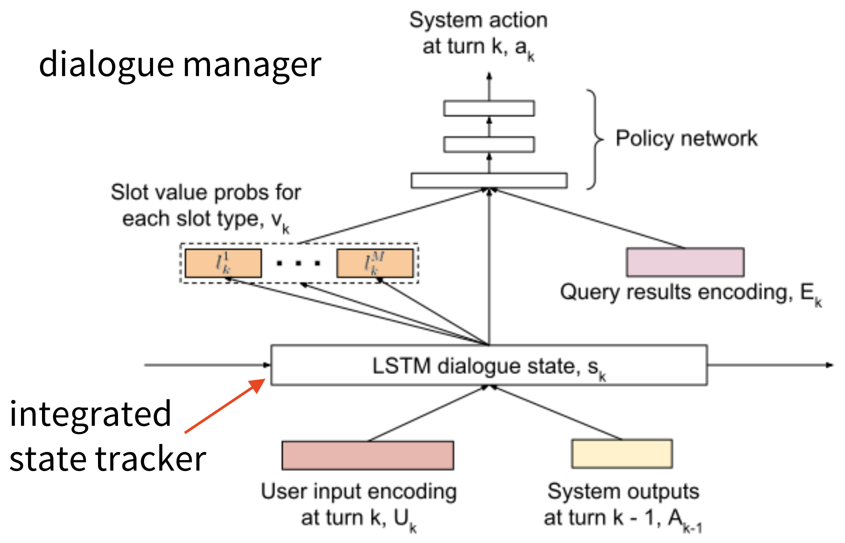
		Parameter	Description
current turn	*Exchange level*	ASRRecognitionStatus	ASR status: *success*, *no match*, *no input*
		ASRConfidence	confidence of top ASR results
		RePrompt?	is the system question the same as in the previous turn?
		ActivityType	general type of system action: *statement*, *question*
		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

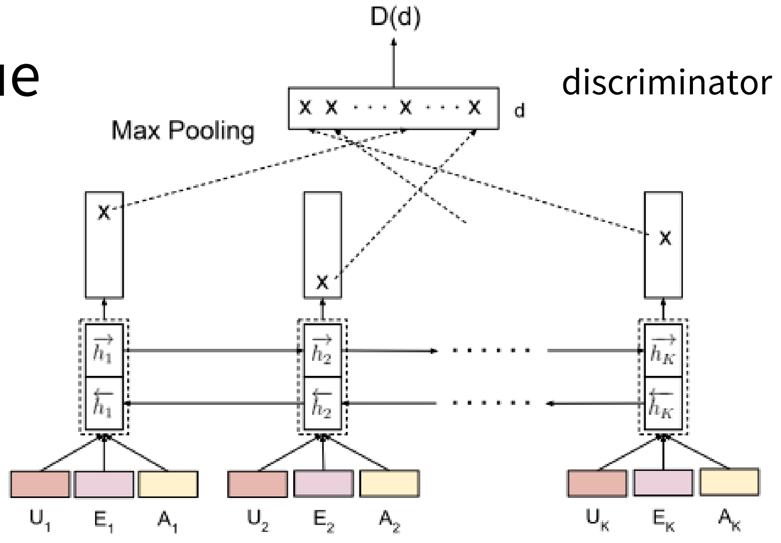
(Liu & Lane, 2018) http://arxiv.org/abs/1805.11762

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 vs. random 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 & add to human data, train to classify success vs. random



dialogue manager
System action at turn k, a_k
Policy network
Slot value probs for each slot type, v_k
l_k^1
l_k^M
Query results encoding, E_k
LSTM dialogue state, s_k
integrated state tracker
User input encoding at turn k, U_k
System outputs at turn k - 1, A_{k-1}



D(d)
discriminator
Max Pooling
d
$\vec{h}_1$ $\overleftarrow{h}_1$
$\vec{h}_2$ $\overleftarrow{h}_2$
$\vec{h}_K$ $\overleftarrow{h}_K$
U_1 E_1 A_1
U_2 E_2 A_2
U_K E_K A_K

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

Reward as discriminator

- DSTC2 data
- comparing rewards

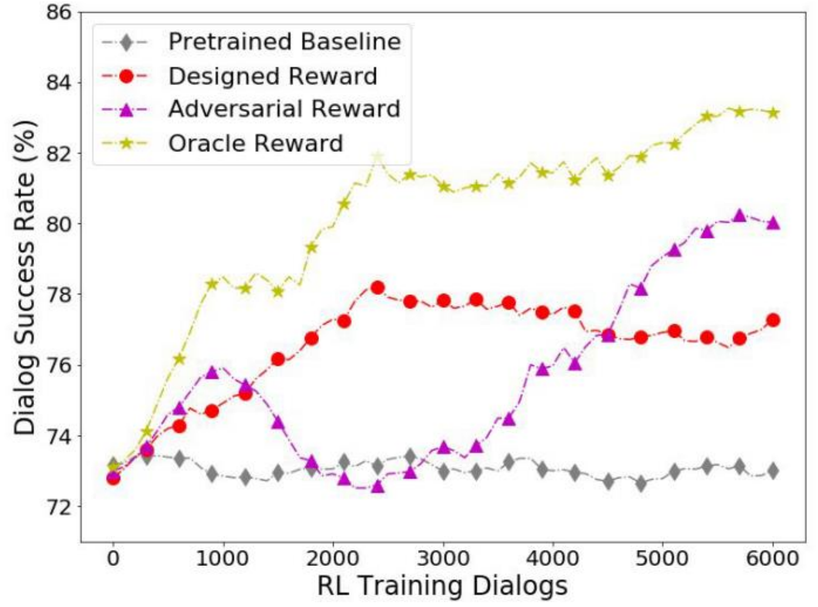
does not copy the actual dialogue success

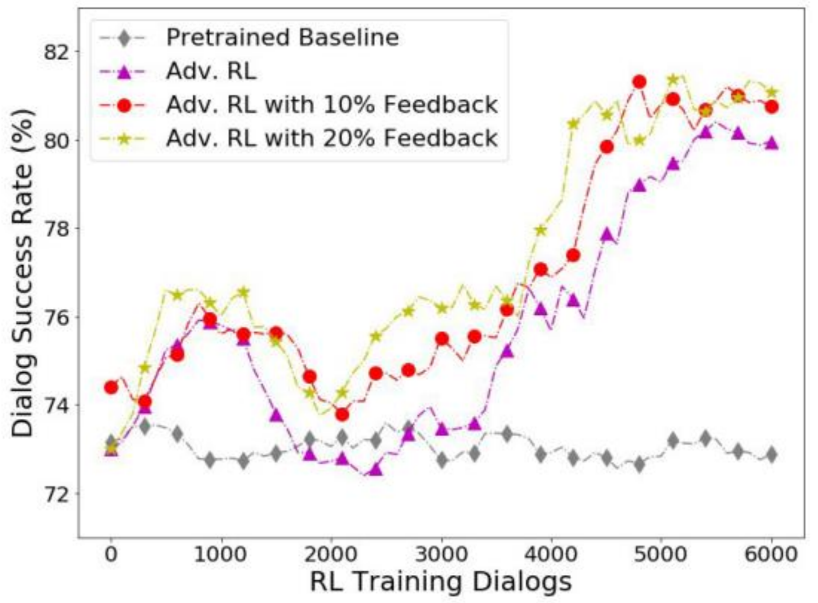
known goal only

  - **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
- 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

