

# NPFL099 Statistical Dialogue Systems

## 8. Natural Language Generation (2)

<http://ufal.cz/npfl099>

**Ondřej Dušek** & Vojtěch Hudeček

24. 11. 2020



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



unless otherwise stated

# Recap from last time: NLG

- system action/DA → text
  - other NLG applications – from DB tables, raw data etc.
  - user model, dialogue history can be considered
  - goal: accurate, natural, human-like, varied
- traditional pipeline:
  - 1) content planning / content selection – selection, pre-ordering (not so much in DSs)
  - 2) sentence planning – aggregation, lexical choice, referring expression
  - 3) surface realization – word order, morphology
  - these steps are often joined in one model
- **templates** – most used in industry
- **neural – seq2seq** + copy/delexicalization + reranking
  - problems: hallucination, not enough diversity

# Data Noise & Cleaning

- 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

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

- 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

(Wang, 2019)  
<https://www.aclweb.org/anthology/W19-8639/>

## 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>

# Data Augmentation

- 1) Get more texts that look like your outputs
    - get texts online that come from the target domain
  - 2) Produce corresponding inputs
    - automatically, noisily
    - need a parser/NLU system for that
  - 3) Mix the result with your training data
    - potentially pretrain on synthetic data, then finetune on real data
- Increases diversity of data, robustness of models
  - Relatively easy to do for broad-coverage surface realizers
    - harder for everything else: where to get the right data?

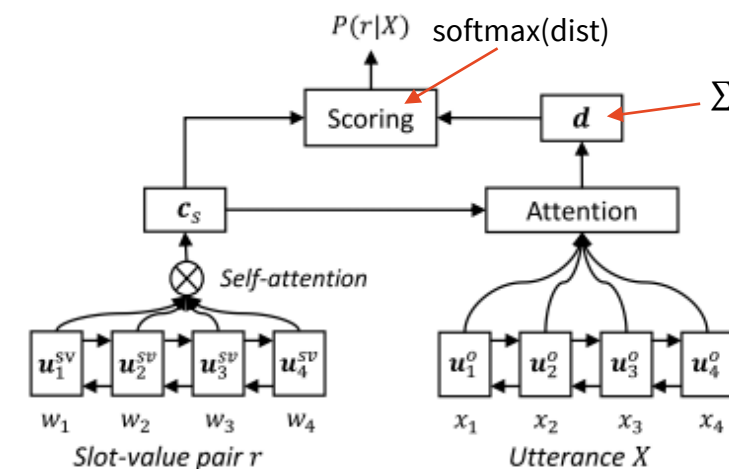
(Elder et al., 2020)

<https://www.aclweb.org/anthology/2020.acl-main.665>

- 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  (25k for each # of slots)
    - sample many DAs at random 
    - **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 
  - 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

# NLG-NLU Combo: NLU data cleaning

- NLU used to clean training data
  - 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



	BLEU(%)	Err(%)
original data		
TGen	65.90	18.09 (114/630)
Slug2Slug	66.19	6.51 (41/630)
plain supervised NLU		
Seq2Seq	66.15	69.37 (374/630)
Seq2Seq+aug	<b>66.49</b>	28.89 (182/630)
iterative NLU training		
Seq2Seq+aug+iter	65.63	2.07 (13/630)
Seq2Seq+aligner	63.81	<b>1.75</b> (11/630)

handcrafted NLU

(Nie et al., 2019)

<https://www.aclweb.org/anthology/P19-1256>

- multi-objective optimization

- basically normal training with regularization for duality:

$$P(x, y) = P(x)P(y|x, \theta_{x \rightarrow y}) = P(y)P(x|y, \theta_{y \rightarrow x})$$

- attempting to model the whole distribution  $P(x, y)$ , should work both ways (via both NLG and NLU)

- regularization term:  $(\log P(x) + \log P(y|x, \theta_{x \rightarrow y}) - \log P(y) - \log P(x|y, \theta_{y \rightarrow x}))^2$

