NPFL099 Statistical Dialogue Systems
8. Natural Language Generation

http://ufal.cz/npfl099

Ondřej Dušek, Vojtěch Hudeček & Tomáš Nekvinda

22.11.2021

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

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

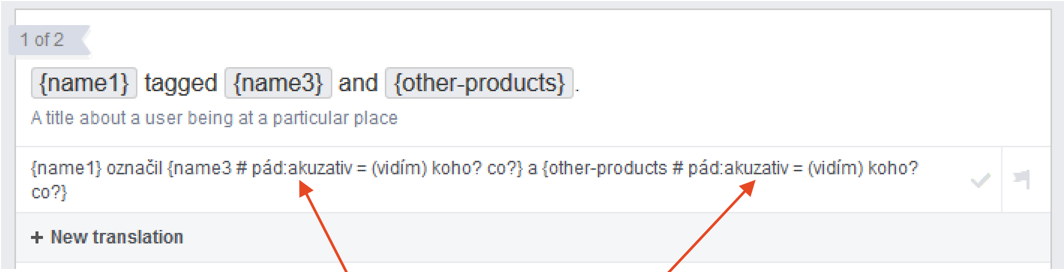
NLG Basic Approaches

- **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

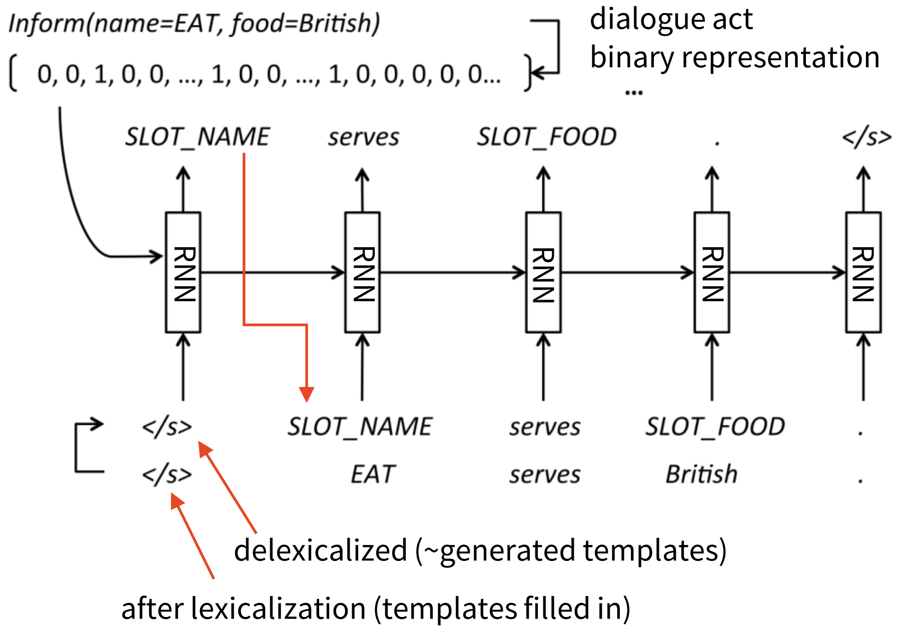
Neural End-to-End NLG: RNNLG

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

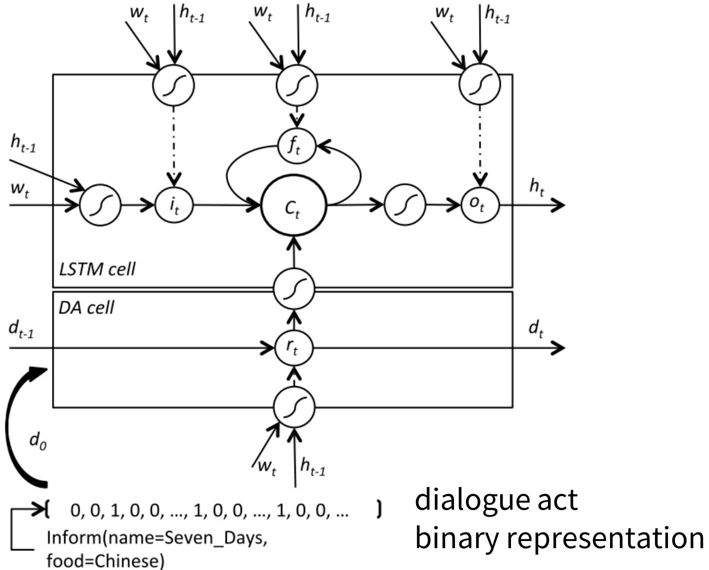
- Unlike previous, doesn’t need alignments
  - no need to know which word/phrase corresponds to which slot