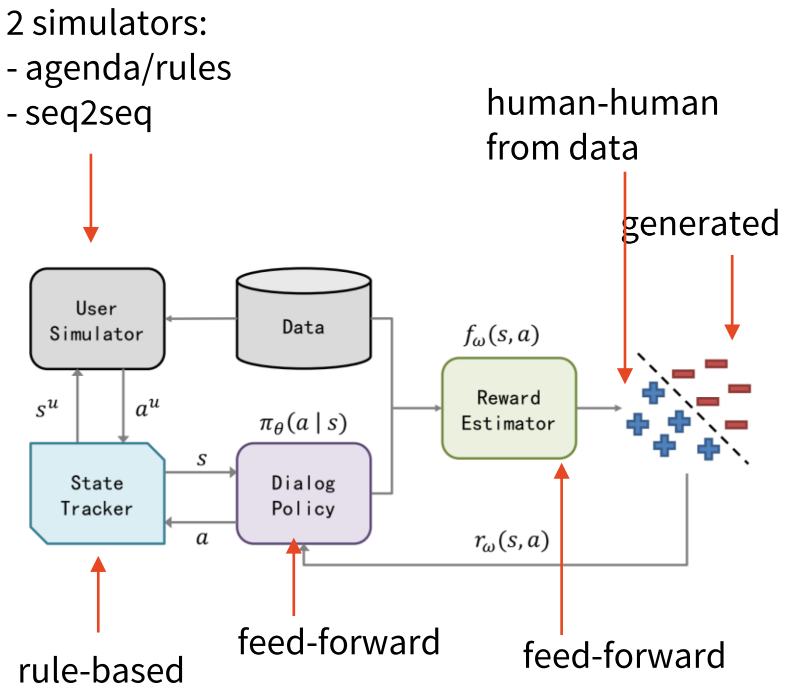


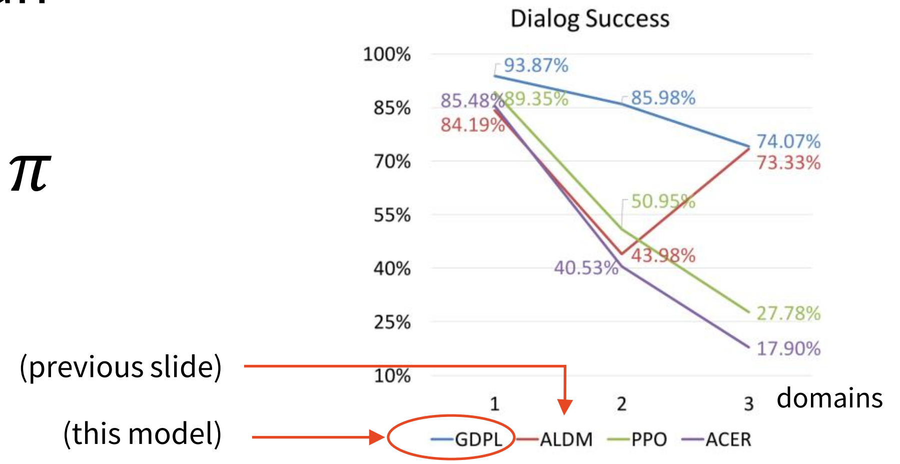


(Takanobu et al., 2019) http://arxiv.org/abs/1908.10719

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





Alternating supervised & RL

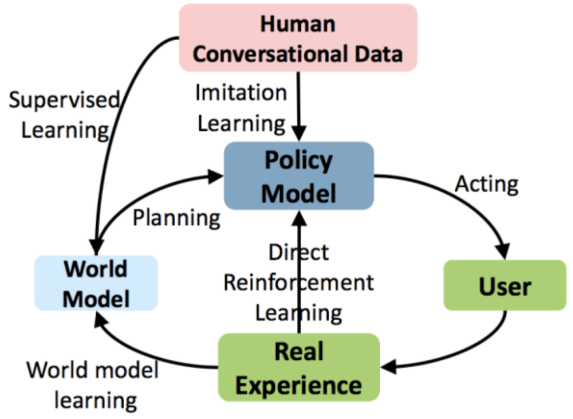
(Xiong et al., 2018)
http://arxiv.org/abs/1806.06187

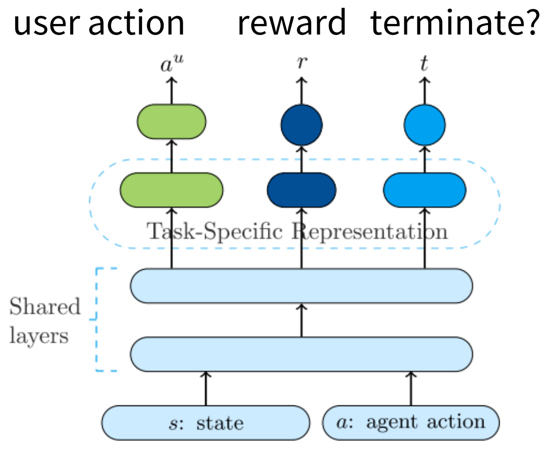
- we can do better than just supervised pretraining
- alternate regularly
  - start with supervised more frequently
    - alleviate sparse rewards, but don't completely avoid exploring
  - later do more RL
    - but don't forget what you learned by supervised learning
- options:
  - schedule supervised 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

(Peng et al., 2018) https://www.aclweb.org/anthology/P18-1203
(Su et al., 2018) https://www.aclweb.org/anthology/D18-1416

Deep Dyna-Q: learning from humans & simulator

- humans are costly, simulators are inaccurate
- ⇒ learn from both, improve simulator as you go
  - direct RL = learn from users
  - world model learning = improve internal simulator
    - supervised, based on previous dialogues with users
  - planning = learn from simulator
- DQN, feed-forward policy
- simulator: feed-forward multi-task net
  - draw a goal uniformly at the start ← movie booking: name, date, # tickets etc.
  - predict actions, rewards, termination
  - use K simulated ("planning") dialogues per 1 real
- discriminative DDQ: only use a simulated dialogue if it looks real (according to a discriminator)

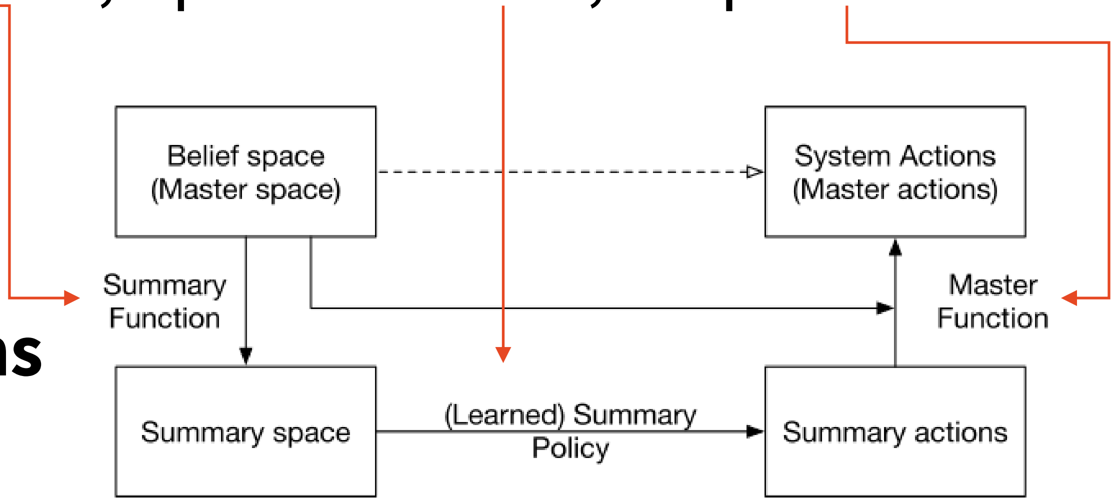




internal simulator = world model

Summary Space

- for a typical DS, the belief state is too large to make RL tractable
- solution: map state into a reduced space, optimize there, map back
- reduced space = **summary space**
  - handcrafted state features
  - e.g. top slots, # found, slots confirmed…
- reduced action set = **summary actions**
  - e.g. just DA types (*inform, confirm, reject*)
  - remove actions that are not applicable
  - with handcrafted mapping to real actions
- state is still tracked in original space
  - we still need the complete information for accurate updates

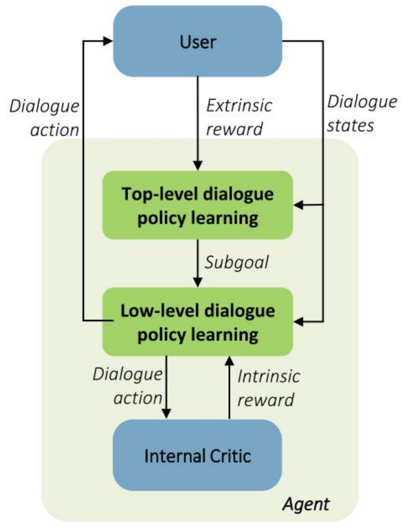


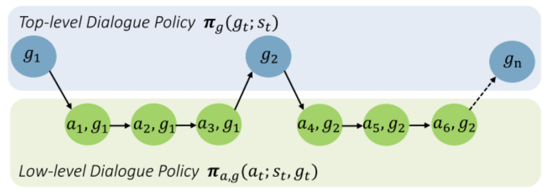
(from Milica Gašić's slides)

