EMNLP 2021 Tutorial

Knowledge-Enriched Natural Language Generation

Wenhao Yu¹, Meng Jiang¹, Zhiting Hu², Qingyun Wang³, Heng Ji³, Nazneen Rajani⁴

¹ University of Notre Dame  ² University of California San Diego  
³ University of Illinois at Urbana-Champaign  ⁴ Salesforce Research
General Methods of Knowledge + NLG

**This part**: General principles and methodologies for integrating knowledge into NLG

**Next part** (by Wenhao): Concrete examples and instantiations of the general methods in recent NLG works
General Methods of Knowledge + NLG

This part: General principles and methodologies for integrating knowledge into NLG

Overview:

- Knowledge-enhanced model architectures
  - Attention/copy mechanisms
  - Graph neural models

- Knowledge-enhanced learning
  - Auxiliary loss/tasks
  - Reinforcement learning with knowledge-informed rewards
  - Learning with knowledge constraints

- Knowledge-enhanced inference
  - Steered decoding
  - Prompts
Knowledge-enhanced model architectures

Bake knowledge into the model through specific architectures

Graph NNs for modeling graph-structured knowledge (e.g., knowledge graphs)

Knowledge

Input text

Model

Output text

Attending to & copying from knowledge
Architectures (I): Attention Mechanisms

- Chooses which information to pay attention to

Figure courtesy: Olah & Carter, 2016
Architectures (I): Attention Mechanisms

- Chooses which information to pay attention to

\[ \text{attention_weights of step } i = \text{softmax}(\text{alignment_scores}(\text{decoder_state } i, \text{encoder_states})) \]

Figure courtesy: Olah & Carter, 2016
Architectures (I): Attention Mechanisms

• Chooses which information to pay attention to

\[
\text{context vector of step } i = \text{weighted_sum(attention_weights of step } i, \text{encoder_states)}
\]

\[
\text{attention_weights of step } i = \text{softmax( alignment_scores(decoder_state } i, \text{encoder_states))}
\]

Figure courtesy: Olah & Carter, 2016
Architectures (I): Attention Mechanisms

- Variations of attention mechanisms
  - Different alignment_scores functions

context vector of step \(i\) = weighted_sum(attention_weights of step \(i\), encoder_states)

attention_weights of step \(i\) = softmax( alignment_scores(decoder_state \(i\), encoder_states) )

---

<table>
<thead>
<tr>
<th>Name</th>
<th>Alignment score function</th>
<th>Citation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-base</td>
<td>(\text{score}(s_i, h_j) = \text{cosine}[s_i, h_j])</td>
<td>Graves2014</td>
</tr>
<tr>
<td>Additive(*)</td>
<td>(\text{score}(s_i, h_j) = v^T_a \tanh(W_a[s_i; h_j]))</td>
<td>Bahdanau2015</td>
</tr>
<tr>
<td>Location-Based</td>
<td>(\alpha_{ij} = \text{softmax}(W_a s_i))</td>
<td>Luong2015</td>
</tr>
<tr>
<td>Note: This simplifies the softmax alignment to only depend on the target position.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>General</td>
<td>(\text{score}(s_i, h_j) = s_i^T W_a h_j ) where (W_a) is a trainable weight matrix in the attention layer.</td>
<td>Luong2015</td>
</tr>
<tr>
<td>Dot-Product</td>
<td>(\text{score}(s_i, h_j) = s_i^T h_j)</td>
<td>Luong2015</td>
</tr>
<tr>
<td>Scaled Dot-Product</td>
<td>(\text{score}(s_i, h_j) = \frac{s_i^T h_j}{\sqrt{n}}) Note: very similar to the dot-product attention except for a scaling factor; where (n) is the dimension of the source hidden state.</td>
<td>Vaswani2017</td>
</tr>
</tbody>
</table>

Figure courtesy: Lilian Weng
Architectures (I): Attention Mechanisms

- context vector of step \( i \) = weighted_sum(attention_weights of step \( i \), encoder_states)
  
  attention_weights of step \( i \) = softmax( alignment_scores(decoder_state \( i \), encoder_states) )