- if duality holds, this is 0



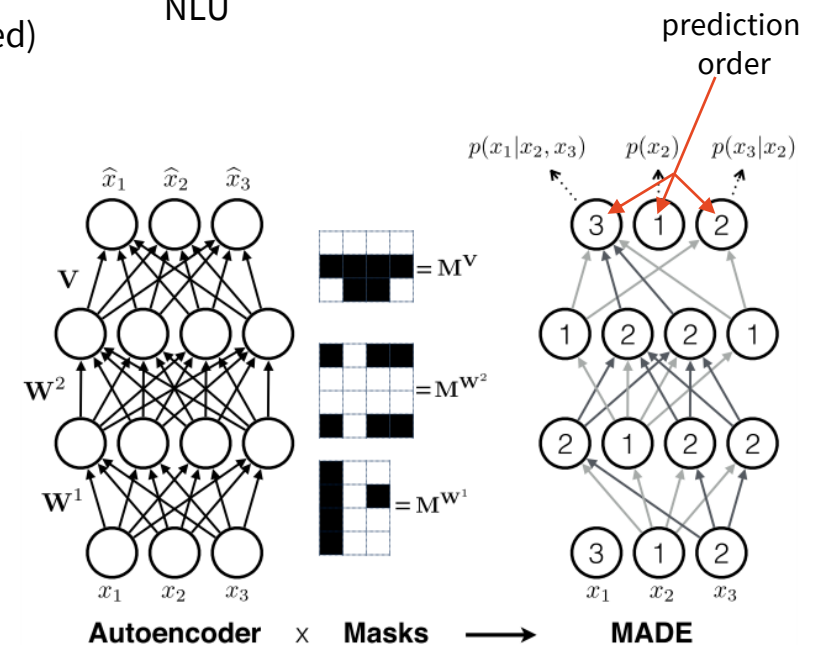
- added to both NLG and NLU training, with given weight

- NLG & NLU = seq2seq (GRU)

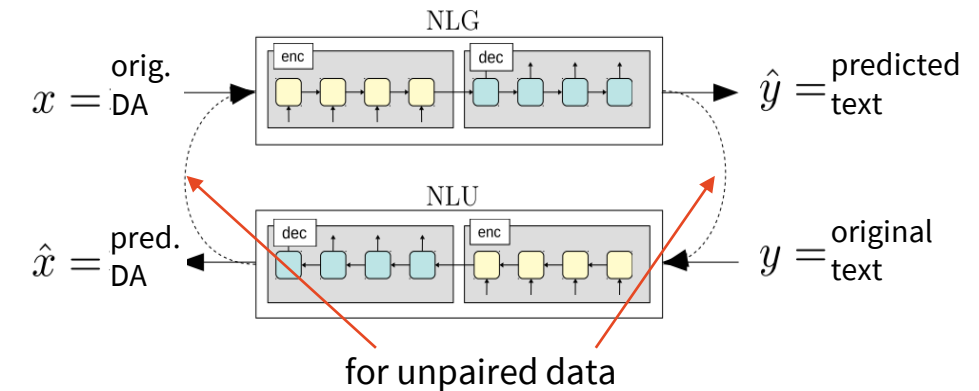
- $P(y)$  = RNN language model

- $P(x)$  = masked autoencoder

- create dependencies among slots
- join multiple possible dependency orders



- 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 categorical token distributions
      - straight-through = real (hard) sampling for forward pass, smooth approximation for backward pass





- “reparameterization trick for discrete distributions”

- reparameterization:  $z \sim \mathcal{N}(\mu, \sigma) \rightarrow z \sim \mu + \sigma \cdot \mathcal{N}(0,1)$ 
  - differentiating w. r. t.  $\mu$  &  $\sigma$  still works, no hard sampling on that path

Normal noise

- Gumbel-max:

- categorical distribution  $\pi$  with probabilities  $\pi_i$
- sampling from  $\pi$ :  $z = \text{onehot}(\arg \max_i (\log \pi_i + g_i))$

Gumbel noise:

$$g_i = -\log(-\log(\text{Uniform}(0,1)))$$

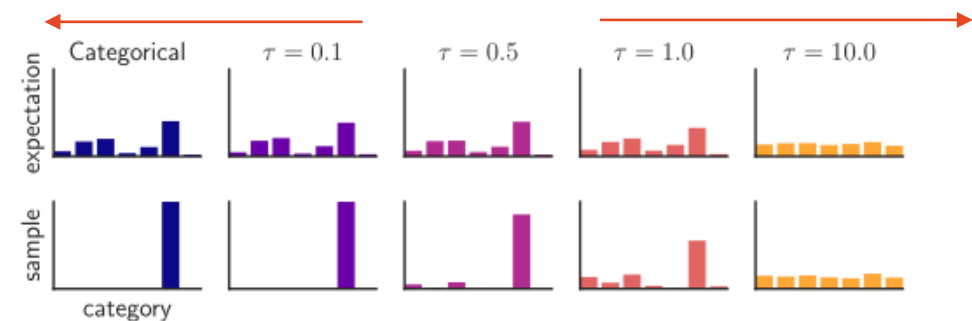
- Swap argmax for softmax with temperature  $\tau$ :

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

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

$\tau \rightarrow 0$ : more like one-hot

$\tau \rightarrow \infty$ : more like uniform



# Few-shot NLG with Pretrained LMs

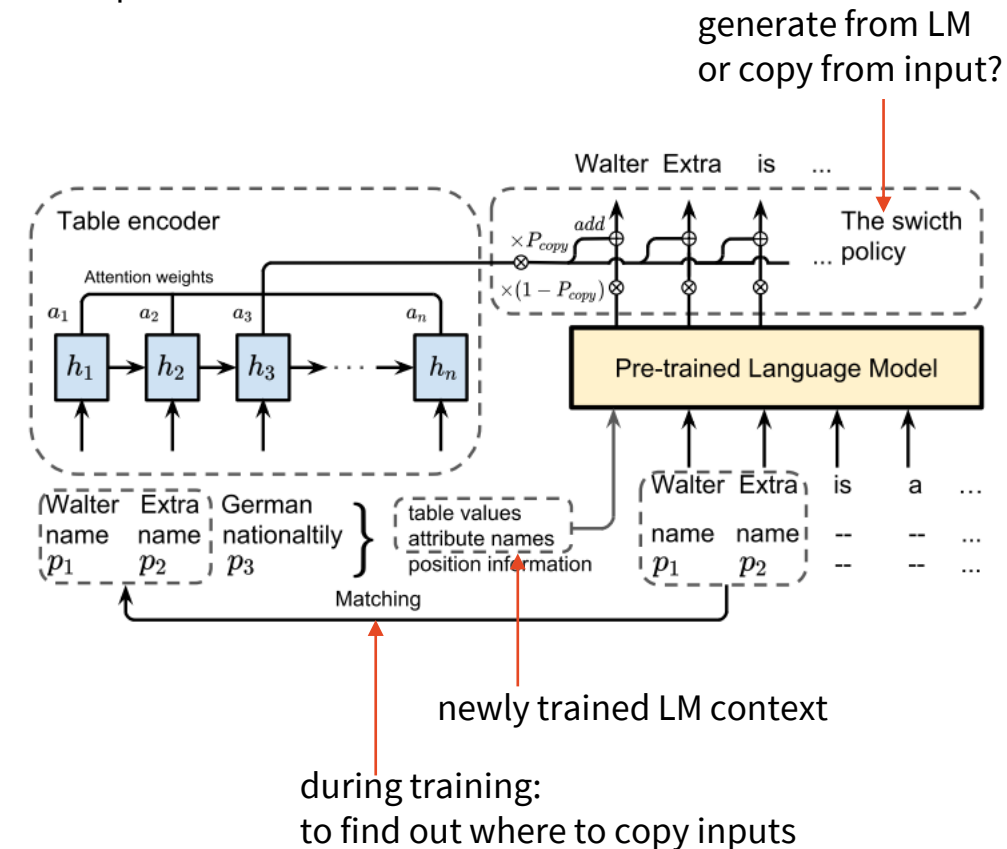
(Chen et al., 2020)

