

NPFL123 Dialogue Systems 7. Neural Policies & Natural Language Generation

<https://ufal.cz/npfl123>

Ondřej Dušek, Simone Balloccu, Mateusz Lango, Kristýna Klesnilová & Jan Cuřín

3. 4. 2024



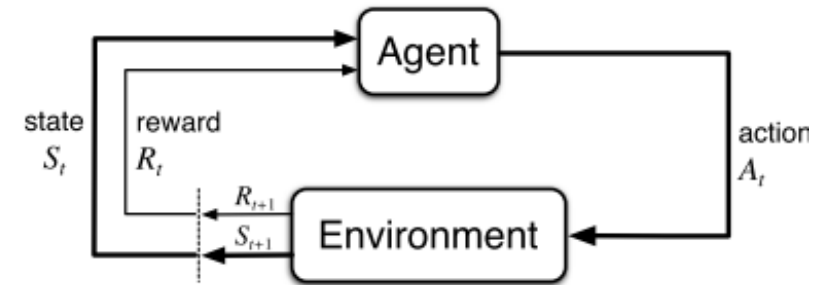
Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics



unless otherwise stated

Deep Reinforcement Learning

- Exactly the same as “plain” RL
- **“deep” = part of the agent is handled by a NN**
 - value function (typically Q)
 - policy
- NN = parametric function approximation approach
 - NN \rightarrow complex non-linear functions
 - **REINFORCE / policy gradients: $\pi(a|s, \theta)$ – works out of the box**
 - value functions: using $V(s; \theta)$ **or** $Q(s, a; \theta)$, regression
- assuming huge state space
 - much fewer weights than possible states
 - update based on one state changes many states
 - no more summary space 😊

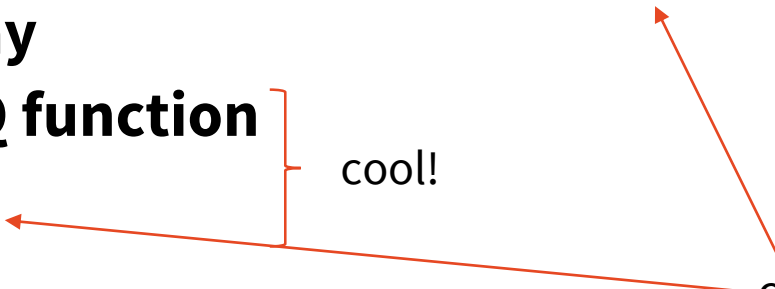


(Sutton & Barto, 2018)

- Q-learning, where Q function is represented by a neural net
- “Usual” Q-learning doesn’t converge well with NNs:
 - a) SGD is unstable
 - b) correlated samples (data is sequential)
 - c) TD updates aim at a moving target (using Q in computing updates to Q)
 - d) scale of rewards & Q values unknown \rightarrow numeric instability
- Fixes in DQN:
 - a) minibatches (updates by averaged n samples, not just one)
 - b) experience replay**
 - c) freezing target Q function**
 - d) clipping rewards

cool!

common NN tricks



DQN tricks ~ making it more like supervised learning

- **Experience replay** – break correlated samples

- run through some episodes (dialogues, games...)
- store all tuples (s, a, r', s') in a buffer
- for training, don't update based on most recent moves – use buffer
 - sample minibatches randomly from the buffer
- overwrite buffer as you go, clear buffer once in a while
- only possible for off-policy

← “generate your own
‘supervised’ training data”

$$\text{loss} := \mathbb{E}_{(s,a,r',s') \in \text{buf}} \left[\left(r' + \gamma \max_{a'} Q(s', a'; \bar{\theta}) - Q(s, a; \theta) \right)^2 \right]$$

- **Target Q function freezing**

- fix the version of Q function used in update targets
 - have a copy of your Q network that doesn't get updated every time
- once in a while, copy your current estimate over

← “have a fixed target,
like in supervised learning”

DQN algorithm

- initialize θ randomly
- initialize replay memory D (e.g. play for a while using current $Q(\theta)$)
- repeat over all episodes:
 - set initial state s
 - for all timesteps $t = 1 \dots T$ in the episode:
 - select action a_t from ϵ -greedy policy based on $Q(\theta)$
 - take a_t , observe reward r_{t+1} and new state s_{t+1}
 - store $(s_t, a_t, r_{t+1}, s_{t+1})$ in D

} storing experience
(1 step of Q-learning exploration)
 - sample a batch B of random (s, a, r', s') 's from D
 - update θ using loss $\mathbb{E}_{(s,a,r',s') \in B} \left[\left(r' + \gamma \max_{a'} Q(s', a'; \bar{\theta}) - Q(s, a; \theta) \right)^2 \right]$