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

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



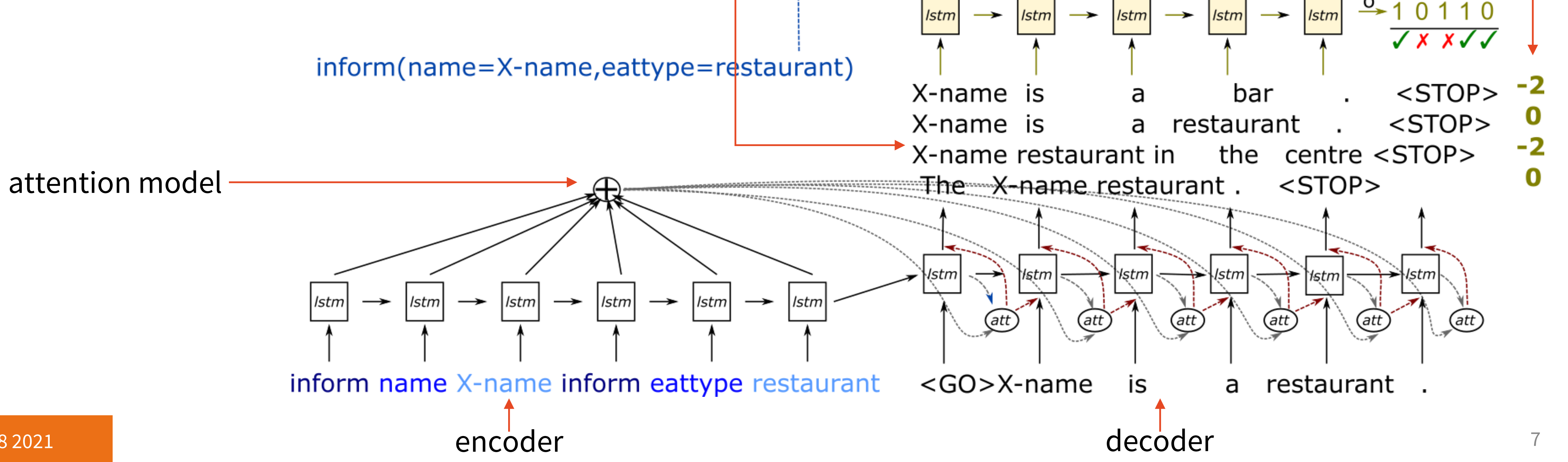
- Using RNNs, generating word-by-word
  - neural language models conditioned on DA
  - generating delexicalized texts
- input DA represented as binary vector
- Enhanced LSTM cells (SC-LSTM)
  - special part of the cell (gate) to control slot mentions



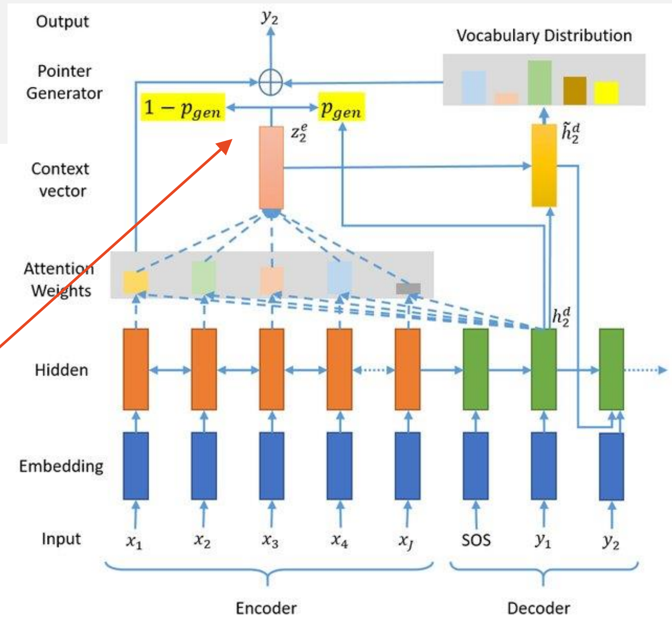
Seq2seq NLG (TGen)

