8. Natural Language Generation

http://ufal.cz/npfl099

Zdeněk Kasner, Ondřej Dušek & Vojtěch Hudeček

21.11.2022
Natural Language Generation

= task of automatically producing text in e.g. English (or any other language)

• covers many subtasks:

<table>
<thead>
<tr>
<th>task</th>
<th>input</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>unconditional language generation</td>
<td>Ø</td>
<td>arbitrary text</td>
</tr>
<tr>
<td>conditional language generation</td>
<td>short text prompt</td>
<td>continuation of the prompt</td>
</tr>
<tr>
<td>machine translation</td>
<td>text in language A</td>
<td>text in language B</td>
</tr>
<tr>
<td>summarization</td>
<td>long text</td>
<td>text summary</td>
</tr>
<tr>
<td>question answering</td>
<td>question</td>
<td>answer</td>
</tr>
<tr>
<td>image captioning</td>
<td>image</td>
<td>image caption</td>
</tr>
<tr>
<td>data-to-text generation</td>
<td>structured data</td>
<td>description of the data</td>
</tr>
<tr>
<td>dialogue response generation</td>
<td>dialogue act</td>
<td>system response</td>
</tr>
</tbody>
</table>

NLG in a narrow sense
NLG Objectives

• general NLG objective:
  
  given input & communication goal
  create accurate + natural, well-formed, human-like text

• additional NLG desired properties:
  • variation (avoiding repetitiveness)
  • simplicity (saying only what is intended)
  • adaptability (conditioning on e.g. user model)
NLG in Dialogue Systems

• in the context of dialogue systems:

NLG: **system action → system response**

"what the system wants to say"  "actually saying it"

• system action
  • selected by the dialogue manager
  • may be conditioned on:
    • dialogue state
    • dialogue history (→ referring expressions, avoiding repetition)
    • user model (→ “user wants short answers”)
NLG Subtasks (Textbook Pipeline) = how proper NLG had to be done before neural approaches

NLG

-deciding what to say
  - inputs
  - content plan
  - sentence planning / microplanning
  - sentence plan
  - surface realization
  - text

-deciding how to say it
  - selecting content according to a communication goal (typically handled by dialogue manager in dialogue systems)
  - organizing content into sentences, merging sentences, choosing referring expressions
  - linearization according to grammar, word order, morphology
Example: classical NLG pipeline in medical domain

(a) Content Determination

(b) Text Structuring

(c) Lexicalisation etc.

(d) Realisation

(Event

(TYPE existential

(PRED be

(TENSE past

(ARGS

THEME \{b_1, b_2, b_3\}

MIN-VAL 69)

S

PRO

V

VP\textsubscript{past}

NP\textsubscript{pl}

PP

there

three successive bradycardias
down to 69

(Gatt & Krahmer, 2018)
NLG Basic Approaches

• **hand-written prompts** ("canned text")
  - most trivial – hard-coded, no variation
  - doesn’t scale (good for DTMF phone systems)

• **templates** ("fill in blanks")
  - 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
  - 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 commercial dialogue 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 (heuristics, random variation)
- can be a good starting point for ML algorithms
  - post-editing / reranking the templates with neural language models
Template-based NLG – Examples

Example: Facebook

{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}.

* Facebook, 2015 *

{user} sdílela (object-owner) uživatele (=album){title}

+ New translation

*(Facebook, 2019)*

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

A title about a user being at a particular place

{name1} označil (name3 # půda: kuzaltìv = (vidím) koho? co?) a {other-products # půda: kuzaltìv = (vidím) koho? co*}

+ New translation

inflection rules
Example: Dialogue assistants

**Alexa**

On the *Intents* detail page, the *Intent Slots* section after the *Sample Utterances* section displays the slots you add. When you highlight a word or phrase in an utterance, you can add a new slot or select an existing slot.

For example, the set of utterances shown earlier now looks like the following example.