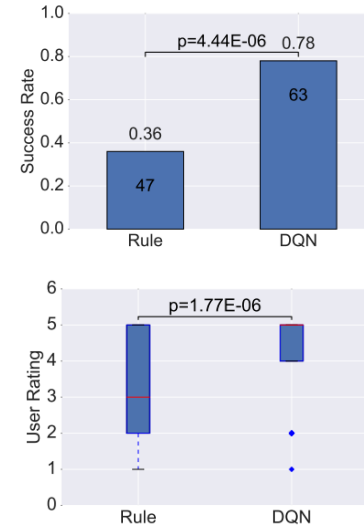
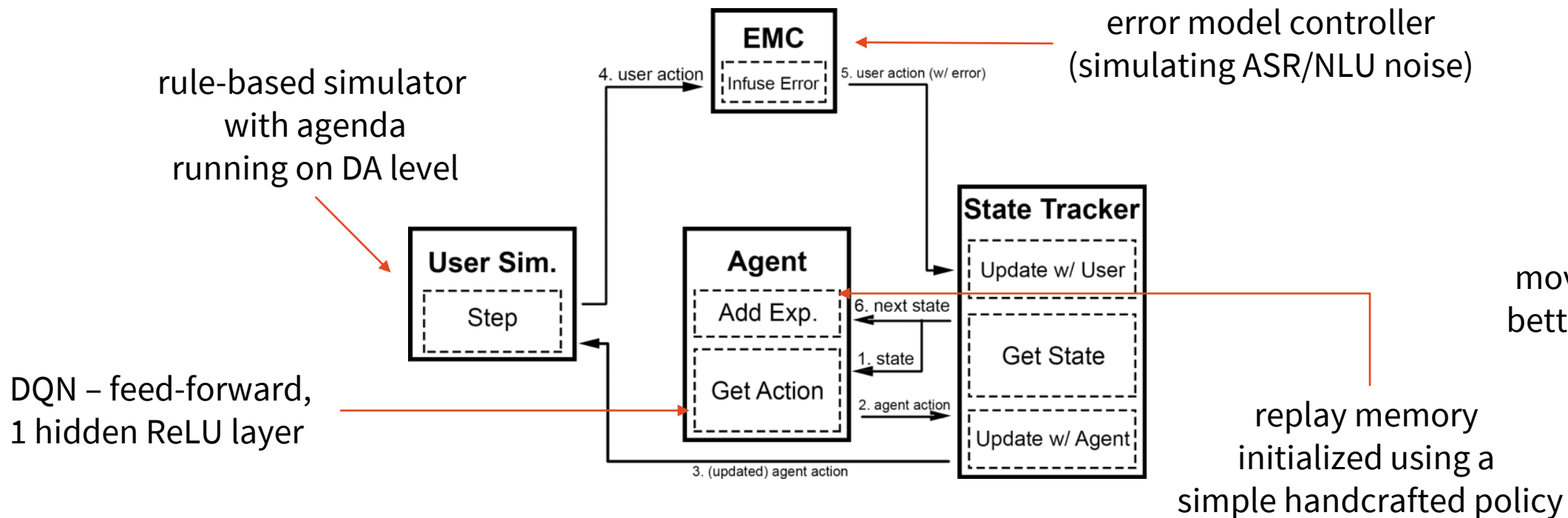
} “replay”
a. k. a. training
(1 update)
- once every λ steps (rarely):
 - $\bar{\theta} \leftarrow \theta$

← update the frozen target function

DQN for Dialogue Systems

(Li et al., 2017)
<https://arxiv.org/abs/1703.01008>
<https://github.com/MiuLab/TC-Bot>

- a simple DQN can drive a dialogue system's action selection
 - DQN is function approximation – works fine for POMDPs
 - no summary space tricks needed here



movie ticket booking:
better than rule-based

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 ← e.g. “user wants short answers”
 - dialogue history ← e.g. for referring expressions, avoiding repetition
- can be any kind of knowledge representation
- general NLG objective:
 - **given input & communication goal**
 - **create accurate + natural, well-formed, human-like text**
 - additional NLG desired properties:
 - variation
 - simplicity
 - adaptability

NLG Use Cases

- **dialogue systems**
 - very different for task/non-task-oriented/QA systems
- **standalone**
 - data-to-text
 - short text generation for web & apps
 - weather, sports reports
 - personalized letters
 - creative generation (stories)
- **machine translation**
 - now mostly integrated end-to-end
 - formerly not the case
- **summarization**

NLG Subtasks (textbook pipeline)

Inputs

- **↓ Content/text/document planning**

- content selection according to communication goal
- basic structuring & ordering

typically handled by
dialogue manager
in dialogue systems

Content plan

- **↓ Sentence planning/microplanning**

- aggregation (facts → sentences)
- lexical choice
- referring expressions

organizing content into sentences
& merging simple sentences

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

Sentence plan

- **↓ Surface realization**

- linearization according to grammar
- word order, morphology

this is needed for NLG
in dialogue systems

Text

deciding
what to say

deciding
how to say it

NLG Implementations

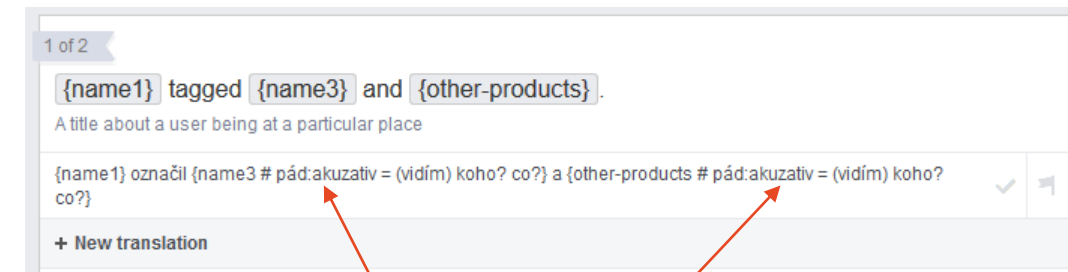
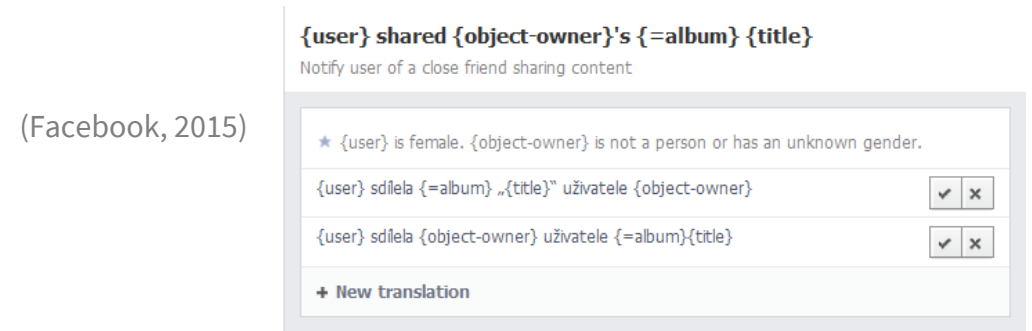
- **Few systems implement the whole pipeline**
 - All stages: mostly domain-specific data-to-text, standalone
 - e.g. weather reports
 - Dialogue systems: just sentence planning + realization
 - Systems focused on content + sentence planning with trivial realization
 - frequent in DS: focus on sentence planning, trivial or off-the-shelf realizer
 - Surface realization only
 - requires very detailed input
 - some systems: just ordering words
- **Pipeline vs. end-to-end approaches**
 - planning + realization in one go – popular for neural approaches
 - pipeline: simpler components, might be reusable (especially realizers)
 - end-to-end: no error accumulation, no intermediate data structures