<https://www.aclweb.org/anthology/2020.acl-main.18/>

- 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

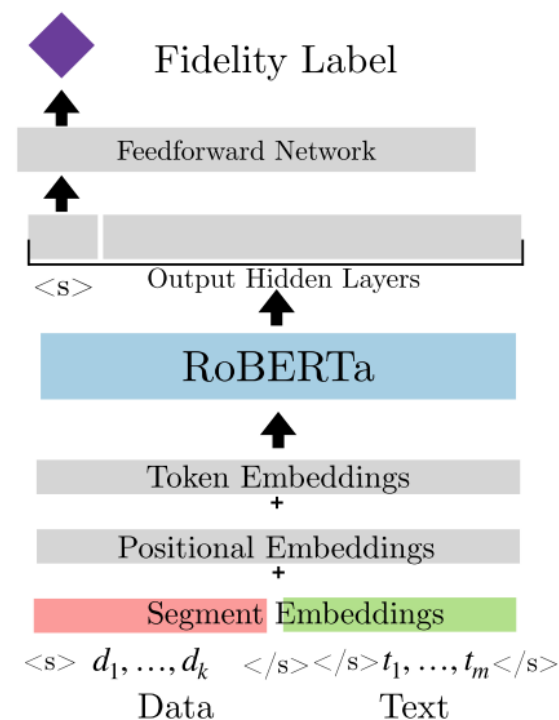


- Simpler than previous
  - Basically the same as seq2seq + reranking
  - just with GPT-2 & RoBERTa instead of LSTMs
- GPT-2 fine-tuned for `<data> name[Zizzi] eatType[bar] <text>`
  - on the target datasets
- beam search decoding
- RoBERTa for classification
  - accurate/omission/repetition/hallucination/value error
  - training data synthesized
    - “accurate” examples from original training data
    - others created by manipulating the data and texts (adding/removing/replacing sentences and/or data items)

prompt (fed into GPT-2)