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

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



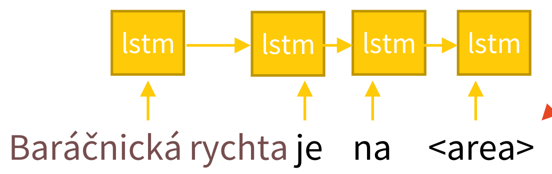
Delexicalization vs. Copy/Pointer net



- 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)

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

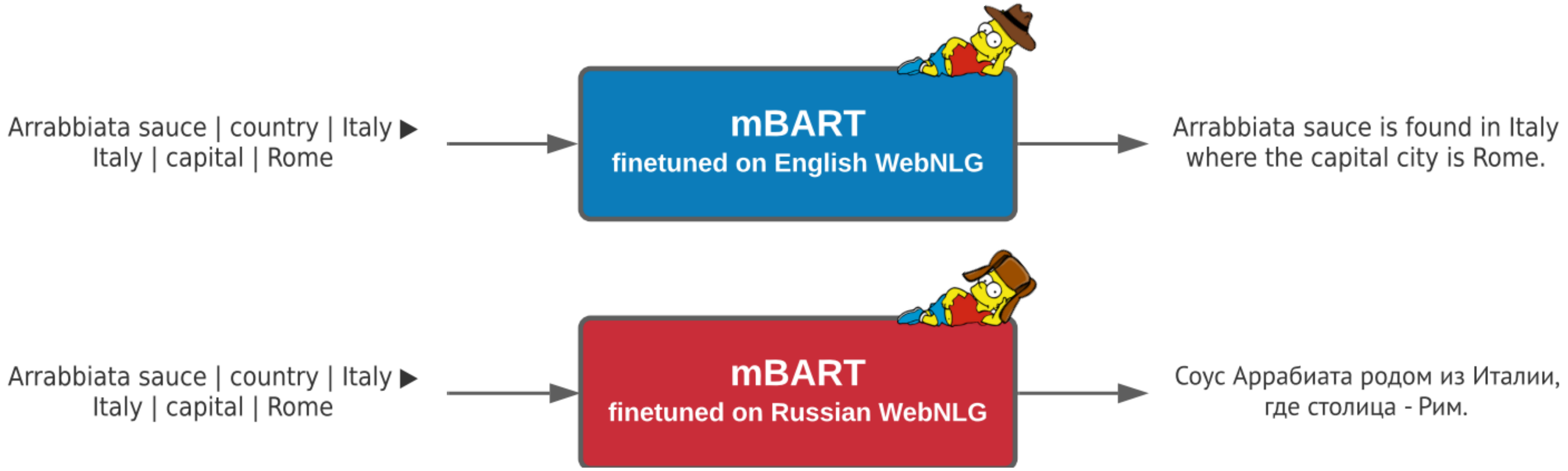
(Shi et al., 2018) http://arxiv.org/abs/1812.02303
(Dušek & Jurčíček, 2019) https://arxiv.org/abs/1910.05298

Pretrained LMs

(Kasner & Dušek, 2020) https://aclanthology.org/2020.webnlg-1.20/
(Kale & Rastogi, 2020) https://www.aclweb.org/anthology/2020.inlg-1.14