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
 - neural nets 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, 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},"
```

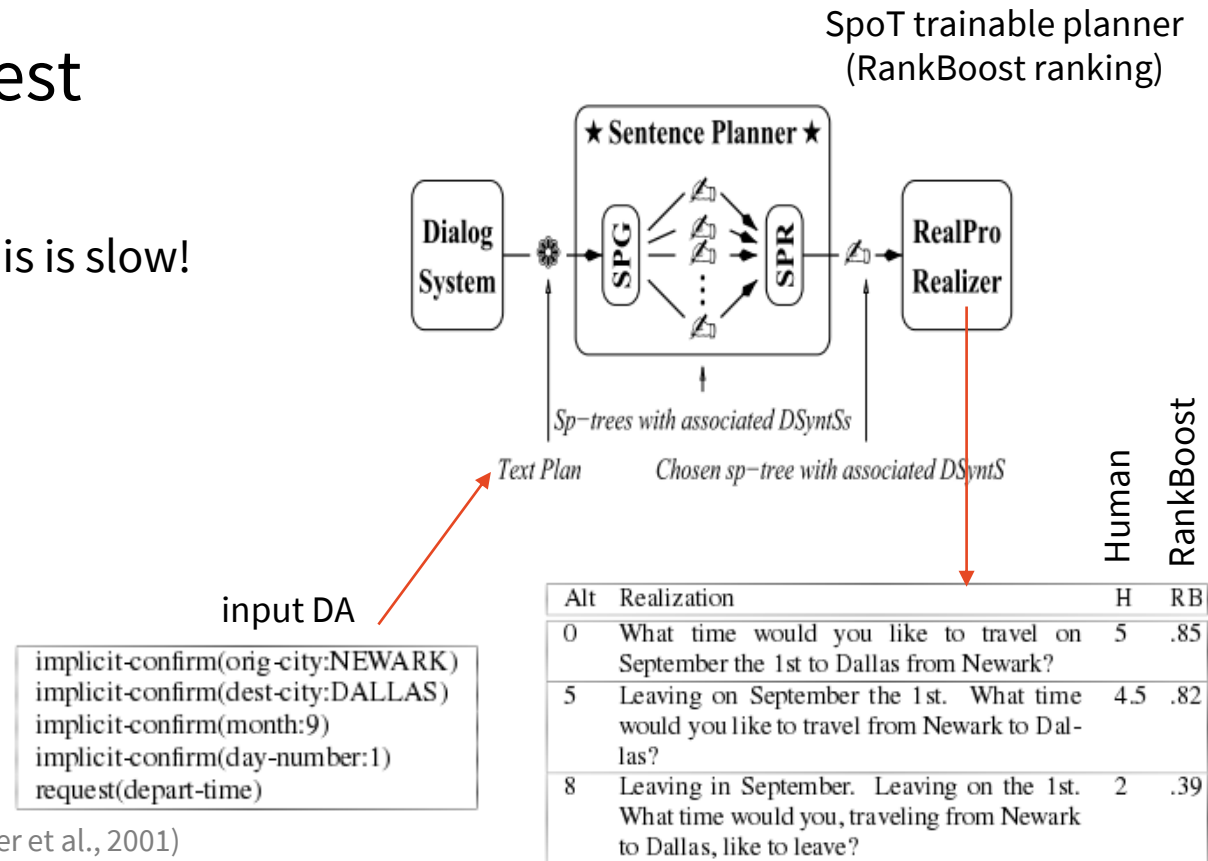
Grammar/Rules for Sentence Planning

- Handcrafted grammar/rules
 - input: base semantics (e.g. dialogue acts)
 - output: detailed sentence representation (=realizer inputs, see →)

- Statistical enhancements:
generate more options & choose the best

- generate multiple outputs
 - underspecified grammar
 - rules with multiple options...
- choose the best one
 - train just the selection – learning to rank
 - any supervised approach possible
e.g. “best” = 1, “not best” = 0

NB: this is slow!



input DA

```
implicit-confirm(orig-city:NEWARK)
implicit-confirm(dest-city:DALLAS)
implicit-confirm(month:9)
implicit-confirm(day-number:1)
request(depart-time)
```

(Walker et al., 2001)
<https://www.aclweb.org/anthology/N01-1003>

Grammar-based realizers

- Various grammar formalisms
 - production / unification rules in the grammar
 - lexicons to go with it
 - expect very detailed input (*sentence plans*)
- typically general-domain, reusable
 - **KPML** – multilingual
 - systemic functional grammar
 - **FUF/SURGE** – English
 - functional unification grammar
 - **OpenCCG** – English
 - combinatory categorial grammar

KPML input for *A dog is in the park.*