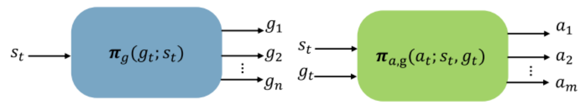
Hierarchical RL

(Peng et al., 2017)
http://aclweb.org/anthology/D17-1237

- good for multiple subtasks
  - e.g. book a flight to London and a hotel for the same day, close to the airport
- top-level policy: select subtask g_i
- low-level policy: actions a_{j,g_i} to complete subtask g_i
  - given initiation/termination conditions
    - keeps on track until terminal state is reached
  - shared by all subtasks (subtask=parameter)
  - internal critic (=prob. that subtask is solved)
- global state tracker (integrates information from subtasks)



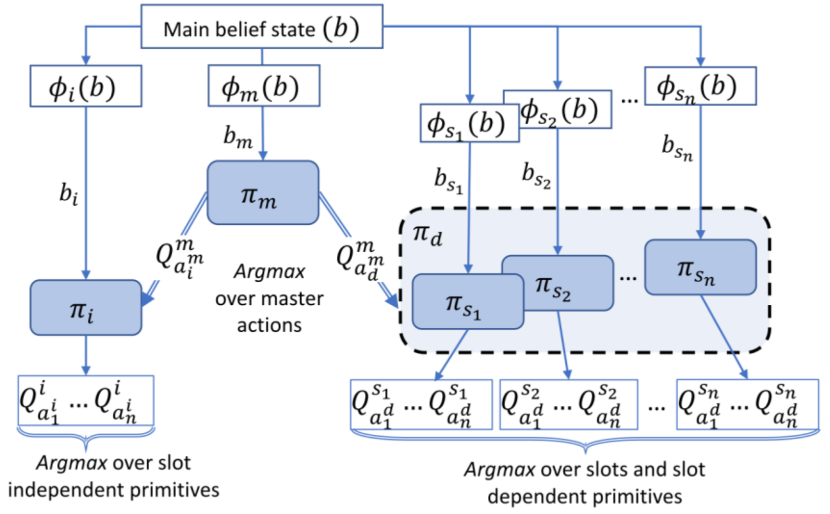

Top-level Dialogue Policy $\pi_g(g_t; s_t)$
g_1
g_2
g_n
a_1, g_1
a_2, g_1
a_3, g_1
a_4, g_2
a_5, g_2
a_6, g_2
Low-level Dialogue Policy $\pi_{a,g}(a_t; s_t, g_t)$



top-level Q-network low-level Q-network

Feudal RL

(Casanueva et al., 2018)
http://arxiv.org/abs/1803.03232



Main belief state (b)
$\phi_i(b)$
$\phi_m(b)$
$\phi_{s_1}(b)$
$\phi_{s_2}(b)$
$\phi_{s_n}(b)$
b_i
b_m
b_{s_1}
b_{s_2}
b_{s_n}
π_m
π_i
π_d
π_{s_1}
π_{s_2}
π_{s_n}
$Q^m_{a^m_i}$
$Q^m_{a^m_d}$
Argmax over master actions
$Q^i_{a^i_1} \dots Q^i_{a^i_n}$
$Q^{s_1}_{a^d_1} \dots Q^{s_1}_{a^d_n}$
$Q^{s_2}_{a^d_1} \dots Q^{s_2}_{a^d_n}$
$Q^{s_n}_{a^d_1} \dots Q^{s_n}_{a^d_n}$
Argmax over slot independent primitives
Argmax over slots and slot dependent primitives

hello, inform

request, confirm

inform = "inform over all slots"

- spatial (slot-based) split instead of temporal
  - doesn't need defined subtasks & sub-rewards
- belief state abstraction
  - handcrafted (could be neural nets)
  - supports sharing parameters across domains
- two-step action selection:
  1) master action: "slot-dependent or not"?
     - master policy
  2) primitive action
     a) slot-independent policy
     b) slot-specific policies (with shared parameters, distinguished only by belief state)
        - chooses max. Q for all slot-action pairs – involves choosing the slot
  - all trained using the same global reward signal

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

  $\}$ can be any kind of knowledge representation

  - user model $\leftarrow$ e.g. “user wants short answers”
  - dialogue history $\leftarrow$ 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

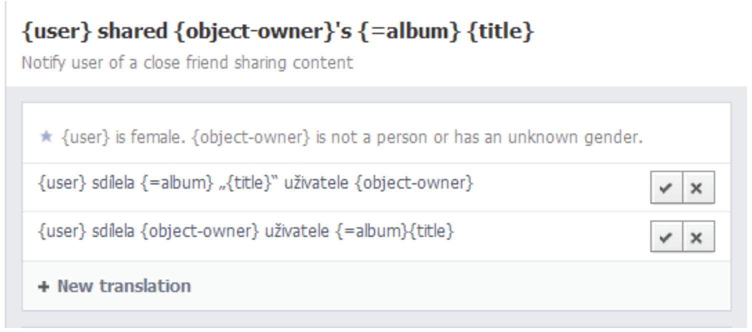
NLG Basic Approaches

- **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
  - RNNs 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)



{user} shared {object-owner}'s {=album} {title}

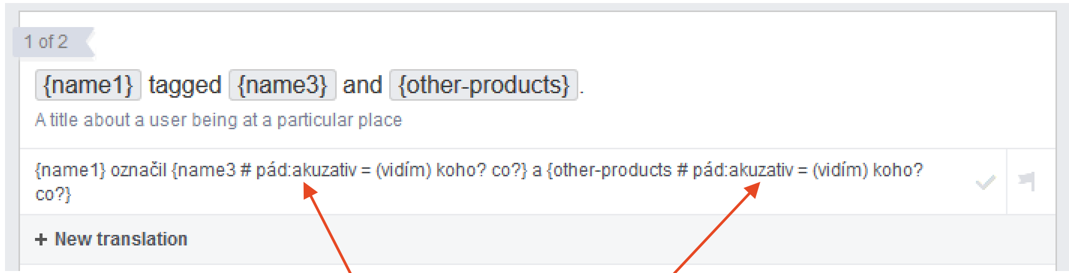
Notify user of a close friend sharing content

★ {user} is female. {object-owner} is not a person or has an unknown gender.

{user} sdílela {=album} „{title}" uživatele {object-owner} ✓ ✕

{user} sdílela {object-owner} uživatele {=album}{title} ✓ ✕

+ New translation



1 of 2

{name1} tagged {name3} and {other-products}.

A title about a user being at a particular place

{name1} označil {name3 # pád:akuzativ = (vidím) koho? co?} a {other-products # pád:akuzativ = (vidím) koho? co?}

+ New translation

(Facebook, 2019)

