NPFL099 Statistical Dialogue Systems

8. Natural Language Generation

http://ufal.cz(npfl099)

Ondřej Dušek, Vojtěch Hudeček & Tomáš Nekvinda

22.11.2021
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
  
• 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
      e.g. *restaurant* vs. *it*

• Sentence plan
  • ↓ **Surface realization**
    • linearization according to grammar
    • word order, morphology

• Text
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

(Alex public transport information rules)

https://github.com/UFAL-DSG/alex
Neural End-to-End NLG: RNNLG

- 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

(Wen et al, 2015; 2016)
http://aclweb.org/anthology/D15-1199
http://arxiv.org/abs/1603.01232
Seq2seq NLG (TGen)

- 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

(Shi et al., 2018)  http://arxiv.org/abs/1812.02303
• BART (or T5) – encoder-decoder LM
  • 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
  • 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

- Basically the same as seq2seq + reranking
  - just with GPT-2 & RoBERTa instead of LSTMs
- GPT-2 fine-tuned for \(<\text{data}>\ name[Zizzi]\ eatType[bar]\ <\text{text}>Zizzi\ is\ a\ bar.\)
  - 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)

(Harkous et al., 2020)
Problems with neural NLG

• 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
• Diversity & complexity of outputs
  • still can’t match humans by far
  • needs specific tricks to improve this
    • vanilla seq2seq models tend to produce repetitive outputs
• Still more hassle than writing up templates 😞
• Some approaches to counter this follow (→), but none are perfect

(Dušek et al., 2020)
http://arxiv.org/abs/1901.07931

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

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

- 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
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

---

**Example corrections**

**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.

**Correction:**
- *name:* Cotto
- *eatType:* coffee shop
- *food:* English
- *priceRange:* less than £20
- *customerRating:* low
- *area:* riverside
- *near:* The Portland Arms

**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:**
- *priceRange:* changed area

**Reference:** Cotto is a cheap coffee shop with one-star located near The Portland Arms.

**Correction:**
- *removed area*

---

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

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

(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)
• **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
  • Sample more data from it
    • 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
      • 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
  • 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

Original data
plain supervised NLU
iterative NLU training
handcrafted NLU

 Nie et al., 2019
https://www.aclweb.org/anthology/P19-1256
NLG-NLU Combo: Semi-supervised

- learn from partially unpaired data
  - some DA-text pairs, some loose DAs, some loose texts
- similar to previous: symmetric models, joint optimization
  - $\text{loss} = \alpha \cdot \text{loss}_{\text{NLG, paired}} + \beta \cdot \text{loss}_{\text{NLG, unpaired}} + \gamma \cdot \text{loss}_{\text{NLU, paired}} + \delta \cdot \text{loss}_{\text{NLU, unpaired}}$
  - 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

(Qader et al., 2019)

---

NPFL099 L8 2021
Few-shot NLG with Pretrained LMs

- **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

(Chen et al., 2020)
https://www.aclweb.org/anthology/2020.acl-main.18/
Few-shot: Templates + Pretrained LM

- 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

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

<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>
<th>...</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>
<td></td>
</tr>
<tr>
<td>Celtics</td>
<td>5</td>
<td>4</td>
<td>105</td>
<td>44</td>
<td>47</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PLAYER | H/V | AST | RB | PTS | FG | CITY | ... |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


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

team ID – home/visiting

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, 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.
Two-step: content planning & realization

- create explicit text plans by aggregating inputs
  - RDF triples → list of trees (one per sentence)
    - joining + ordering (↔)
  - create all possibilities + rank
    - product of experts for given features:
      - individual arrow directions
      - % of reversed
      - sentence split + # of triplets in each
      - relation bigrams (e.g. \(p(\text{capital} | \text{residence})\))
    - 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

(Moryossef et al., 2019)
Realizing from Trees

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

• Input: tree-shaped MRs
  • hierarchy: discourse relation ↓ dialogue act ↓ slot
  • can be automatically induced, much flatter than usual syntactic trees
    • discourse connectives, sentence splits
  • could potentially use other tree-like structures (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

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

(Rao et al., 2019) [https://www.aclweb.org/anthology/W19-8611/](https://www.aclweb.org/anthology/W19-8611/)
(Li et al., 2021) [https://aclanthology.org/2021.inlg-1.10](https://aclanthology.org/2021.inlg-1.10)

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

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

Labs in 10 minutes
Assignment 4

Next week: End-to-end models