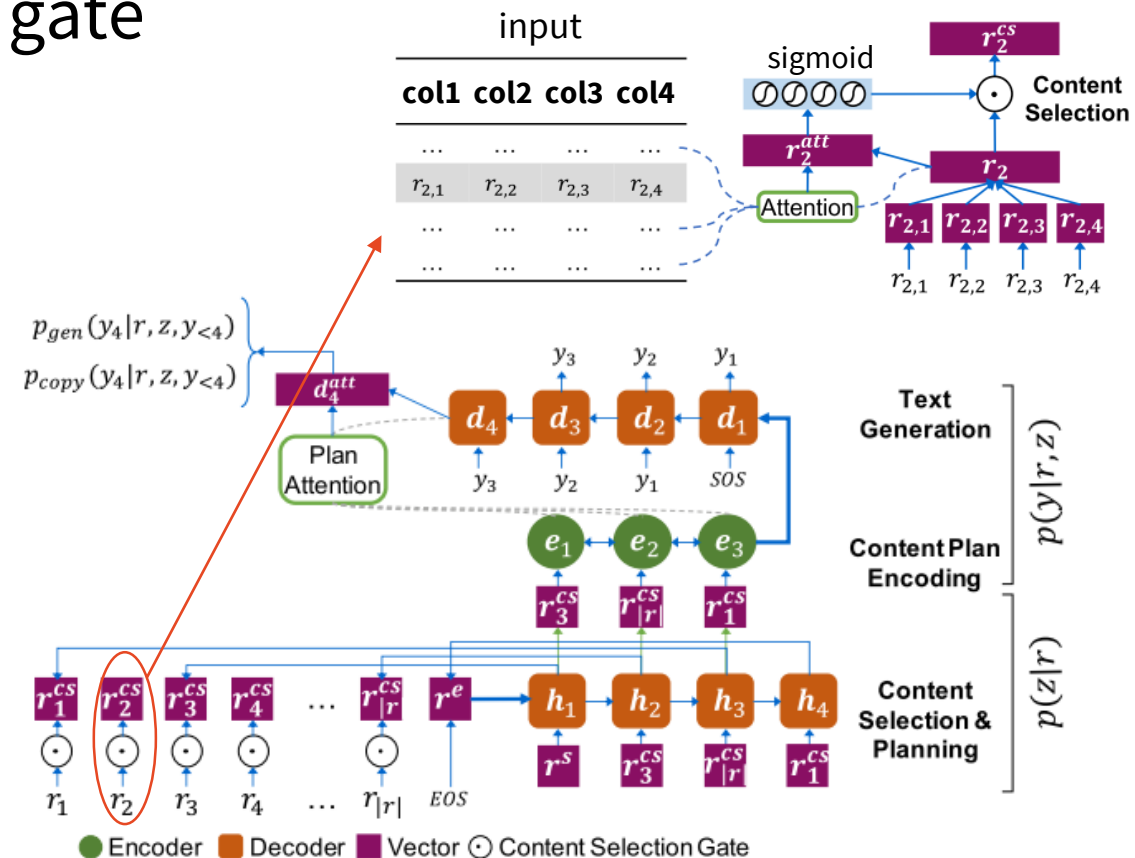
this is decoded  
given the prompt

Zizzi is a bar .



# 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

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

## 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: field goals, CITY: player team city.

## content plan

- automatic conversion
- content selection is done here (shown for 1<sup>st</sup> 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 glut 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

(Moryossef et al., 2019)  
<http://arxiv.org/abs/1904.03396>

- create explicit text plans by aggregating inputs

- RDF triples → list of trees (one per sentence)

- joining + ordering ( $\leftrightarrow$ )

- 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.  $p(\text{capital}|\text{residence})$ )
- can select the best plan, or a random highly-rated one
  - most plans beyond a certain threshold are fine

- training plans extracted automatically

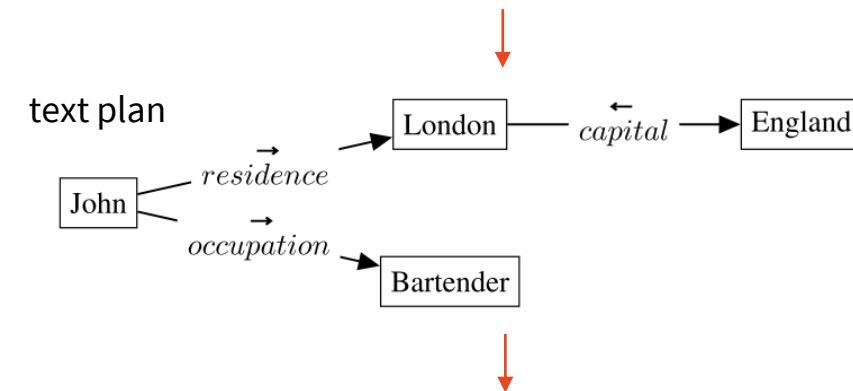
- text is consistent with a plan if it has the right sentence split & assignment + order of entities
- relations are not checked (this is much harder than for entities)