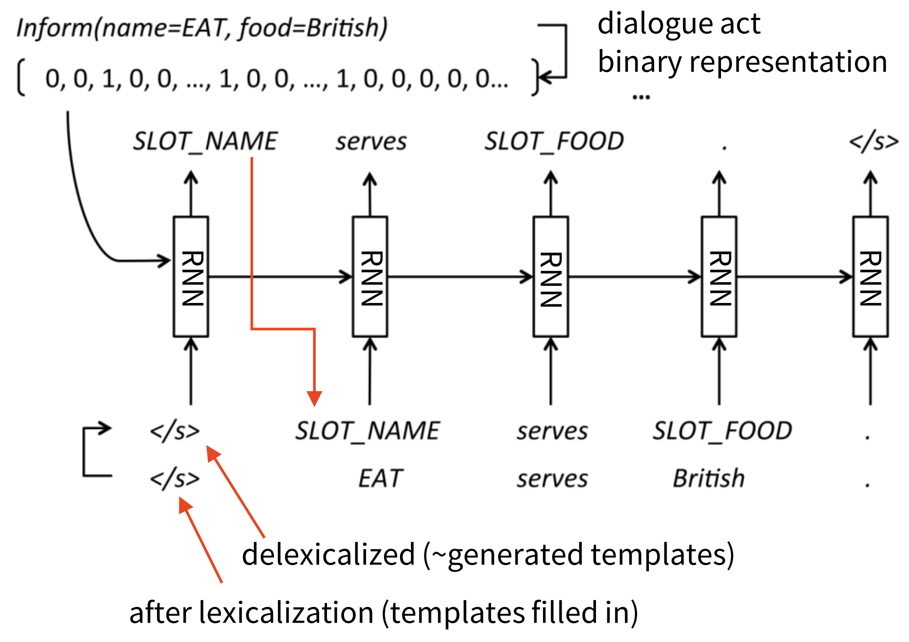
inflection rules

```
'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

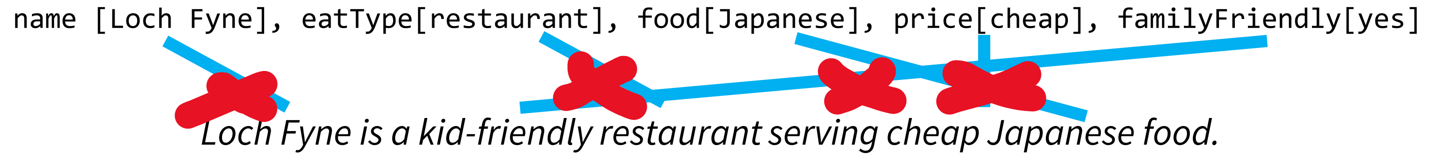
Neural End-to-End NLG: RNNLG

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



Inform(name=EAT, food=British)
[0, 0, 1, 0, 0, ..., 1, 0, 0, ..., 1, 0, 0, 0, 0, 0...]
dialogue act binary representation
...
SLOT_NAME
serves
SLOT_FOOD
.
</s>
RNN
</s>
SLOT_NAME
serves
SLOT_FOOD
.
</s>
EAT
serves
British
.
delexicalized (~generated templates)
after lexicalization (templates filled in)

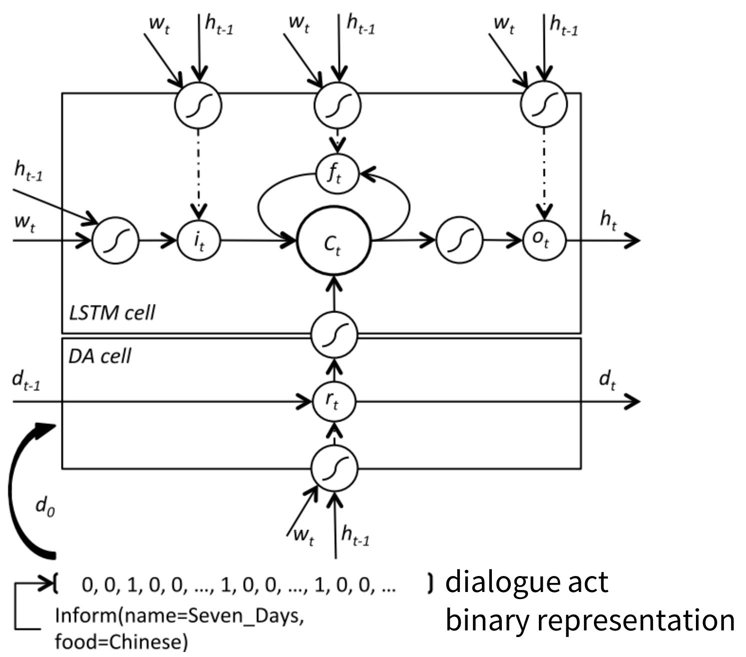
- Unlike previous, doesn't need alignments
  - no need to know which word/phrase corresponds to which slot



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

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

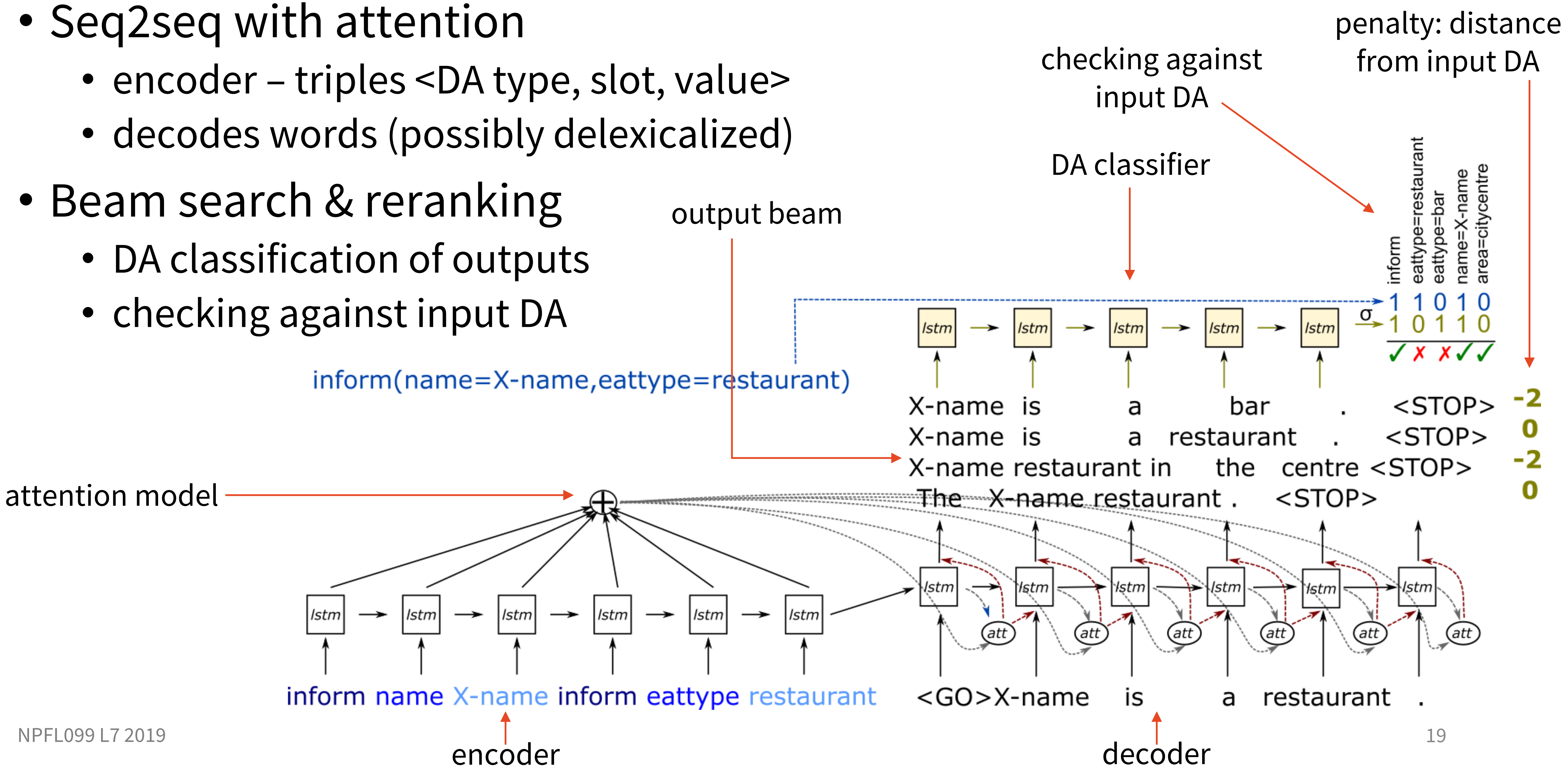
- Using RNNs, generating word-by-word
  - neural language models conditioned on DA
  - generating delexicalized texts
- input DA represented as binary vector
- Enhanced LSTM cells (SC-LSTM)
  - special part of the cell (gate) to control slot mentions



Seq2seq NLG (TGen)

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

- Seq2seq with attention
  - encoder – triples <DA type, slot, value>
  - decodes words (possibly delexicalized)
- Beam search & reranking
  - DA classification of outputs
  - checking against input DA



penalty: distance from input DA
checking against input DA
DA classifier
output beam
inform
eattype=restaurant
eattype=bar
name=X-name
area=citycentre
1 1 0 1 0
σ
1 0 1 1 0
✓ ✗ ✗ ✓ ✓
lstm
inform(name=X-name,eattype=restaurant)
X-name is a bar . <STOP>
X-name is a restaurant . <STOP>
X-name restaurant in the centre <STOP>
The X-name restaurant . <STOP>
-2
0
-2
0
attention model
att
inform name X-name inform eattype restaurant
<GO> X-name is a restaurant .
encoder
decoder

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

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

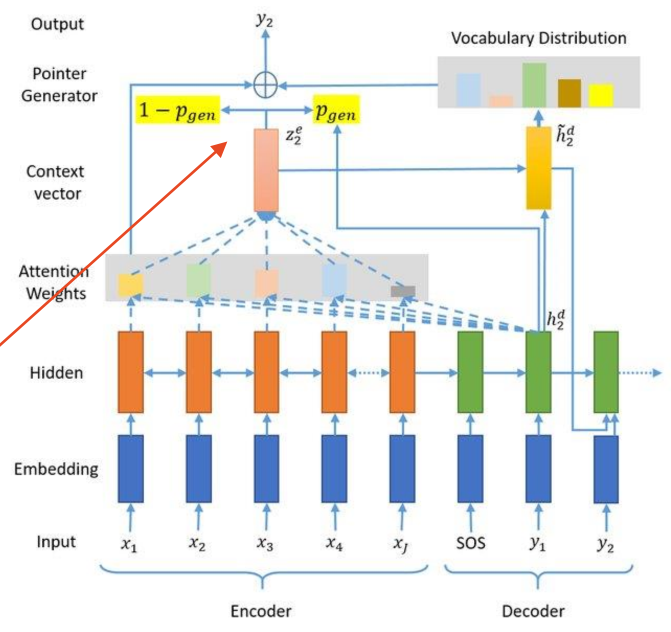
- Diversity & complexity of outputs
  - still can't match humans by far
  - needs specific tricks to improve this
    - vanilla seq2seq models tend to produce repetitive outputs
- Still more hassle than writing up templates 😏

Delexicalization

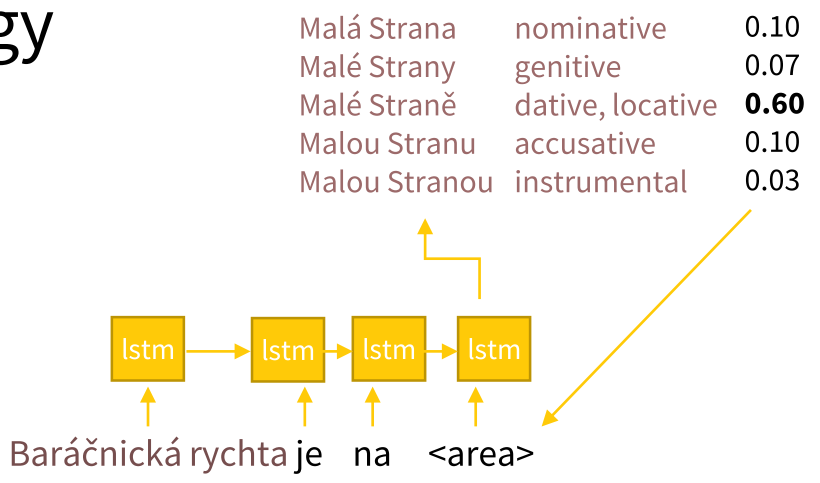
(Shi et al., 2018) http://arxiv.org/abs/1812.02303

(Dušek & Jurčíček, 2019) https://arxiv.org/abs/1910.05298

- 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)



Malá Strana	nominative	0.10
Malé Strany	genitive	0.07
Malé Straně	dative, locative	**0.60**
Malou Stranu	accusative	0.10
Malou Stranou	instrumental	0.03

Baráčnická rychta is in Malá Strana

Ensembling

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

- “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

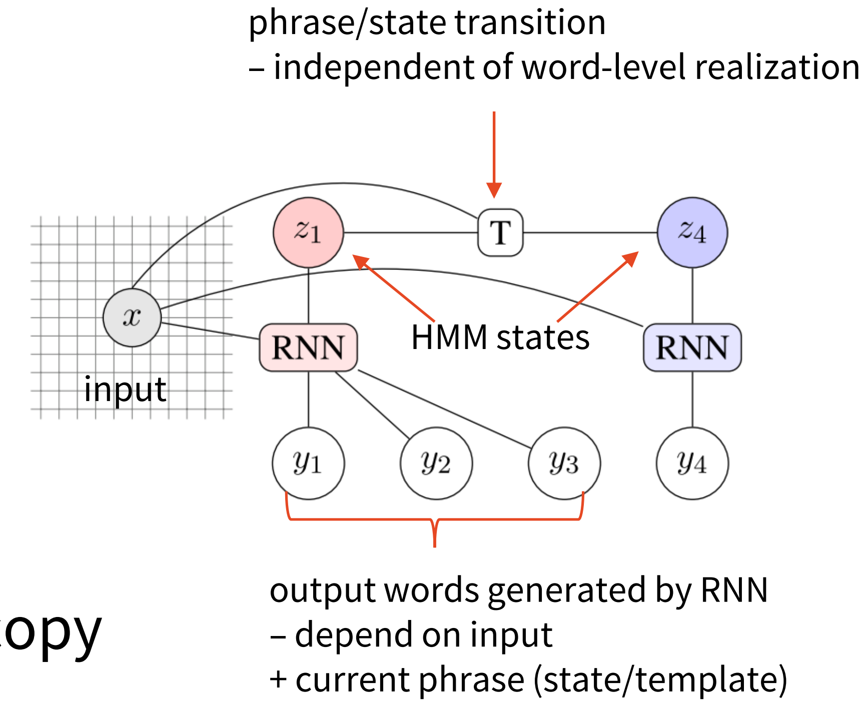
Ensembling

- 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

(Wiseman et al., 2018) http://aclweb.org/anthology/D18-1356

Hidden Semi-Markov Model

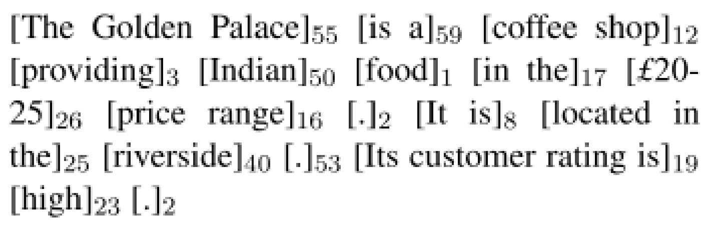