```
(l0 / spatial-locating
:speechact (a0 / assertion :polarity positive
            :speaking-time t0)
:reference-time-id t0
:event-time (t0 / time)
:theme d0
:domain (d0 / object :lex dog
         :identifiability-q notidentifiable)
:range (p0 / three-d-location :lex park
        :identifiability-q identifiable))
```

FUF/SURGE input for *She hands the draft to the editor*

$$\left[\begin{array}{l} \text{cat} \\ \text{process} \\ \text{partic} \end{array} \begin{array}{l} \text{clause} \\ \left[\begin{array}{l} \text{type} \\ \text{relation} \\ \text{lex} \end{array} \right. \\ \left. \begin{array}{l} \text{composite} \\ \text{possessive} \\ \text{"hand"} \end{array} \right] \\ \left[\begin{array}{l} \text{agent} \\ \text{affected} \\ \text{possessor} \\ \text{possessed} \end{array} \right. \\ \left. \left[\begin{array}{l} \left[\begin{array}{l} \text{cat} \\ \text{gender} \end{array} \right. \right. \\ \left. \left[\begin{array}{l} \text{pers_pro} \\ \text{feminine} \end{array} \right] \right] \\ \left[\begin{array}{l} \boxed{1} \\ \left[\begin{array}{l} \text{cat} \\ \text{lex} \end{array} \right. \right. \\ \left. \left[\begin{array}{l} \text{np} \\ \text{"editor"} \end{array} \right] \right] \\ \boxed{1} \\ \left[\begin{array}{l} \text{cat} \\ \text{lex} \end{array} \right. \right. \\ \left. \left[\begin{array}{l} \text{np} \\ \text{"draft"} \end{array} \right] \right] \end{array} \right] \end{array} \right]$$

OpenCCG input for *The cheapest flight is on Ryanair*

```
be [tense=pres info=rh id=n1]
<Arg> flight [num=sg det=the info=th id=f2]
      <HasProp> cheapest [kon+= id=n2]
<Prop> has-rel [id=n3]
      <Of> f2
      <Airline> Ryanair [kon+= id=n4]
```

Procedural realizers

SimpleNLG

- **SimpleNLG** – no grammar, code to build sentence
 - “do-it-yourself” style – only cares about the grammar
 - system then linearizes
 - built for English, ports to other languages available
- **RealPro** (Meaning-Text-Theory)
 - deep syntax/semantics → surface syntax → morphology
- **Treex** (Functional Generative Description)
 - deep syntax → surface syntax → morphology, linearization
 - Perl code operating over syntax trees

```
Lexicon lexicon = new XMLLexicon("my-lexicon.xml");
NLGFactory nlgFactory = new NLGFactory(lexicon);
Realiser realiser = new Realiser(lexicon);

SPhraseSpec p = nlgFactory.createClause();

p.setSubject("Mary");
p.setVerb("chase");
p.setObject("the monkey");

p.setFeature(Feature.TENSE, Tense.PAST);

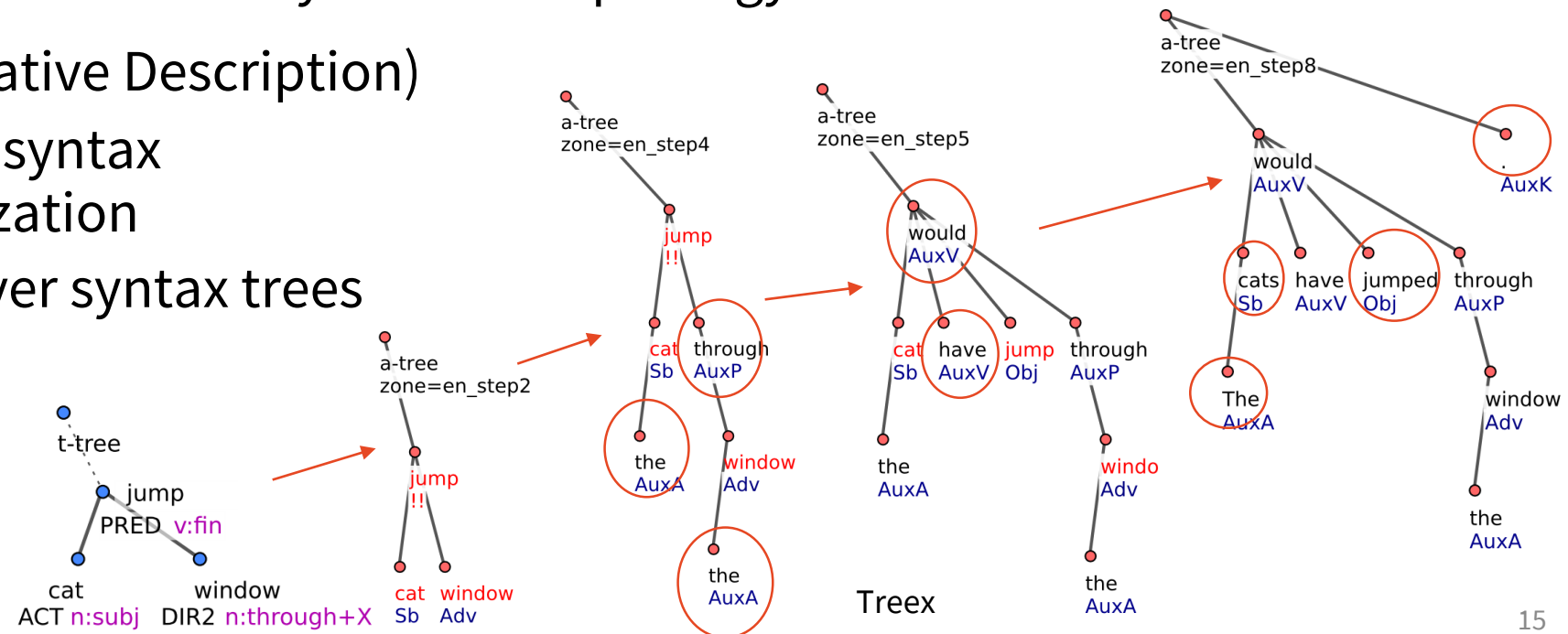
String output = realiser.realiseSentence(p);
System.out.println(output);