- BART (or T5) – encoder-decoder LM (Lewis et al., 2019) https://arxiv.org/abs/1910.13461
  - pretrained for **denoising** autoencoding
- works nicely when simply finetuned for data-to-text
  - encode linearized data, decode text, just like seq2seq
- mBART (multilingual) → allows multilingual generation (Liu et al., 2020) http://arxiv.org/abs/2001.08210
  - can generate Russian outputs from English triples
- You can even recast whole NLG as denoising (“unsupervised”)
  - train seq2seq for “important words” → sentence
  - use slot values as the important words

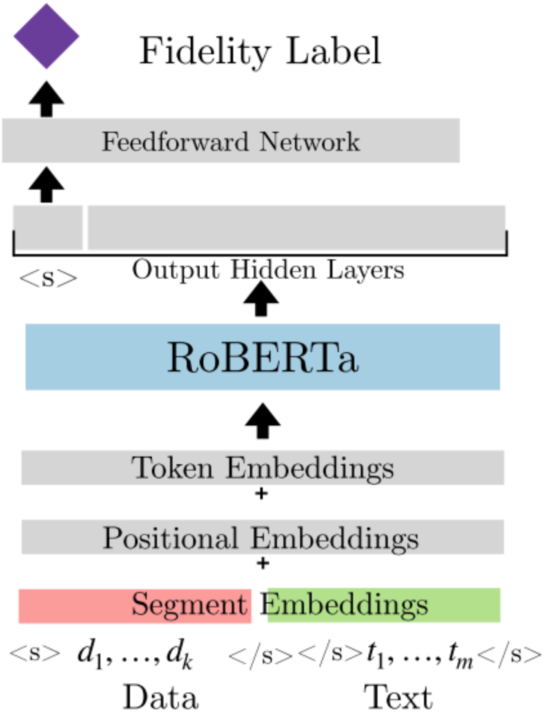
(Freitag & Roy, 2018) http://aclweb.org/anthology/D18-1426



Pretrained LMs with Reranking

(Harkous et al., 2020)
http://arxiv.org/abs/2004.06577

- Basically the same as seq2seq + reranking
  - just with GPT-2 & RoBERTa instead of LSTMs
- GPT-2 fine-tuned for $\underbrace{\text{<data> name[Zizzi] eatType[bar] <text>}}_{\text{prompt (fed into GPT-2)}}$ $\overbrace{\text{Zizzi is a bar .}}^{\text{this is decoded given the prompt}}$
  - 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)



Problems with neural NLG

(Dušek et al., 2020)
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, pretrained LMs

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

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

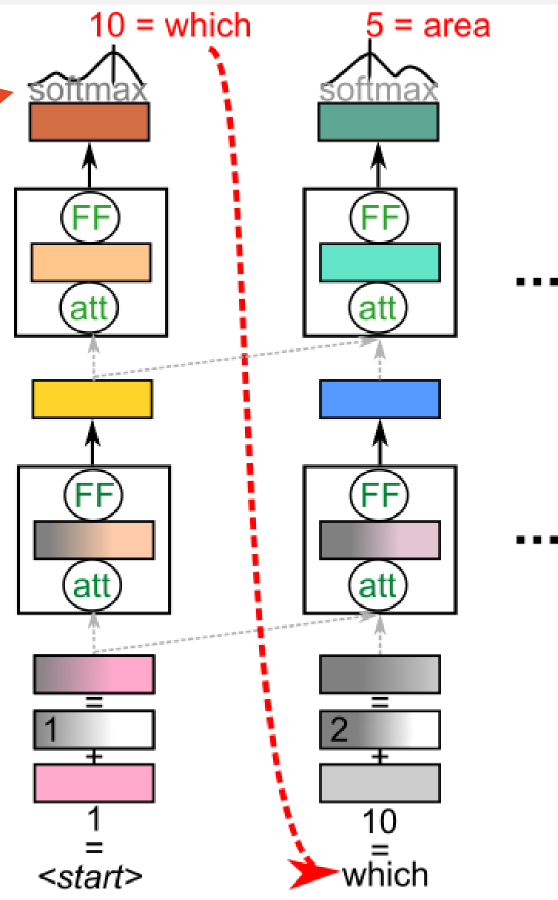
- Still more hassle than writing up templates 😏
- Some approaches to counter this follow (→), but none are perfect