- I am going on a trip on {travelDate}
- I want to visit {toCity}
- I want to travel from {fromCity} to {toCity} {travelDate}
- I'm {travelMode} from {fromCity} to {toCity}
- I'm {travelMode} to {toCity} to go {activity}


**Mycroft**

Rather than writing out all combinations of possibilities, you can embed them into a single line by writing each possible option inside parentheses with | in between each part. For example, that same intent above could be written as:

(https://mycroft-ai.gitbook.io/docs/mycroft-technologies/padatious)
Example: Research systems

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

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

- based on top of linguistic theories
- state-of-the-art research systems until NLG the arrival of NNs
- rules for building tree-like structures → rules for tree linearization
- reliable, more natural than templates
- takes a lot of effort, naturalness still not human-level
- see NPFL123 for more details

(Mille et al., 2019)
https://aclanthology.org/W19-8659.pdf
Neural NLG

- learning the task from the data
- sequence-to-sequence generation / editing / re-ranking
- fluency can match human-level, minimal hand-crafting
- not controllable ("black-box"), semantic inaccuracies (omissions / hallucinations), low diversity, expensive data gathering, expensive training, expensive deployment

→ promising research area 😊
**Seq2seq Generation**


- **encoder-decoder**
  - *RNN*: encoder updates the hidden state → decoder is initialized with the hidden state
  - *Transformer*: encoder generates a sequence of hidden states → decoder attends to this sequence
  - pretrained Transformers: BART, T5 (trained on sequence denoising)

- **decoder-only**
  - input sequence is prepended as a context, the decoder generates continuation
  - pretrained Transformer models: GPT-2, GPT-3 (trained on autoregressive language modelling)

- training vs. inference:
Decoding Algorithms

- for each time step $t$, the decoder outputs a probability distribution: $P(y_t \mid y_{1:t-1}, X)$
- how to use it?
- **exact inference**: find a sequence maximizing $P(y_{1:T} \mid X)$
  - not possible in practice (why? and is it our goal?)
- **approximation algorithms**
  - greedy search
  - beam search
- **stochastic algorithms**
  - random sampling
  - top-k sampling
  - nucleus sampling (=top-p sampling)

(+ repetition penalty → decreasing probabilities of generated tokens)
● **greedy search**: always take the argmax
  ○ does not necessarily produce the most probable sequence (why?)
  ○ often produces dull responses

Example:

**Context:**
Try this cake. I baked it myself.

**Optimal Response:**
This cake tastes great.

**Greedy search:**
This is okay.

many examples start with “This is”, no possibility to backtrack
Decoding Algorithms

- **beam search**: try $k$ continuations of $k$ hypotheses, keep $k$ best
  - better approximation of the most probable sequence, bounded memory & time
  - allows re-ranking generated outputs
  - $k=1 \rightarrow$ greedy search

---

**Reranking:**

-2.914 No route found later, sorry
-3.544 The next connection is not found.
-3.690 I'm sorry, I can not find a later ride.
-3.836 I can not find the next one sorry.
-4.003 I'm sorry, a later connection was not found.

(Ondřej's PhD thesis, Fig. 7.7)
- **top-k sampling**: choose top k options (~5-500), sample from them
  - avoids the long tail of the distribution
  - more diverse outputs
- **nucleus sampling**: choose top options that cover $\geq p$ probability mass ($\sim 0.9$)
  - “$k$” is adapted according to the distribution shape

\[
\sum_{w \in V_{\text{top-}p}} P(w|\text{“The”}) = 0.94
\]

\[
\sum_{w \in V_{\text{top-}p}} P(w|\text{“The”, “car”}) = 0.97
\]
RNN-based Approaches: TGen

- standard LSTM with attention
  - encoder – triples <intent, slot, value>
  - decodes words (possibly delexicalized)

- beam search & reranking
  - DA classification of outputs
  - checking against input DA

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

- using enhanced LSTM cells (SC-LSTM)
  - special gate to control slot mentions
- autoregressive generation
- conditioned on DA represented as binary vector
  - generating delexicalized texts
- domain adaptation
  - replacing delexicalized slots
  - very related domains only