>>> Mary chased the monkey.
```

(Gatt & Reiter, 2009)
<https://www.aclweb.org/anthology/W09-0613>

(Lavoie & Rambow, 1997)
<http://dl.acm.org/citation.cfm?id=974596>

(Popel & Žabokrtský 2010; Dušek et al., 2015)
https://ufal.mff.cuni.cz/~popel/papers/2010_icetal.pdf
<https://www.aclweb.org/anthology/W15-3009>



Trainable Realizers

• Overgenerate & Rerank

- same approach as for sentence planning
- assuming a flexible handcrafted realizer (e.g., OpenCCG)
- underspecified input → more outputs possible
- generate more & use statistical reranker, based on:

this means
the grammar
may be smaller

- n-gram language models NITROGEN (Langkilde & Knight, 1998) <https://www.aclweb.org/anthology/P98-1116>
HALOGEN (Langkilde-Geary, 2002) <https://www.aclweb.org/anthology/W02-2103>
- Tree language models FERGUS (Bangalore & Rambow, 2000) <https://aclweb.org/anthology/C00-1007>
- expected text-to-speech output quality (Nakatsu & White, 2006) <https://www.aclweb.org/anthology/P06-1140>
- personality traits & alignment/entrainment CRAG (Isard et al., 2006) <https://www.aclweb.org/anthology/W06-1405>
- more variance, but at computational cost

• Grammar/Procedural-based

- same as RealPro or TectoMT, but predict each step using a classifier

StuMaBa (Bohnet et al., 2010)
<https://www.aclweb.org/anthology/C10-1012>

Non-Neural End-to-End NLG

- NLG as language models

- hierarchy of language models (HMM/MEMM/CRF style)
- DA → slot → word level

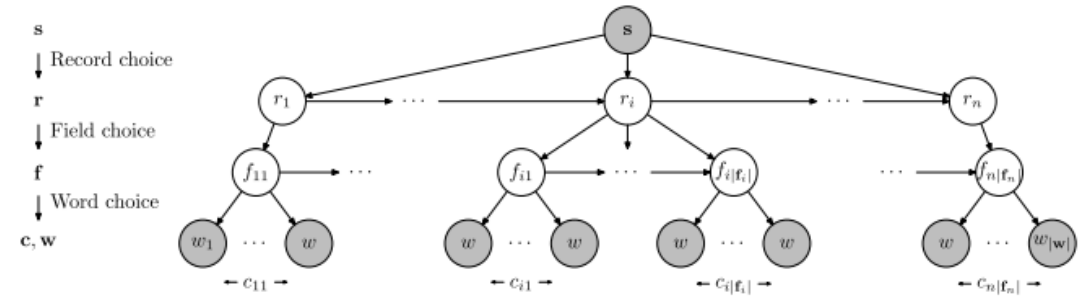
- NLG using context-free grammars

- a) “language models” by probabilistic CFGs

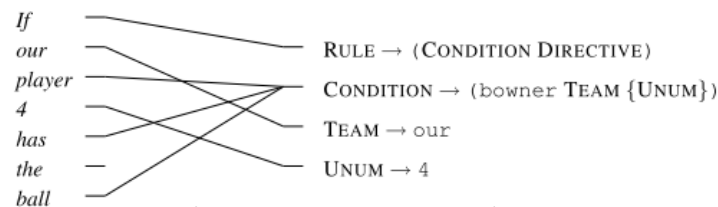
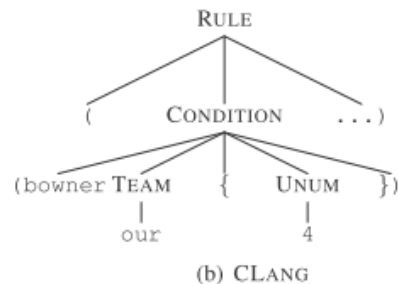
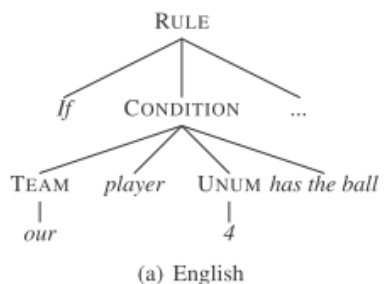
- approximate search for best CFG derivation

- b) synchronous PCFGs – MRs & text

- “translation” with hierarchical phrase-based system