- sentence-by-sentence generation: pointer-generator net

- more faithful than generating everything in one step

input RDF

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

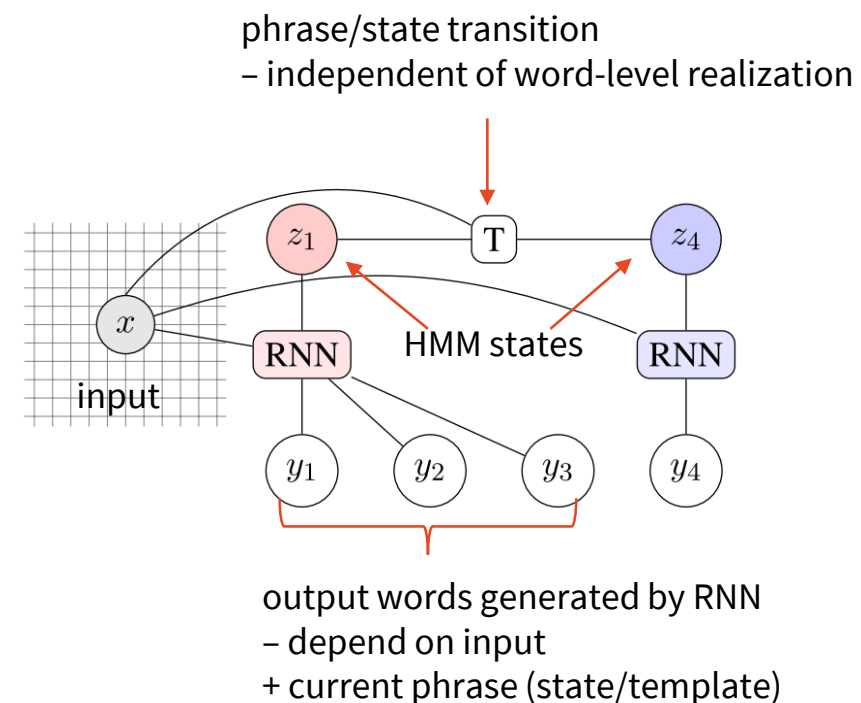


John lives in London, the capital of England,  
and works as a bartender.

$\Pi$  of cond.  
distributions

# 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



# Hidden Semi-Markov Model

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

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

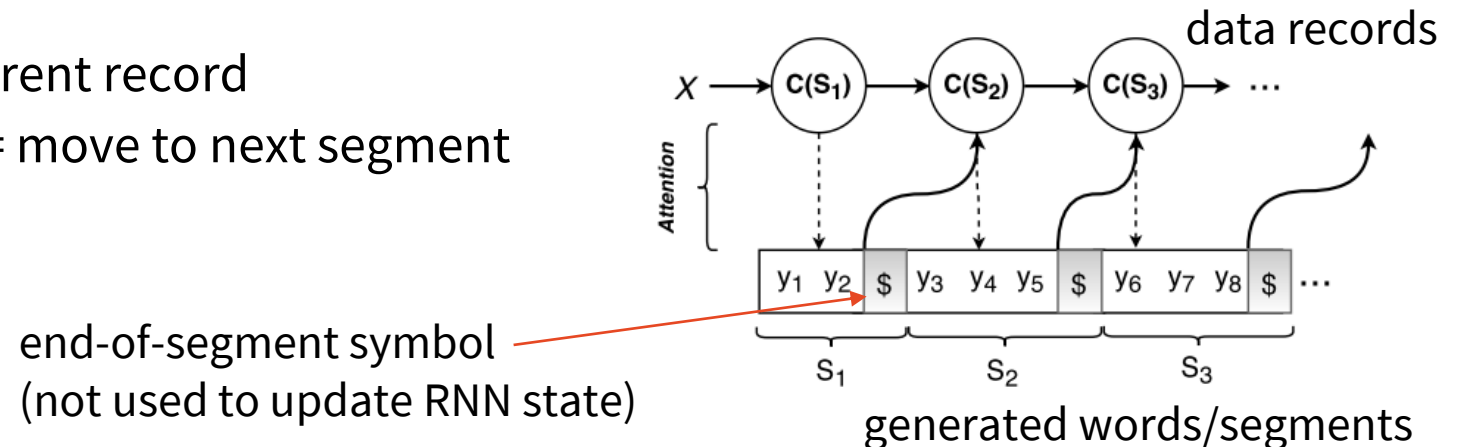
name[\_], type[\_], rating[\_], food[\_], area[\_], price[\_]