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

- **copy mechanism** (see NLU & the next slide)
  - select (or interpolate) between:
    - generating a new token
    - copying a token from input
  - does away with the pre/postprocessing
- **inflection model**
  - useful for languages with rich morphology (e.g., Czech)
  - a simple LM such as RNN LM
- **pretrained models**
  - the model learns to copy and inflect words during pretraining → may work implicitly on downstream tasks
  - works well for high-resource languages

<table>
<thead>
<tr>
<th>Inflection Form</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>nominative</td>
<td>0.10</td>
</tr>
<tr>
<td>genitive</td>
<td>0.07</td>
</tr>
<tr>
<td>dative, locative</td>
<td>0.60</td>
</tr>
<tr>
<td>accusative</td>
<td>0.10</td>
</tr>
<tr>
<td>instrumental</td>
<td>0.03</td>
</tr>
</tbody>
</table>

(Dušek & Jurčiček, 2019)

Delexicalization Alternatives – Copy Mechanism

Probability of copying a token from the input: \( x(1 - p_{\text{gen}}) \)

Probability of generating a new token from the vocabulary: \( xp_{\text{gen}} \)

Final Distribution

"Argentina"

Context Vector

Vocabulary Distribution

Decoder Hidden States

Encoder Hidden States

Attention Distribution

Source Text

Partial Summary

(See et al., 2017)
http://arxiv.org/abs/1704.04368
Pretrained LMs

• GPT-2 or BART / T5 finetuned for NLG
  • different pretraining tasks – similar outcomes
• works nicely when simply finetuned for data-to-text
  • encode linearized data, decode text
  • the model learns copying implicitly
• mBART / mT5 (multilingual) → allows multilingual generation
  • can generate e.g. Russian outputs from English triples
• are we done now?

(Kale & Rastogi, 2020)
https://www.aclweb.org/anthology/2020.inlg-1.14

(Liu et al., 2020)

(Kasner & Dušek, 2020)
https://aclanthology.org/2020.webnlg-1.20/
Pretrained LMs with Reranking

- **aiming to improve semantic accuracy**
- seq2seq + reranking with GPT-2 & RoBERTa
- GPT-2 fine-tuned for <data> name[Zizzi] eatType[bar] <text> Zizzi is a bar .
  
  prompt (fed into GPT-2)  decoded given the prompt

- 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)
Few-shot NLG with Pretrained LMs

- learning from very few (less than ~200) training examples
- GPT-2 with a copy mechanism
  - LM fine-tuned, forced to copy inputs
  - additional loss term for copying
- retrieving “prototypes” guiding the generator
  - prototype: most similar exemplar according to BERT cosine similarity
  - prototype concatenated with the input
- few-shot prompting
  - prepending a few (~3) input-output examples as a context
  - generating the output with GPT-2 XL
  - no finetuning

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

(Su et al., 2021)
http://arxiv.org/abs/2108.12516

Few-shot Prompt

Translate Graph to English:

Graph:  

<
H

>

Paulo Sousa

<
R

>

CLUB

<
T

>

ACF Fiorentina

English: Paulo Sousa plays for ACF Fiorentina.

Graph:  

<
H

>

Dave Challinor

<
R

>

CLUB

<
T

> Colwyn Bay F.C.

English: Dave Challinor plays for Colwyn Bay F.C.

Graph:  

<
H

>

Alan Martin (footballer)

<
R

>

CLUB

<
T

> Hamilton Academical F.C.

English:
Few-shot NLG: Templates + Pretrained LM

- start with simple templates (one piece per system action)
  - a bit of handcrafting, but manageable for many datasets
- concatenate them and then use pretrained LMs (e.g. T5/BART) to rephrase them
  - basically text-to-text denoising, i.e. what the models were originally trained to do
- needs less data & generalizes to new domains

**templates**

<table>
<thead>
<tr>
<th>Template</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIRM!!date!!@</td>
<td>The date is @.</td>
</tr>
<tr>
<td>CONFIRM!!party_size!!@</td>
<td>The reservation is for @ people.</td>
</tr>
<tr>
<td>CONFIRM!!restaurant_name!!@</td>
<td>Booking a table at @.</td>
</tr>
<tr>
<td>CONFIRM!!time!!@</td>
<td>The reservation is at @.</td>
</tr>
<tr>
<td>GOODBYE</td>
<td>Have a good day.</td>
</tr>
<tr>
<td>INFORM!!cuisine!!@</td>
<td>They serve @ kind of food.</td>
</tr>
<tr>
<td>INFORM!!has_live_music!!False</td>
<td>They do not have live music.</td>
</tr>
<tr>
<td>INFORM!!has_live_music!!True</td>
<td>They have live music.</td>
</tr>
</tbody>
</table>

**system**

User: Is it a good restaurant? Are they expensive?