- learning latent “templates” (sequences of phrases)
  - discrete, induced during training automatically
  - provide (some) interpretation
  - can be used to condition generation
- HMM on the level of phrases + word-level RNN
  - encoder: max-pooling of item embs. + ReLU
  - transitions: softmax of
    dot prod. of state embs. + transformed inputs
  - lengths: uniform
  - emissions: RNN with attention over input items + copy
- training – backward algorithm
  - can be end-to-end differentiable


phrase/state transition
– independent of word-level realization
z_1
T
z_4
x
input
RNN
HMM states
RNN
y_1
y_2
y_3
y_4
output words generated by RNN
– depend on input
+ current phrase (state/template)

(Wiseman et al., 2018) http://aclweb.org/anthology/D18-1356

Hidden Semi-Markov Model

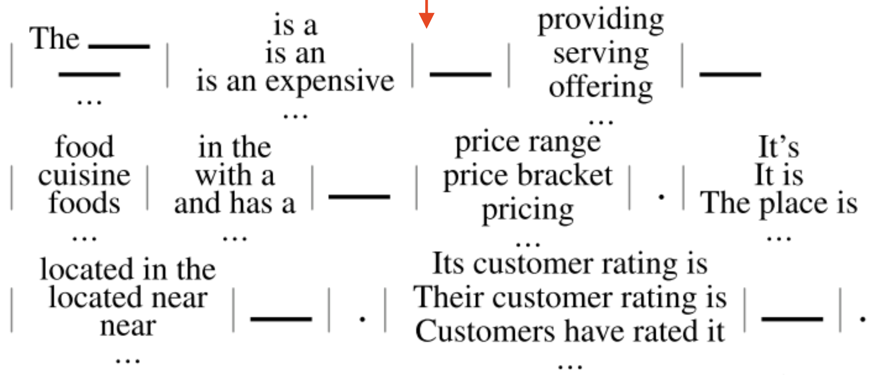
- phrases can be associated with state numbers
  - “Viterbi segmentation” on training data
  - this provides the interpretation


[The Golden Palace]$_{55}$ [is a]$_{59}$ [coffee shop]$_{12}$ [providing]$_{3}$ [Indian]$_{50}$ [food]$_{1}$ [in the]$_{17}$ [£20-25]$_{26}$ [price range]$_{16}$ [.]$_{2}$ [It is]$_{8}$ [located in the]$_{25}$ [riverside]$_{40}$ [.]$_{53}$ [Its customer rating is]$_{19}$ [high]$_{23}$ [.]$_{2}$

- generation – can do “template extraction” first
  - collect frequent templates (sequences of phrases/states) from training data
  - restrict generation to just one/some of them
    - constrained beam search (within phrases only, state transitions are given)
  - allows for diversity
    - choosing different templates each time
  - allows checking what slots are generated
- outputs not as fluent as plain seq2seq

name[_], type[_], rating[_], food[_], area[_], price[_]

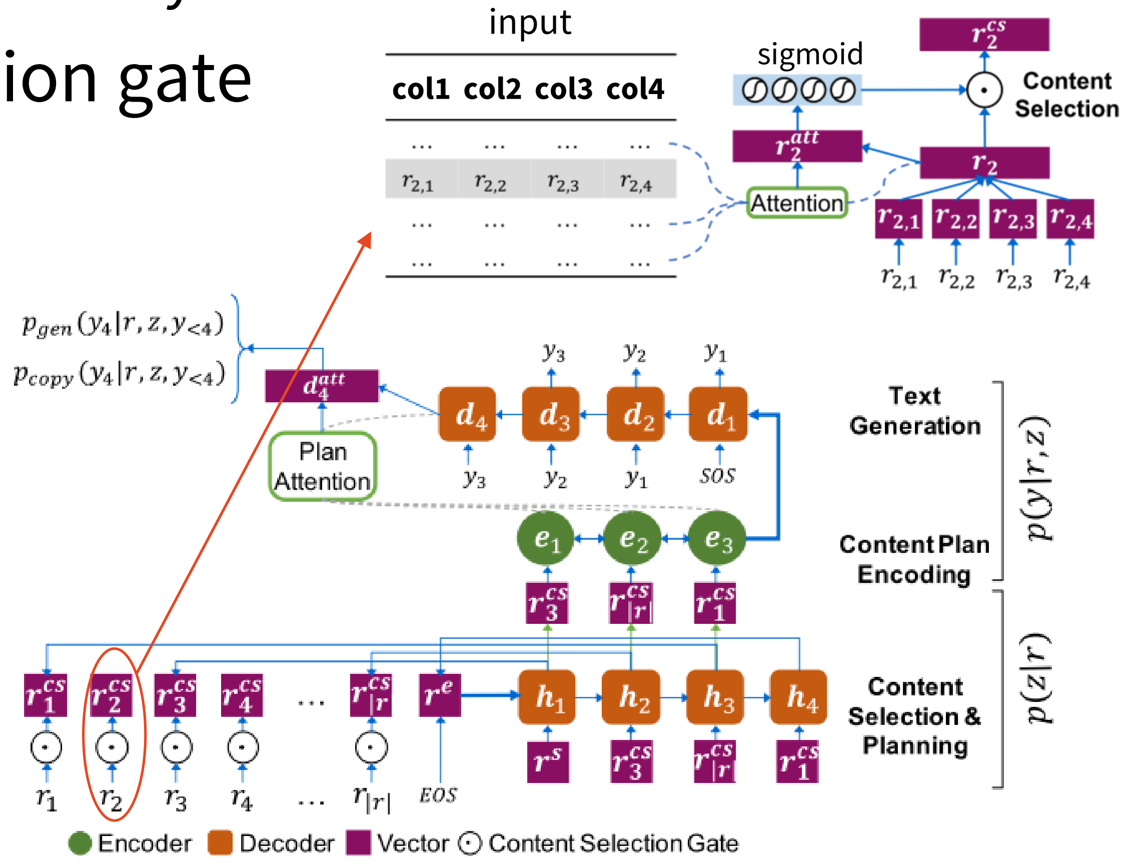
55 – 59 – 12 – 3 – 50 – 1 -17 – 26 – 16 – 2 – 8 – 25 – 40 – 53 – 19 – 23 – 2



(Puduppully et al., 2019) http://arxiv.org/abs/1809.00582

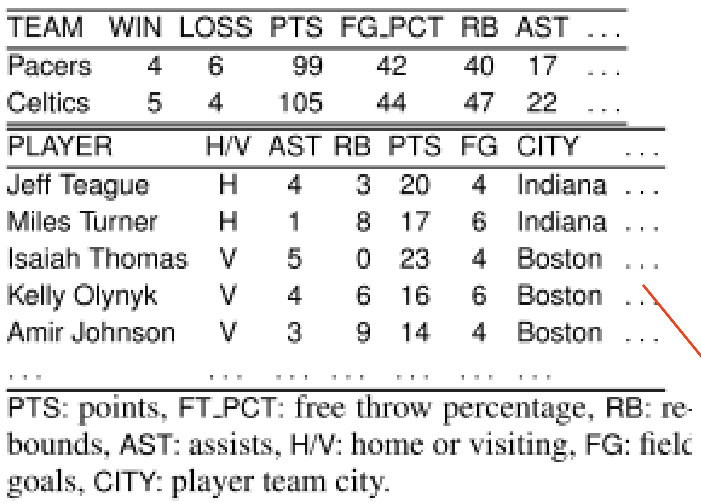
Two-step: content selection & realization

- explicit content planning step (selection & ordering)
  - designed for sports report generation – longer texts, selection needed
  - records (team / entity / type / value) → summary
- record encoder: feed-forward + attention gate
- content selection: pointer network
  - decode records with top attention
- generation: pointer-generator net
  - generating/copying tokens
  - attending over selected records
- two-stage training
  - selected records extracted automatically from texts



Two-step: content selection & realization

source statistics

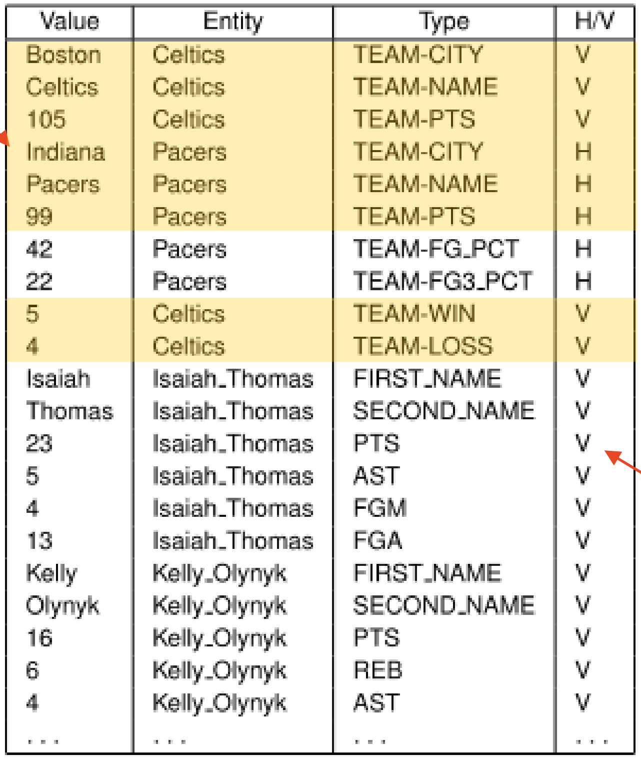


TEAM	WIN	LOSS	PTS	FG_PCT	RB	AST	. . .
Pacers	4	6	99	42	40	17	. . .
Celtics	5	4	105	44	47	22	. . .

PLAYER	H/V	AST	RB	PTS	FG	CITY	. . .
Jeff Teague	H	4	3	20	4	Indiana	. . .
Miles Turner	H	1	8	17	6	Indiana	. . .
Isaiah Thomas	V	5	0	23	4	Boston	. . .
Kelly Olynyk	V	4	6	16	6	Boston	. . .
Amir Johnson	V	3	9	14	4	Boston	. . .
. . .	. . .	. . .	. . .	. . .	. . .	. . .	

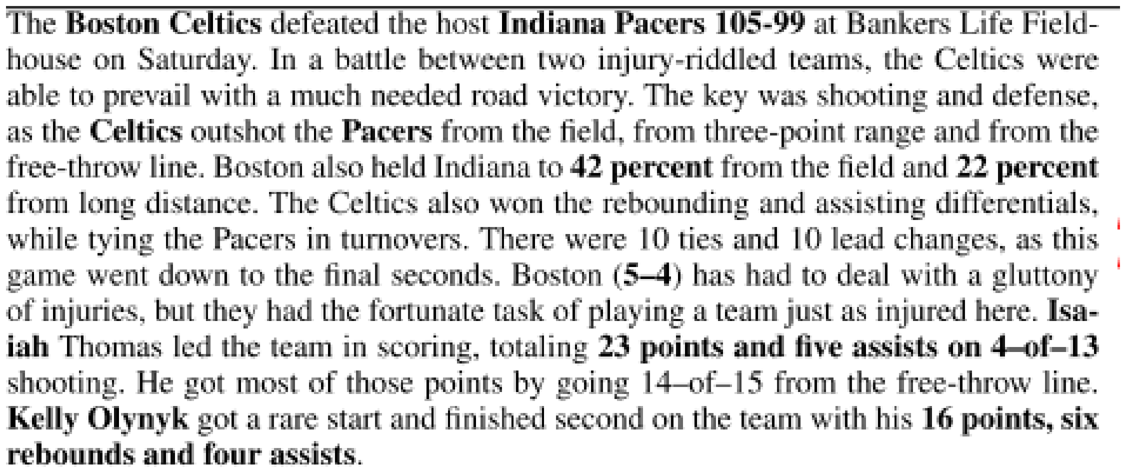
PTS: points, FT_PCT: free throw percentage, RB: rebounds, AST: assists, H/V: home or visiting, FG: fielc goals, CITY: player team city.