Decoding approaches

https://huggingface.co/blog/how-to-generate
https://towardsdatascience.com/decoding-strategies-that-you-need-to-know-for-response-generation-ba95ee0faadc

- same model, different approaches to choosing words
  - sequence generation models have a softmax on top
  - up to you what (sub)word you choose & feed back to the model
  - large influence on the generation outputs – quality & diversity
- **greedy** – basic, always do the arg max
- **sampling** – can be wild (top-k/nucleus counter this)
  - **random** – sample according to softmax distribution
  - **top-k** – choose just top k options (~5-500), sample from them
  - **nucleus** – choose top options that cover $\geq p$ probability (~0.9)
- **beam search** – can be too conservative, still not optimal
  - try n continuations for each of n hypotheses, then discard all but n best
  - lends itself to reranking well
- in addition, you can e.g. penalize repeated tokens

(Holtzmann et al., 2020)
https://arxiv.org/abs/1904.09751



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
- 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

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

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

NLG-NLU Combo: Self-training

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

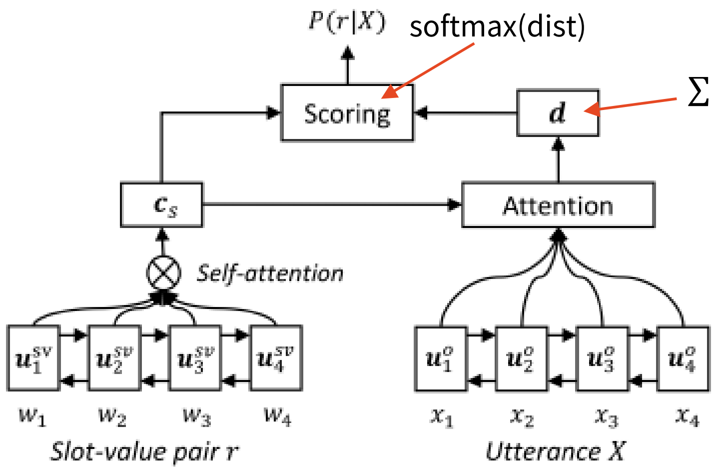
RNN + rule/CNN | seq gen + classif

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

      → ensure clean generated data
  - Train generator on original & sampled data (can loop more)
- Near perfect accuracy with basic seq2seq+attention as generator
  - on E2E restaurants data (relatively simple but noisy dataset)

NLG-NLU Combo: NLU data cleaning

- NLU used to clean training data
  - NLU model – BiLSTM + attention & vector distance ranking (choose "closest" value ∀slot)
- 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(%)
TGen	65.90	18.09 (114/630)
Slug2Slug	66.19	6.51 (41/630)
Seq2Seq	66.15	69.37 (374/630)
Seq2Seq+aug	**66.49**	28.89 (182/630)
Seq2Seq+aug+iter	65.63	2.07 (13/630)
Seq2Seq+aligner	63.81	**1.75** (11/630)

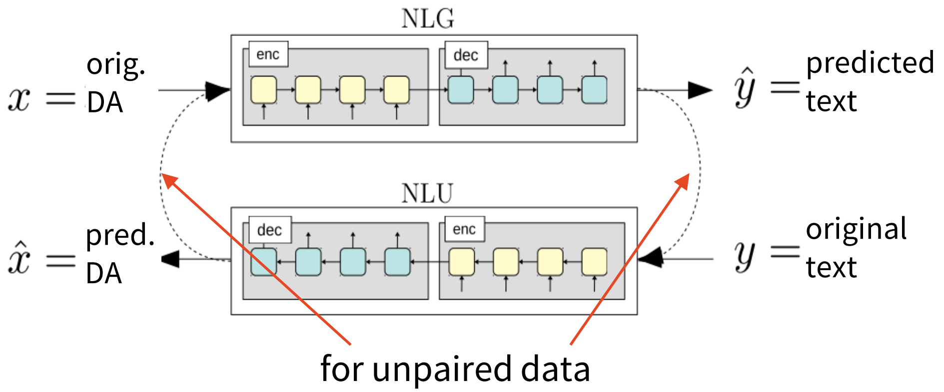
original data → Seq2Seq
plain supervised NLU → Seq2Seq+aug
iterative NLU training → Seq2Seq+aug+iter
handcrafted NLU → Seq2Seq+aligner