- parsing MR & generating text



(Oh & Rudnicky, 2002) [https://doi.org/10.1016/S0885-2308\(02\)00012-8](https://doi.org/10.1016/S0885-2308(02)00012-8)
 (Angeli et al., 2010) <https://www.aclweb.org/anthology/D10-1049>
 (Liang et al., 2009) <https://www.aclweb.org/anthology/P09-1011>
 (Mairesse et al., 2010) <https://www.aclweb.org/anthology/P10-1157>
 (Mairesse & Young, 2014) <https://www.aclweb.org/anthology/J14-4003>

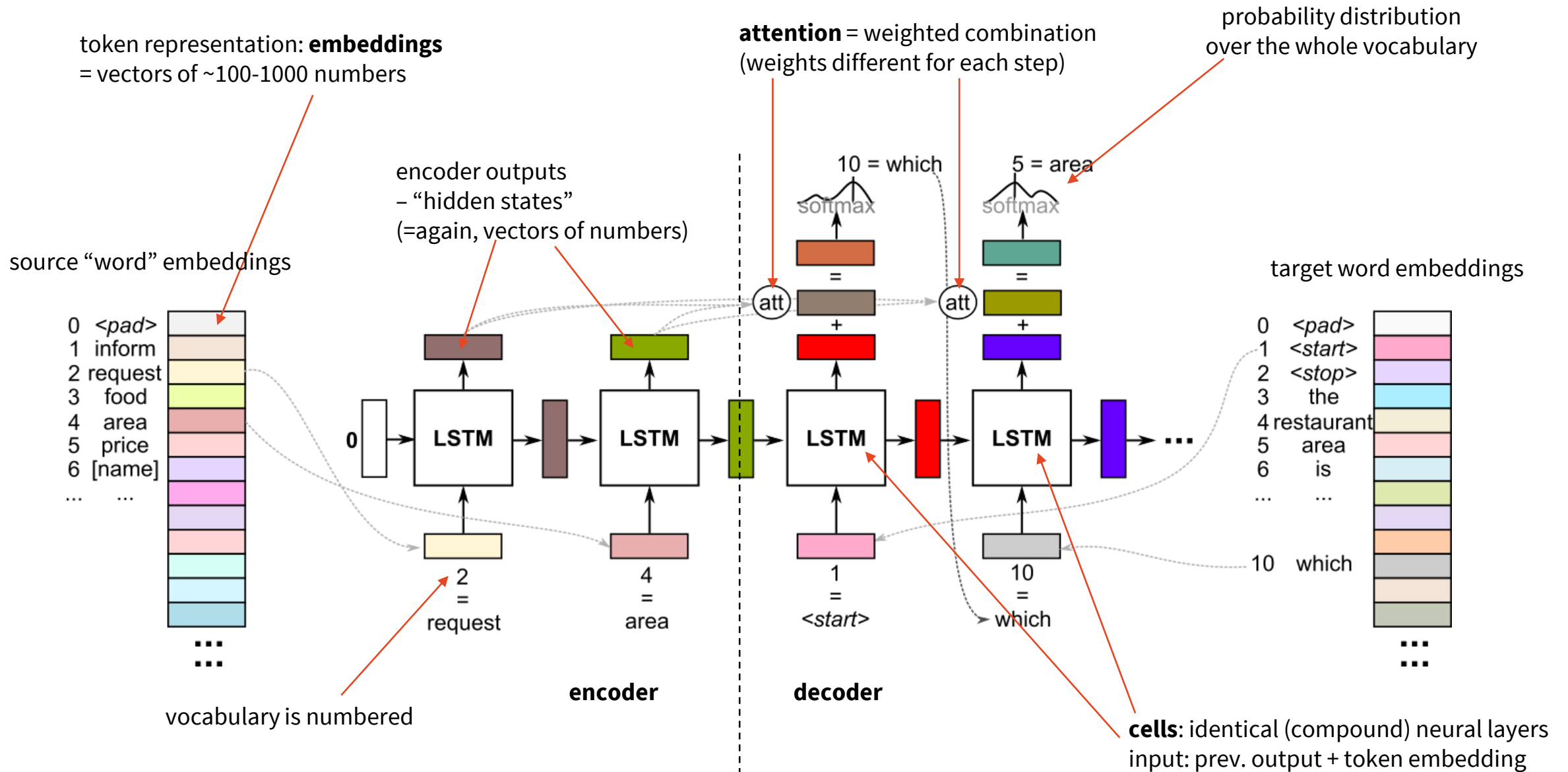


(Wong & Mooney, 2007) <https://www.aclweb.org/anthology/N07-1022>

rule	prob./parameter
1. $S \rightarrow R(start)$	$[Pr = 1]$
2. $R(r,t) \rightarrow FS(r_j, start) R(r_j, t)$	$[P(r_j, t r, t) \cdot \lambda]$
3. $R(r,t) \rightarrow FS(r_j, start)$	$[P(r_j, t r, t) \cdot \lambda]$
4. $FS(r, r, f_i) \rightarrow F(r, r, f_j) FS(r, r, f_j)$	$[P(f_j f_i)]$
5. $FS(r, r, f_i) \rightarrow F(r, r, f_j)$	$[P(f_j f_i)]$
6. $F(r, r, f) \rightarrow W(r, r, f) F(r, r, f)$	$[P(w w_{-1}, r, r, f)]$
7. $F(r, r, f) \rightarrow W(r, r, f)$	$[P(w w_{-1}, r, r, f)]$
9. $W(r, r, f) \rightarrow g(f, v)$	$[P(\alpha r, r, f, f, t = int)]$

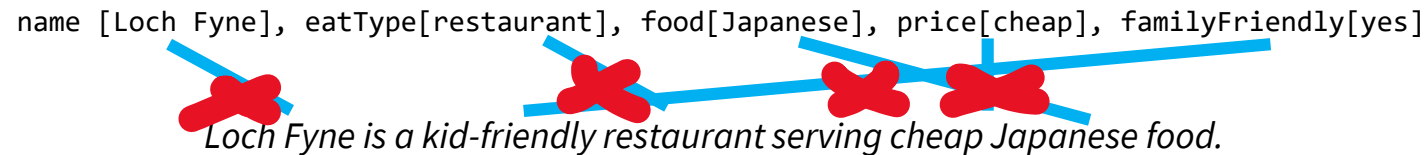
(Konstas & Lapata, 2012) <https://www.aclweb.org/anthology/P12-1039>

Neural Generation: Seq2seq RNNs (see NLU for RNN intro)

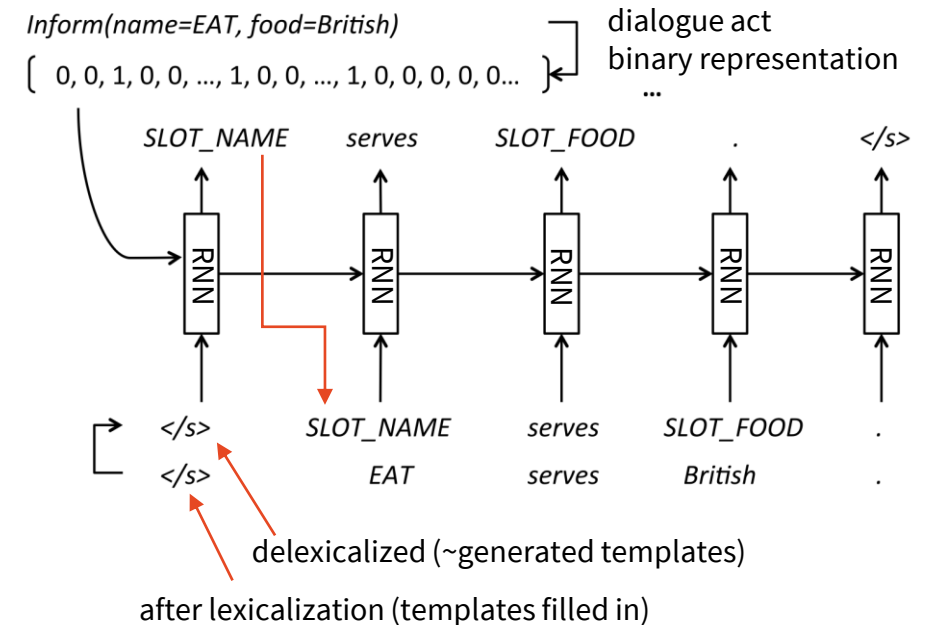


Neural End-to-End NLG: RNNs

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



- 1st system: RNN language model conditioned on DA (~decoder only)
 - input: binary-encoded DA
 - 1 if intent/slot-value present, 0 if not
 - delexicalized: much fewer values, shorter vector
 - modified LSTM cells
 - input DA passed in every time step
 - generating delexicalized texts word-by-word
 - i.e. decoder only



Seq2seq NLG with reranking (TGen)

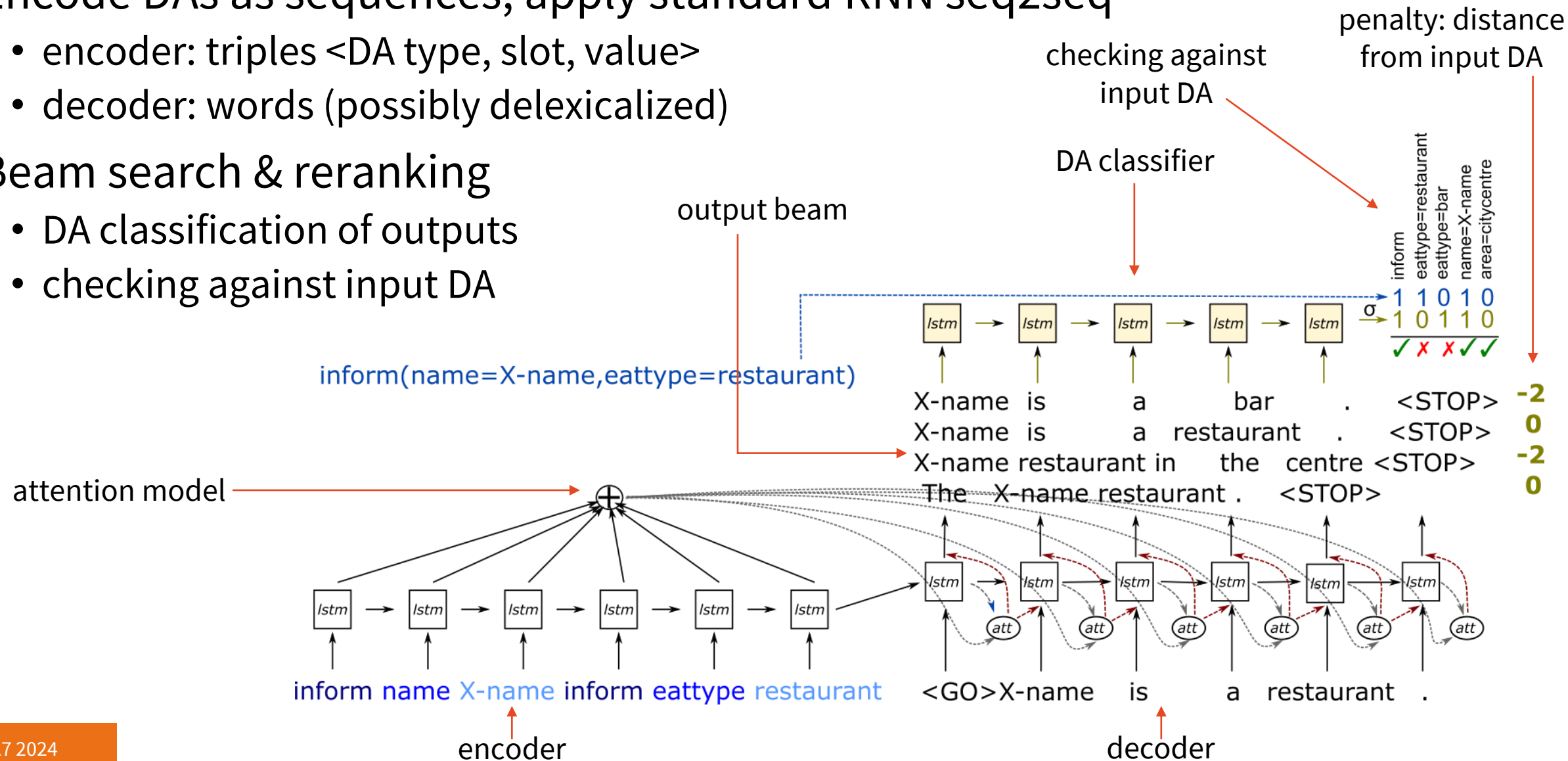
(Dušek & Jurčiček, 2016)