content plan
- automatic conversion
- content selection is done here (shown for 1st sentence)



Value	Entity	Type	H/V
Boston	Celtics	TEAM-CITY	V
Celtics	Celtics	TEAM-NAME	V
105	Celtics	TEAM-PTS	V
Indiana	Pacers	TEAM-CITY	H
Pacers	Pacers	TEAM-NAME	H
99	Pacers	TEAM-PTS	H
42	Pacers	TEAM-FG_PCT	H
22	Pacers	TEAM-FG3_PCT	H
5	Celtics	TEAM-WIN	V
4	Celtics	TEAM-LOSS	V
Isaiah	Isaiah_Thomas	FIRST_NAME	V
Thomas	Isaiah_Thomas	SECOND_NAME	V
23	Isaiah_Thomas	PTS	V
5	Isaiah_Thomas	AST	V
4	Isaiah_Thomas	FGM	V
13	Isaiah_Thomas	FGA	V
Kelly	Kelly_Olynyk	FIRST_NAME	V
Olynyk	Kelly_Olynyk	SECOND_NAME	V
16	Kelly_Olynyk	PTS	V
6	Kelly_Olynyk	REB	V
4	Kelly_Olynyk	AST	V
. . .	. . .	. . .	. . .

target text



The **Boston Celtics** defeated the host **Indiana Pacers 105-99** at Bankers Life Fieldhouse on Saturday. In a battle between two injury-riddled teams, the Celtics were able to prevail with a much needed road victory. The key was shooting and defense, as the **Celtics** outshot the **Pacers** from the field, from three-point range and from the free-throw line. Boston also held Indiana to **42 percent** from the field and **22 percent** from long distance. The Celtics also won the rebounding and assisting differentials, while tying the Pacers in turnovers. There were 10 ties and 10 lead changes, as this game went down to the final seconds. Boston (**5–4**) has had to deal with a gluttony of injuries, but they had the fortunate task of playing a team just as injured here. **Isaiah** Thomas led the team in scoring, totaling **23 points and five assists on 4–of–13** shooting. He got most of those points by going 14–of–15 from the free-throw line. **Kelly Olynyk** got a rare start and finished second on the team with his **16 points, six rebounds and four assists**.

team ID – home/visiting

Two-step: content planning & realization

- create explicit text plans by aggregating inputs
  - RDF triples → list of trees (one per sentence)
    - joining + ordering (←→)
  - create all possibilities + rank
    - product of experts for given features:
      - individual arrow directions
      - % of reversed
      - sentence split + # of triplets in each
      - relation bigrams (e.g. p(capital|residence))
    - can select the best plan, or a random highly-rated one
      - most plans beyond a certain threshold are fine
  - training plans extracted automatically
    - text is consistent with a plan if it has the right sentence split & assignment + order of entities
    - relations are not checked (this is much harder than for entities)
- sentence-by-sentence generation: pointer-generator net
  - more faithful than generating everything in one step

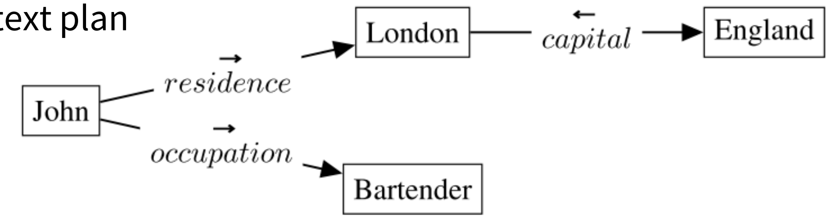
Π of cond. distributions →

input RDF

John | residence | London
John | occupation | bartender
England | capital | London

↓

text plan



↓

John lives in London, the capital of England, and works as a bartender.

Realizing from Trees

(Balakrishnan et al., 2019)
http://arxiv.org/abs/1906.07220