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

NLG-NLU Combo: Semi-supervised

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

- learn from partially unpaired data
  - some DA-text pairs, some loose DAs, some loose texts
- similar to previous: symmetric models, joint optimization
  - $\text{loss} = \alpha \cdot \text{loss}_{\text{NLG}}^{\text{paired}} + \beta \cdot \text{loss}_{\text{NLG}}^{\text{unpaired}} + \gamma \cdot \text{loss}_{\text{NLU}}^{\text{paired}} + \delta \cdot \text{loss}_{\text{NLU}}^{\text{unpaired}}$
  - losses for paired data are as usual (MLE, seq2seq models)
  - unpaired case: models are connected, reconstruction loss
    - loss is difference from original text/DA when passing through the whole loop
    - greedy decoding
    - trick for making it fully differentiable:
      Straight-Through Gumbel-Softmax
      - Gumbel-Softmax: approximate sampling from categorial token distributions (Lecture 4)
      - straight-through = real (hard) sampling for forward pass, smooth approximation for backward pass



Few-shot NLG with Pretrained LMs

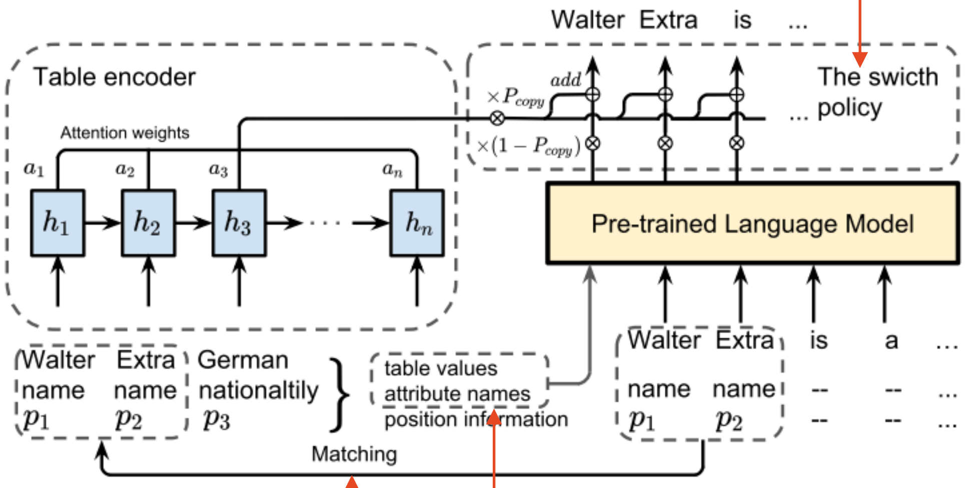
(Chen et al., 2020)
https://www.aclweb.org/anthology/2020.acl-main.18/

- 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

generate from LM
or copy from input?