<https://aclweb.org/anthology/P16-2008>

- Encode DAs as sequences, apply standard RNN seq2seq

- encoder: triples <DA type, slot, value>
- decoder: words (possibly delexicalized)

- Beam search & reranking

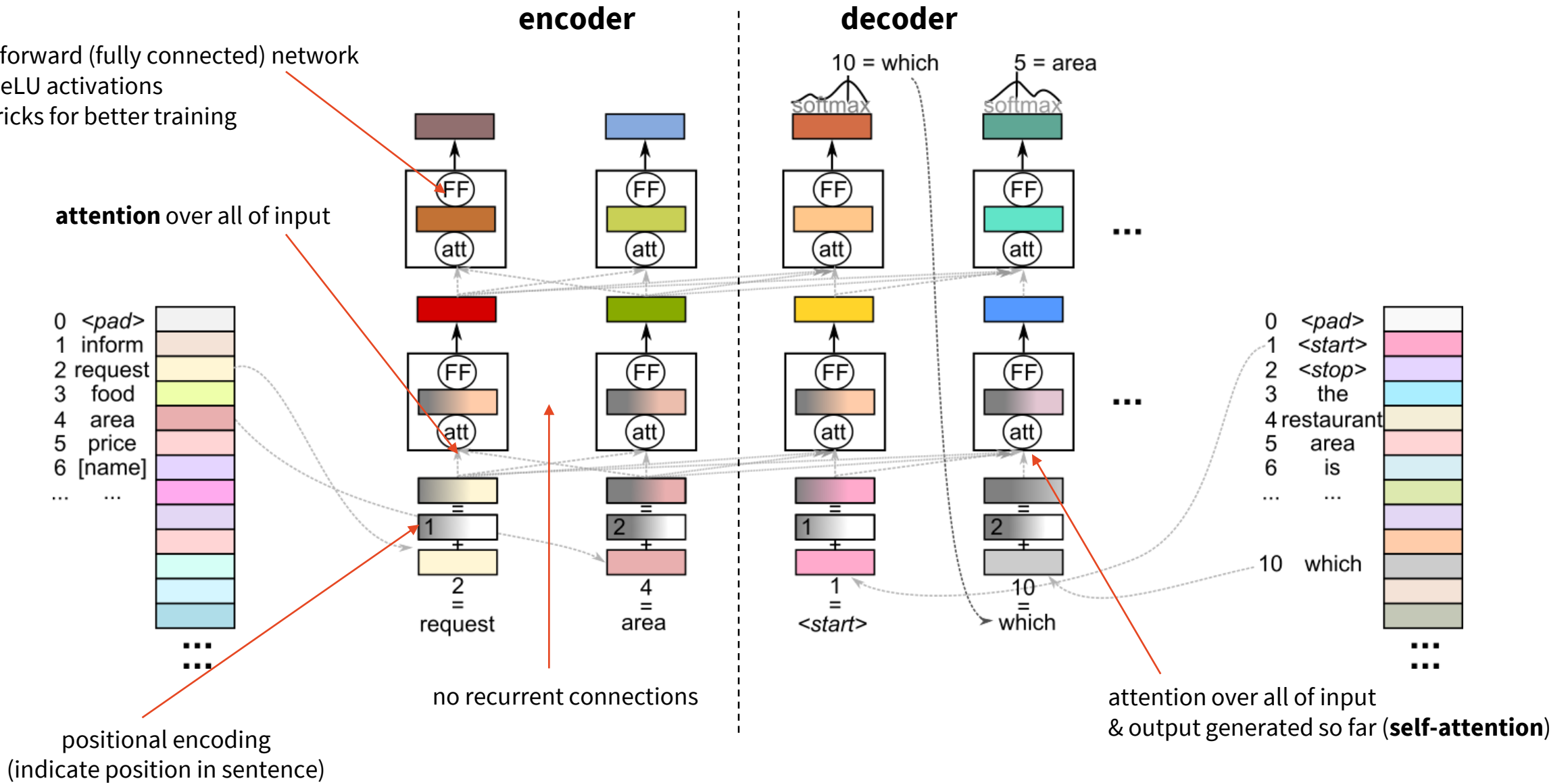
- DA classification of outputs
- checking against input DA



Transformer = seq2seq, with feed-forward & attention nets (instead of RNN)

feed-forward (fully connected) network

- ReLU activations
- tricks for better training



Transformers & Pretrained Language Models

- **Transformer architecture** (Vaswani et al., 2017) <http://arxiv.org/abs/1706.03762>
 - encoder-decoder, but using feed-forward & attention instead of RNNs
 - positional encoding used to indicate sentence position
 - predefined “pattern” functions (based on sin & cos)
 - simply added to word embeddings
 - no RNN → parallel training → faster, allows larger models (more layers)
- **Pretrained language models** – on large data w/o annotation (**self-supervised**)
 - guess masked word (encoder only: BERT) (Devlin et al., 2019) <https://www.aclweb.org/anthology/N19-1423>
 - generate next word (decoder only: GPTx) (Radford et al., 2019) <https://openai.com/blog/better-language-models/>
 - fix distorted sentences (both: BART, T5) (Lewis et al., 2020) <https://www.aclweb.org/anthology/2020.acl-main.703>
(Raffel et al., 2020) <http://jmlr.org/papers/v21/20-074.html>
- Can be **finetuned** for your domain & task (Chen et al., 2020) <https://www.aclweb.org/anthology/2020.acl-main.18/>
(Kasner & Dušek, 2020) <https://www.aclweb.org/anthology/2020.webnlg-1.20/>
 - less data than w/o pretraining, extremely fluent
- Large LMs (GPT3+): **prompting**, no finetuning needed (Brown et al., 2020) <http://arxiv.org/abs/2005.14165>
(Wang et al., 2022) <http://arxiv.org/abs/2204.07705>