System:

1. **OFFER_INTENT(ReserveRestaurant)**
   - **INFORM(price_range = "expensive")**
   - **INFORM(rating = "4")**

2. **Would you like to make a reservation at a restaurant?**
   - It is an **expensive** restaurant.
   - Its rating is **4.**

3. **It is a fancy restaurant with a 4 star rating. Would you like me to reserve a table there?**

(Kale & Rastogi, 2020)
https://www.aclweb.org/anthology/2020.emnlp-main.527
Zero-shot NLG: Templates + Pretrained LM

- **NLG without human-written references**
- start with templates → postprocess them with BART-based models
  - trained for text-based operations learned from Wikipedia

  ![Diagram of template usage](image)

  (Kasner & Dušek, 2022)

- improvement: using prompted GPT-3 instead of hand-crafted templates

  (Xiang et al., 2022)
Content Planning: Content Selection

• **explicit content selection**
• usually done by DM in dialogue systems
• needed for complex inputs, e.g. sports report generation
  • records (team / entity / type / value) → summary
  • content selection: pointer network
• still largely unsolved problem w.r.t. semantic accuracy

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

(seq2seq + copy) seq gen

(Thomson & Reiter, 2022)
http://arxiv.org/abs/2108.05644
Example of NLG with content planning

source statistics (excerpt)

<table>
<thead>
<tr>
<th>TEAM</th>
<th>WIN</th>
<th>LOSS</th>
<th>PTS</th>
<th>FG_PCT</th>
<th>RB</th>
<th>AST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pacers</td>
<td>4</td>
<td>6</td>
<td>99</td>
<td>42</td>
<td>40</td>
<td>17</td>
</tr>
<tr>
<td>Celtics</td>
<td>5</td>
<td>4</td>
<td>105</td>
<td>44</td>
<td>47</td>
<td>22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PLAYER</th>
<th>H/V</th>
<th>AST</th>
<th>RB</th>
<th>PTS</th>
<th>FG</th>
<th>CITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jeff Teague</td>
<td>H</td>
<td>4</td>
<td>3</td>
<td>20</td>
<td>4</td>
<td>Indiana</td>
</tr>
<tr>
<td>Miles Turner</td>
<td>H</td>
<td>1</td>
<td>8</td>
<td>17</td>
<td>6</td>
<td>Indiana</td>
</tr>
<tr>
<td>Isaiah Thomas</td>
<td>V</td>
<td>5</td>
<td>0</td>
<td>23</td>
<td>4</td>
<td>Boston</td>
</tr>
<tr>
<td>Kelly Olynyk</td>
<td>V</td>
<td>4</td>
<td>6</td>
<td>16</td>
<td>6</td>
<td>Boston</td>
</tr>
<tr>
<td>Amir Johnson</td>
<td>V</td>
<td>3</td>
<td>9</td>
<td>14</td>
<td>4</td>
<td>Boston</td>
</tr>
</tbody>
</table>


content plan (for the 1st sentence)