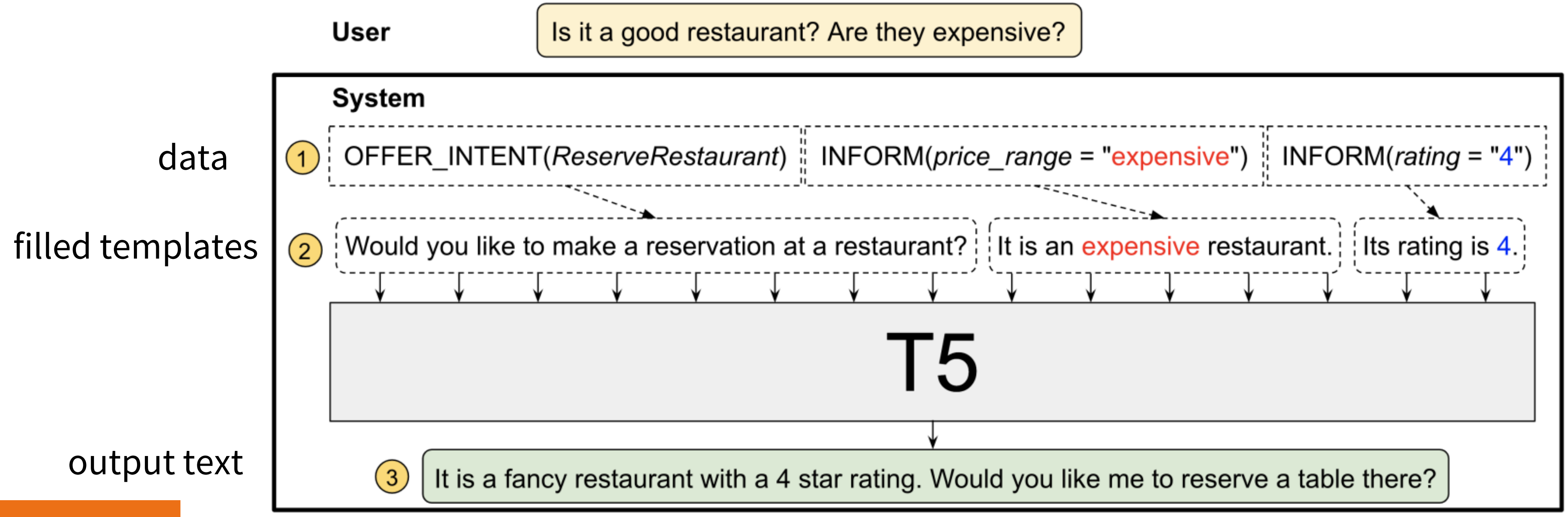
newly trained LM context

during training:
to find out where to copy inputs

Few-shot: Templates + Pretrained LM

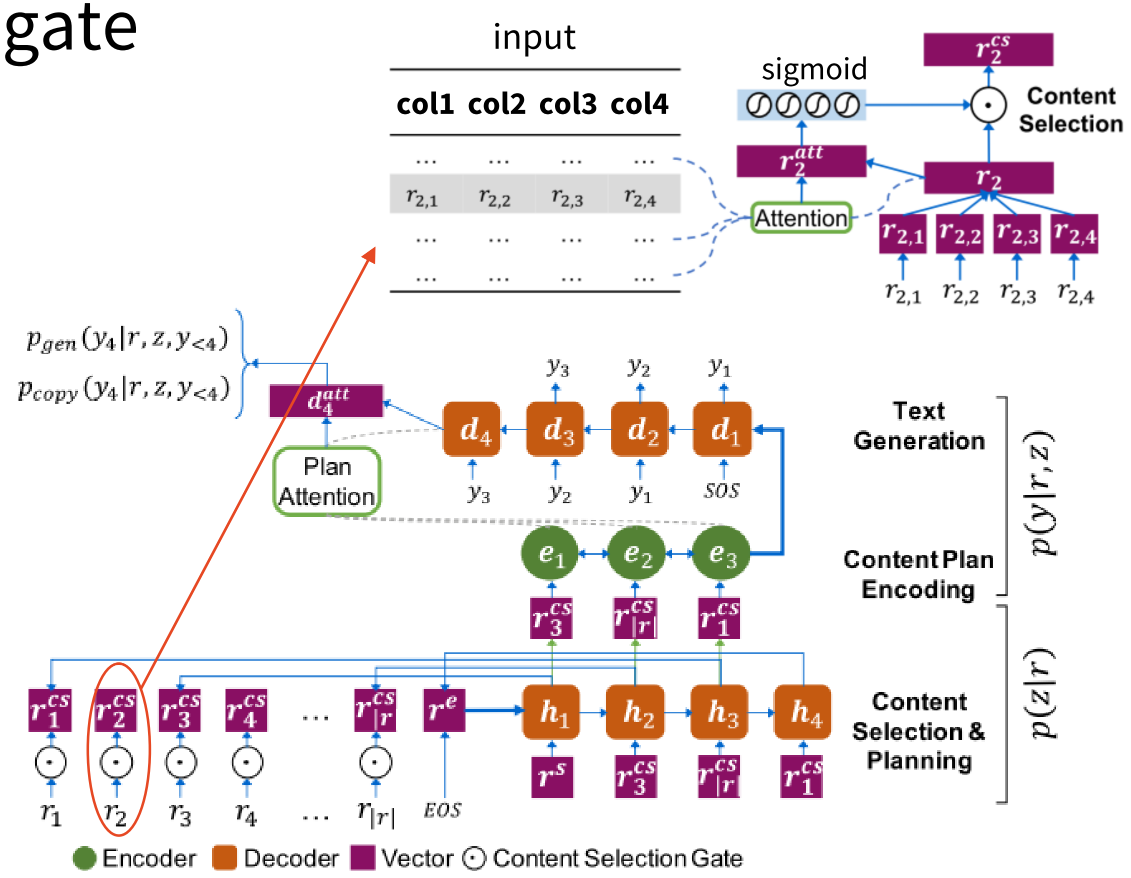
(Kale & Rastogi, 2020)
https://www.aclweb.org/anthology/2020.emnlp-main.527