- Checking the **semantics**
 - neural models tend to forget input / make up irrelevant stuff
 - reranking / decoding changes work, but not perfectly
 - generally **hard to control** (especially LLMs)
- Needs quite a lot of data (except LLMs, with prompting)
- Delexicalization needed (at least some slots) ← open sets, verbatim on the output (e.g., restaurant/area names)
 - typically OK for pretrained LMs
- Diversity & complexity of outputs
 - still can't match humans
 - needs specific tricks to improve this
- Still more hassle than writing up templates 😊

Deep Reinforcement Learning

- same as plain RL – agent + states, actions, rewards – just Q or π is a NN
- function approximation for Q – mean squared value error
- **Deep Q Networks** – Q learning where Q is a NN + tricks
 - experience replay, target function freezing
- **Policy networks** – policy gradients where π is a NN

Natural Language Generation

- steps: content planning, **sentence planning**, **surface realization**
 - not all systems implement everything (content planning is DM's job in DS)
 - pipeline vs. end-to-end
- approaches: templates, grammars, statistical
- **templates** work great
- neural: **RNN / Transformer**, pretrained models

Contact us:

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

Labs in 10 minutes

Get these slides here:

<http://ufal.cz/npfl123>

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/>
- Deep RL for NLP tutorial: <https://sites.cs.ucsb.edu/~william/papers/ACL2018DRL4NLP.pdf>
- Mnih et al. (2013): Playing Atari with Deep Reinforcement Learning: <https://arxiv.org/abs/1312.5602>
- Mnih et al. (2015): Human-level control through deep reinforcement learning: <https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf>
- Gatt & Kraemer (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>