- Input: tree-shaped MRs
  - hierarchy: discourse relation ↓ dialogue act ↓ slot
  - can be automatically induced, much flatter than usual syntactic trees
    - discourse connectives, sentence splits
  - could potentially use other tree-like structures (see previous slide)
- Output: annotated responses
  - generate trees parallel to MRs – more guidance for the generator
    - less ambiguity, the MR shows a sentence plan as well
  - can use standard seq2seq, with linearized trees

MR

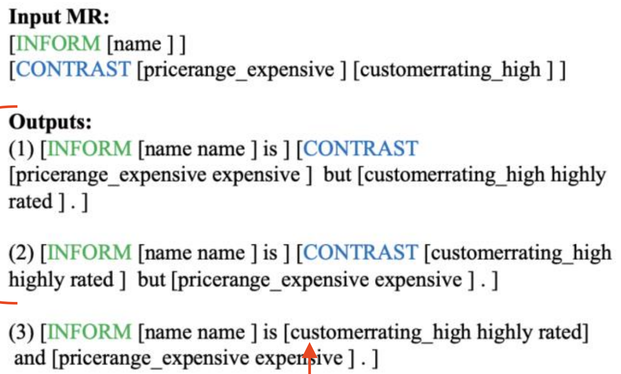
```
[CONTRAST
  [INFORM_1
    [LOCATION [CITY Parker] ] [CONDITION_NOT snow ]
    [DATE_TIME [DAY 29] [MONTH September] [YEAR 2018] ]
  ]
  [INFORM_2
    [DATE_TIME [DAY 29] [MONTH September] [YEAR 2018] ]
    [LOCATION [CITY Parker] ]
    [CONDITION heavy rain showers] [CLOUD_COVERAGE partly cloudy]
  ]
]
```

[CONTRAST [INFORM_1 [LOCATION [CITY Parker]] is not expecting any [CONDITION_NOT snow]], but [INFORM_2 [DATE_TIME [COLLOQUIAL today]] there's a [PRECIP_CHANCE_SUMMARY very likely chance] of [CONDITION heavy rain showers] and it'll be [CLOUD_COVERAGE partly cloudy]]]

Parker is not expecting any snow, but today there's a very likely chance of heavy rain showers and it'll be partly cloudy

(Balakrishnan et al., 2019) http://arxiv.org/abs/1906.07220
(Rao et al., 2019) https://www.inlg2019.com/assets/papers/10_Paper.pdf

Realizing from Trees


Input MR:
[INFORM [name]]
[CONTRAST [pricerange_expensive] [customerrating_high]]

Outputs:
(1) [INFORM [name name] is] [CONTRAST [pricerange_expensive expensive] but [customerrating_high highly rated] .]

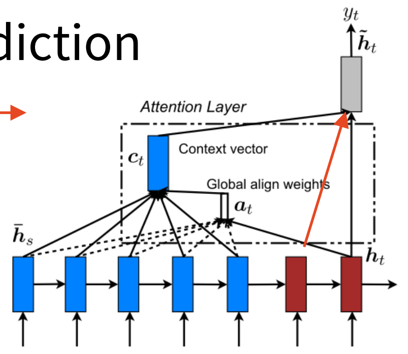
(2) [INFORM [name name] is] [CONTRAST [customerrating_high highly rated] but [pricerange_expensive expensive] .]

(3) [INFORM [name name] is [customerrating_high highly rated] and [pricerange_expensive expensive] .]

OK

this token will be disallowed

- Consistency checks – **constrained decoding**
  - when decoding, check any non-terminal against the MR
    - disallow any opening tokens not covered by MR
    - disallow any closing brackets until all children from MR are generated
- Tree-aware model
  - n-ary **TreeLSTM** encoder – copies the input MR tree structure bottom-up
    - LSTM conditioned not on just previous, but all child nodes
      - all LSTM equations sum N nodes (padded with zeros for fewer children)
  - Tree-aware decoder
    - nothing special, just use both current & previous hidden state in final prediction (Luong attention + previous hidden state)
      - previous state is often the parent tree node
- all of this improves consistency & data-efficiency



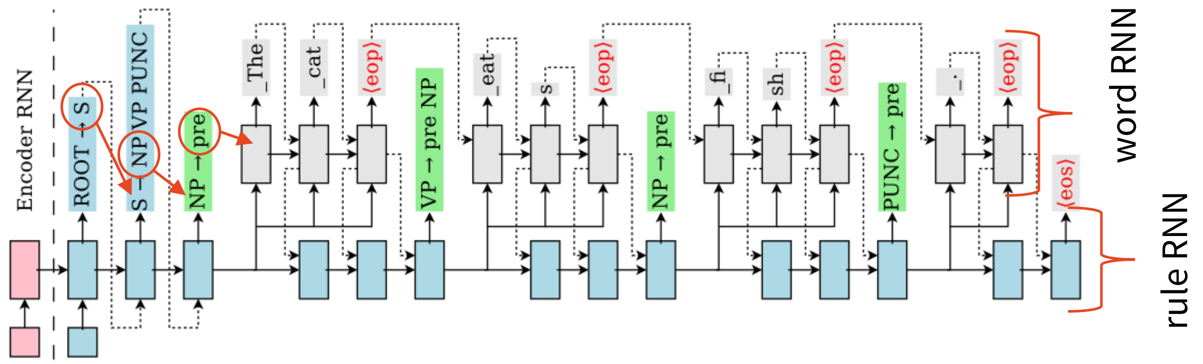
(Luong et al., 2015) http://arxiv.org/abs/1508.04025

Generating trees

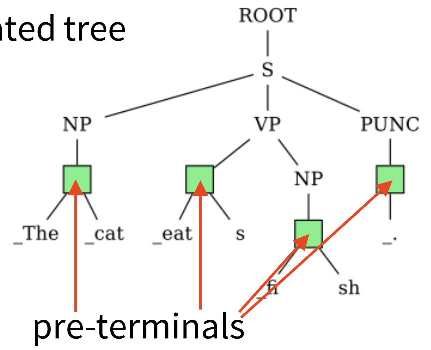
(Wang et al., 2018)
https://www.aclweb.org/anthology/D18-1509/

- Adapting seq2seq to produce real (not just linearized) trees
  - generating tree topology along with the output
- using 2 LSTM decoders:
  - **rule RNN**
    - produces CFG rules