<table>
<thead>
<tr>
<th>Value</th>
<th>Entity</th>
<th>Type</th>
<th>H/V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boston</td>
<td>Celtics</td>
<td>TEAM-CITY</td>
<td>V</td>
</tr>
<tr>
<td>Celtics</td>
<td>Celtics</td>
<td>TEAM-NAME</td>
<td>V</td>
</tr>
<tr>
<td>Indiana Pacers</td>
<td>Pacers</td>
<td>TEAM-CITY</td>
<td>H</td>
</tr>
<tr>
<td>Pacers</td>
<td>Pacers</td>
<td>TEAM-NAME</td>
<td>H</td>
</tr>
<tr>
<td>22</td>
<td>Pacers</td>
<td>TEAM-PCTS</td>
<td>H</td>
</tr>
<tr>
<td>22</td>
<td>Pacers</td>
<td>TEAM-FG_PCTS</td>
<td>H</td>
</tr>
<tr>
<td>5</td>
<td>Celtics</td>
<td>TEAM-WIN</td>
<td>V</td>
</tr>
<tr>
<td>4</td>
<td>Celtics</td>
<td>TEAM-LOSS</td>
<td>V</td>
</tr>
<tr>
<td>Isaiah Thomas</td>
<td>Isaiah Thomas</td>
<td>FIRST_NAME</td>
<td>V</td>
</tr>
<tr>
<td>Isaiah Thomas</td>
<td>Isaiah Thomas</td>
<td>SECOND_NAME</td>
<td>V</td>
</tr>
<tr>
<td>Isaiah Thomas</td>
<td>Isaiah Thomas</td>
<td>PTS</td>
<td>V</td>
</tr>
<tr>
<td>Isaiah Thomas</td>
<td>Isaiah Thomas</td>
<td>AST</td>
<td>V</td>
</tr>
<tr>
<td>Isaiah Thomas</td>
<td>Isaiah Thomas</td>
<td>FGM</td>
<td>V</td>
</tr>
<tr>
<td>Isaiah Thomas</td>
<td>Isaiah Thomas</td>
<td>FGA</td>
<td>V</td>
</tr>
<tr>
<td>Kelly Olynyk</td>
<td>Kelly Olynyk</td>
<td>FIRST_NAME</td>
<td>V</td>
</tr>
<tr>
<td>Kelly Olynyk</td>
<td>Kelly Olynyk</td>
<td>SECOND_NAME</td>
<td>V</td>
</tr>
<tr>
<td>Kelly Olynyk</td>
<td>Kelly Olynyk</td>
<td>Pts</td>
<td>V</td>
</tr>
<tr>
<td>Kelly Olynyk</td>
<td>Kelly Olynyk</td>
<td>REB</td>
<td>V</td>
</tr>
<tr>
<td>Kelly Olynyk</td>
<td>Kelly Olynyk</td>
<td>AST</td>
<td>V</td>
</tr>
</tbody>
</table>

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 glutony 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.

(Puduppully et al., 2019)

http://arxiv.org/abs/1809.00582
Content Planning: Ordering & Aggregation

- ordering the facts + aggregating them into sentences
- content already selected at this point
- can help the generator not to miss any facts
- for graphs with oriented edges:
  - generating all possible content plans using DFS (possibly pruning unpromising branches) → re-ranking the plans using a feature-based classifier
- for a set of key-value pairs:
  - using Conditional Random Field (CRF) for finding the optimal plan

(Moryossef et al., 2019a,b)

(Su et al., 2020)
http://arxiv.org/abs/2108.13740
• **NLG with tree-shaped inputs**
  
  • simple case: discourse relations (discourse connectives, sentence splits) between individual fields
    
    • much flatter than usual syntactic trees
  
  • improvements to account for the input structure:
    
    • constrained beam search decoding, tree-LSTM, self-training on synthetic data
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

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

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

- using synthetic data for improving model performance and robustness
- quite hard for NLG
- approaches:
  - task-specific pretraining on synthetic data
  - mixing synthetic data with the training data
- where to get the data:
  - Run labeled text → information extraction (OpenIE)

(Elder et al., 2020)
(Montella et al., 2020)
Summary

• **NLG**: system action → system response
• **templates** work pretty well
• **seq. generation** with pretrained Transformer LMs = best among data-driven
• problems – hallucination, not enough diversity, needs lots of data
  • reranking (with NLU), delexicalization/copy nets, pretrained models
  • different decoding styles (sampling)
  • NLG-NLU joint training
    • for data cleaning, self-training, semi-supervised training
• 2-step: planning & realization
• tree decoding – more supervision
• few-shot with pretrained LMs
• data manipulation: cleaning, augmentation
Thanks

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

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

Icons from https://www.flaticon.com/