- Have some simple templates (1 piece of info each)
  - a bit of handcrafting, but manageable for many datasets
- Use pretrained LMs (e.g. T5/BART) to combine them into nice sentences
  - basically text-to-text denoising, i.e. what the models were originally trained to do
- Works well, needs less data, generalizes to new domains



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



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

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 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
...	...	...	...

team ID – home/visiting

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**.

Two-step: content planning & realization

seq2seq + copy | seq gen

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

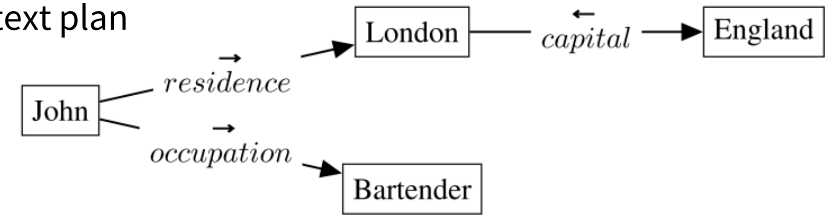
- create explicit text plans by aggregating inputs
  - RDF triples → list of trees (one per sentence)
    - joining + ordering (←→)
  - create all possibilities + rank
    - product of experts for given features:
      - individual arrow directions
      - % of reversed
      - sentence split + # of 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

seq2seq + copy | seq gen

- 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 (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

(Balakrishnan et al., 2019) http://arxiv.org/abs/1906.07220
(Rao et al., 2019) https://www.aclweb.org/anthology/W19-8611/
(Li et al., 2021) https://aclanthology.org/2021.inlg-1.10

tree/seq2seq + copy | seq gen

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

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

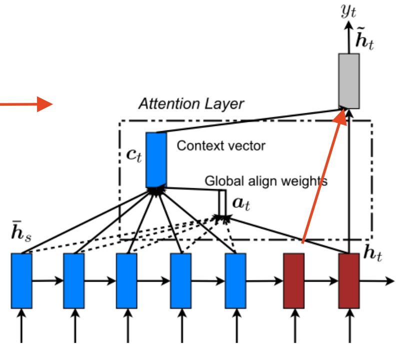
(2) [INFORM [name name] is] [CONTRAST [customerrating_high highly rated] but [pricerange_expensive expensive] .]

OK

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

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
- can be used for **self-training** → even more perf. gain



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

Summary

- **NLG**: system DA → text
  - templates work pretty well
  - **seq2seq** & similar = best data-driven
    - problems: hallucination, not enough diversity
    - fixes: reranking, delexicalization/copy nets, ensembling
- 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,nekvinda}@ufal.mff.cuni.cz
Skype/Meet/Zoom/Troja (by agreement)

Labs in 10 minutes
Assignment 4

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