    - applies them top-down, left-to-right (expand current non-terminal)
  - **word RNN**
    - turned on upon seeing a pre-terminal
    - generates terminal phrase word-by-word
    - ends with <eop> token, switch back to rule RNN
  - rule RNN's state is updated when word RNN generates
- can work for any type of trees
  - but found to work best for binary trees without linguistic information 😯

decoding process



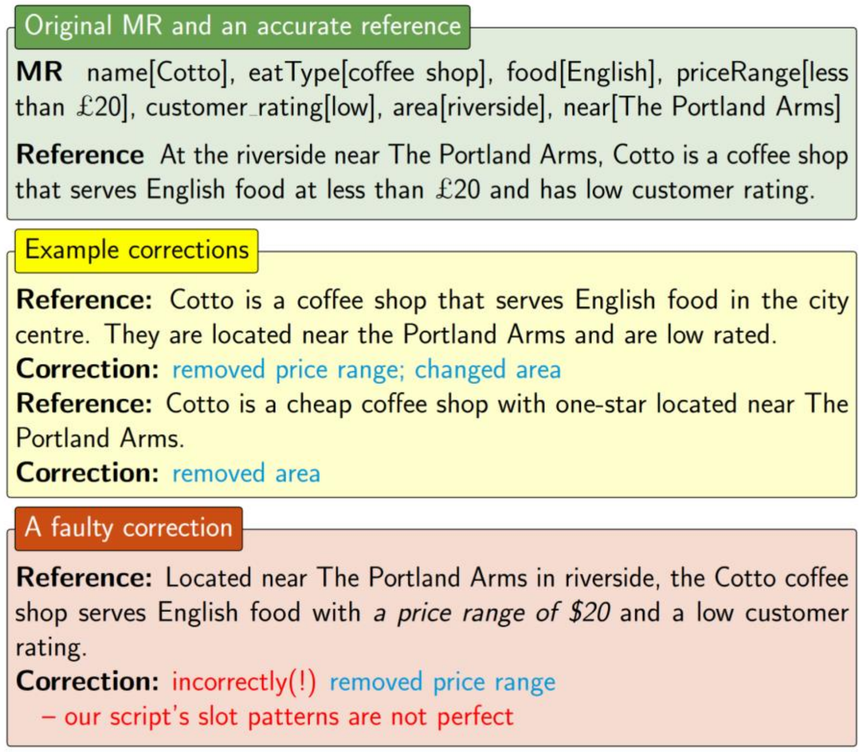
generated tree



Fact grounding

- NLG errors are often caused by **data errors**
  - ungrounded facts (← hallucinating)
  - missing facts (← forgetting)
  - domain mismatch
  - noise (e.g. source instead of target)
    - just 5% untranslated stuff kills an NMT system
- Easy-to-get data are noisy
  - web scraping – lot of noise, typically not fit for purpose
  - crowdsourcing – workers forget/don't care
- **Cleaning** improves situation a lot
  - can be done semi-automatically up to a point
  - 94-97% semantic error reduction on cleaned E2E restaurant data
  - cleaning RotoWire sports report data improves accuracy

(Khayrallah & Koehn, 2018)
https://www.aclweb.org/anthology/W18-2709



Original MR and an accurate reference

MR name[Cotto], eatType[coffee shop], food[English], priceRange[less than £20], customer_rating[low], area[riverside], near[The Portland Arms]

Reference At the riverside near The Portland Arms, Cotto is a coffee shop that serves English food at less than £20 and has low customer rating.

Example corrections

Reference: Cotto is a coffee shop that serves English food in the city centre. They are located near the Portland Arms and are low rated.
Correction: removed price range; changed area
Reference: Cotto is a cheap coffee shop with one-star located near The Portland Arms.
Correction: removed area

A faulty correction

Reference: Located near The Portland Arms in riverside, the Cotto coffee shop serves English food with *a price range of $20* and a low customer rating.
Correction: incorrectly(!) removed price range
– our script's slot patterns are not perfect

(Dušek et al., 2019)
https://arxiv.org/abs/1911.03905

(Wang et al., 2019)
https://www.inlg2019.com/assets/papers/32_Paper.pdf

Summary

Policy learning

- RL rewards are critical for good performance, can be learned
  - supervised, adversarial / dialogue- or turn-level
- RL & supervised: warm start / supervised after RL does bad (gets rarer over time)
- user simulators: good to use more & mix with humans
- multiple tasks: hierarchical / feudal RL

NLG: system DA → text

- templates work well, seq2seq & co. is the best data-driven
- problems: hallucination, not enough diversity
- attempted fixes:
  - delexicalization
  - ensembling
  - two-step: content planning & realization (simplifying the task)
  - tree-based approaches (more supervision)
  - fixing training data

Thanks

Contact us:

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

Get these slides here:

http://ufal.cz/npfl099

Labs today
14:00 SW1
Projects updates

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