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

The \_\_\_\_\_ | is a | providing  
\_\_\_\_\_ | is an | serving  
... | is an expensive | offering  
... | ... |  
food | in the | price range  
cuisine | with a | price bracket | It's  
foods | and has a | pricing | It is  
... | ... | . | The place is  
... | ... | ... |  
located in the | Its customer rating is  
located near | Their customer rating is  
near | Customers have rated it |  
... | ... |



- Same idea, just 1 segment = 1 data record
  - exception: “null record” for phrases like *and, is a, there is*
  - HSMM used more fine-grained segments – this has better interpretation
- Segment-record alignment is learned + explicit
  - enumerating all possibilities – forward/EM algorithm
- Generation on 2 levels, using LSTMs:
  - choosing record  $\rightarrow$  decoding its words
  - record transitions – depends on 1 previous record & all previous words
  - word generation
    - attention is limited to current record
    - end-of-segment symbol = move to next segment



- Input: tree-shaped MRs
  - hierarchy: discourse relation  $\downarrow$  dialogue act  $\downarrow$  slot
  - can be automatically induced, much flatter than usual syntactic trees
    - discourse connectives, sentence splits
  - could potentially use other tree-like structures (such as the text plans made from RDF)
- 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/pointer-generator, 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*

# Realizing from Trees

- 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

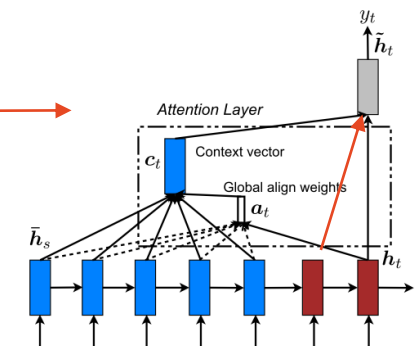
- all of this improves consistency & data-efficiency

**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



- 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)

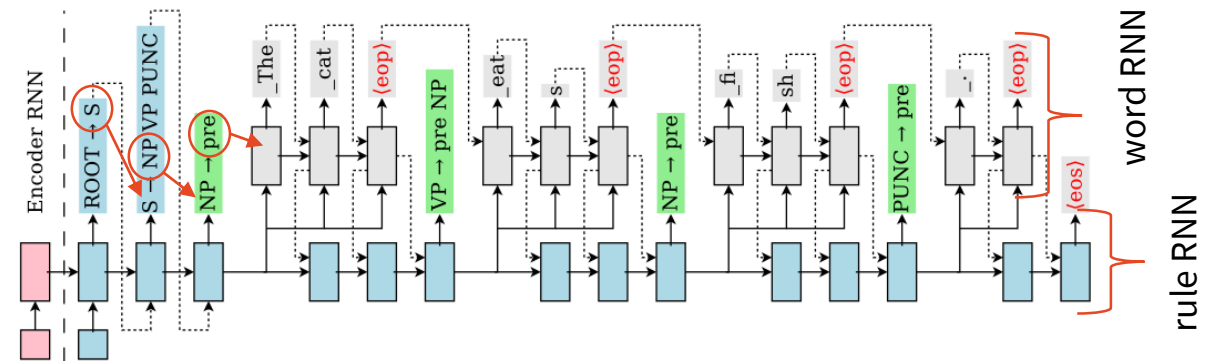
- **word RNN**

- 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

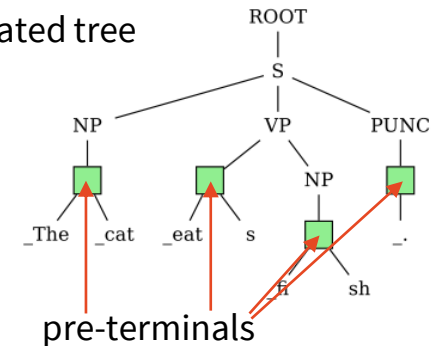
- can work for any type of trees

- but found to work best for binary trees without linguistic information 🤔

decoding process



generated tree



# “Unsupervised” NLG

- 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)
- 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

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

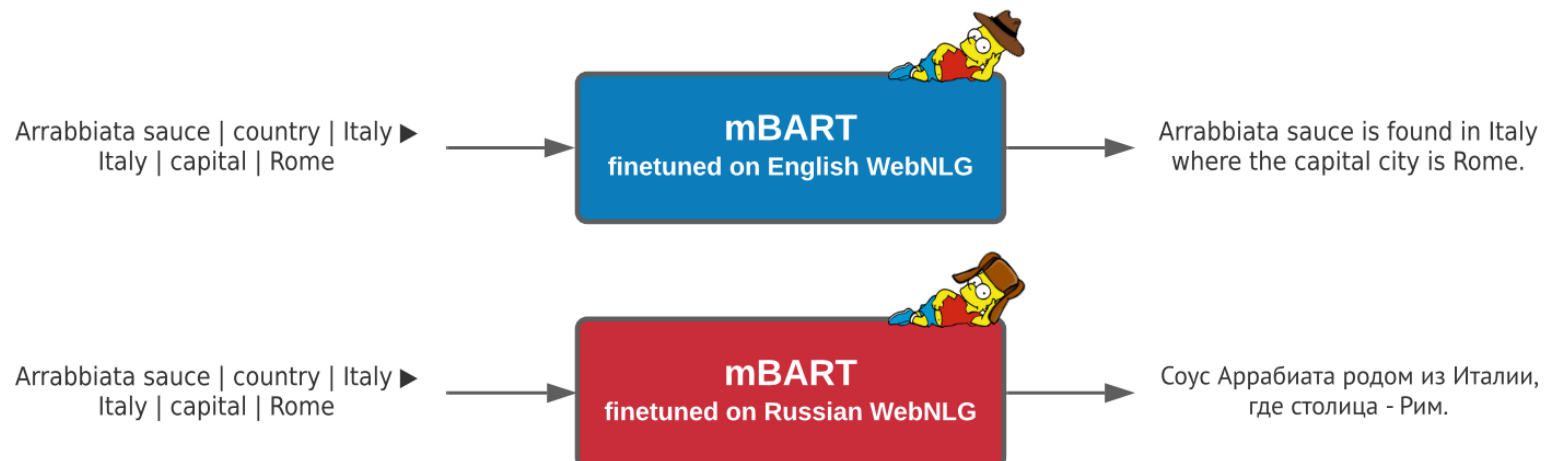
- BART ~ BERT (encoder) + GPT-2 (decoder)
  - LM pretrained for denoising autoencoding
- works nicely when finetuned for data-to-text
  - encode linearized data, decode text
  - just like seq2seq
- multilingual BART → allows multilingual generation
  - the model works well for machine translation
  - can generate Russian outputs from English triples

(Lewis et al., 2019)

<https://arxiv.org/abs/1910.13461>

(Liu et al., 2020)

<http://arxiv.org/abs/2001.08210>



# Summary

- good NLG = seq2seq + reranking
  - problems: hallucination, omission etc.
- improvements:
  - GPT-2 + RoBERTa reranking
  - data manipulation: cleaning, augmentation
  - NLG-NLU joint training
    - for data cleaning, augmentation, semi-supervised
  - 2-step: planning & realization
  - more supervision – tree decoding
- “unsupervised” NLG – denoising (incl. BART – pretrained for denoising)

# Thanks

## Contact us:

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

**Labs in 10 minutes**

**Project topic description – today!**

**Next week: End-to-end models**

## 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>