- Variations of attention mechanisms
  - Different alignment_scores functions
  - Self attention: Query = Keys = Values
  - Multi-head attention (Transformers)
  - Kernelized attention
  - ...
Architectures (II): Copy/Pointing Mechanisms

• Copy relevant information to the output text

\[ p(y_t) = p_m \cdot p_{gen}(y_t) + (1 - p_m) \cdot p_{copy}(y_t) \]

- Probability of choosing the **generation mode**
- Probability of choosing the **copy mode**

- Probability of generating the token \( y_t \)
- Probability of copying the token \( y_t \) from knowledge / input
Architectures (III): Graph Neural Models

- Representation and reasoning over graph-structured knowledge
- Bridge the gap between graph representation and text generation
Architectures (III): Graph Neural Models

**Main idea:** Pass massages between nodes to refine node (and possibly edge) representations

**Notation:** \( G = (A, X) \)
- Adjacency matrix \( A \in \mathbb{R}^{N \times N} \)
- Feature matrix \( X \in \mathbb{R}^{N \times F} \)

Slide courtesy: Thomas Kipf
Architectures (III): Graph Neural Models

Main idea: Pass massages between nodes to refine node (and possibly edge) representations

Notation: $G = (A, X)$
- Adjacency matrix $A \in \mathbb{R}^{N \times N}$
- Feature matrix $X \in \mathbb{R}^{N \times F}$
Architectures (III): Graph Neural Models

Graph Convolutional Networks (GCNs), Kipf & Welling 2017

Consider this undirected graph:

Calculate update for node in red:

Update rule:

$$h_{i}^{(l+1)} = \sigma \left( h_{i}^{(l)} W_{0}^{(l)} + \sum_{j \in N_{i}} \frac{1}{c_{ij}} h_{j}^{(l)} W_{1}^{(l)} \right)$$

Scalability: subsample messages [Hamilton et al., NIPS 2017]

$N_{i}$: neighbor indices

$c_{ij}$: norm. constant (fixed/trainable)

Slide courtesy: Thomas Kipf
A brief history of graph neural networks

- **“Spatial methods”**
  - Original GNN: Gori et al. (2005)
  - GG-NN: Li et al. (ICLR 2016)
  - MoNet: Monti et al. (CVPR 2017)
  - Neural MP: Gilmer et al. (ICML 2017)
  - GCN: Kipf & Welling (ICLR 2017)
  - Relation Nets: Santoro et al. (ICLR 2018)
  - GraphSAGE: Hamilton et al. (NIPS 2017)
  - Programs as Graphs: Allamanis et al. (ICLR 2018)
  - NRI: Kipf et al. (ICML 2018)
  - GAT: Veličković et al. (ICLR 2018)

- **“Spectral methods”**
  - Spectral Graph CNN: Bruna et al. (ICLR 2015)
  - ChebNet: Defferrard et al. (NIPS 2016)

**Other early work:**
- Duvenaud et al. (NIPS 2015)
- Dai et al. (ICML 2016)
- Niepert et al. (ICML 2016)
- Battaglia et al. (NIPS 2016)
- Atwood & Towsley (NIPS 2016)
- Sukhbaatar et al. (NIPS 2016)

(slide inspired by Alexander Gaunt’s talk on GNNs)
This part: General principles and methodologies for integrating knowledge into NLG

Overview:

• Knowledge-enhanced model architectures
  • Attention/copy mechanisms
  • Graph neural models
• Knowledge-enhanced learning
  • Auxiliary loss/tasks
  • Reinforcement learning with knowledge-informed rewards
  • Learning with knowledge constraints
• Knowledge-enhanced inference
  • Steered decoding
  • Prompts
Knowledge-enhanced learning

- Design knowledge-informed learning problems
  - Auxiliary tasks
  - Reward
  - Constraints

- Model is trained to solve the problems
  - So that knowledge information is absorbed into model parameters

- Often agnostic to model architectures:
  - Thus, can combine the learning methods with any knowledge-enhanced architectures we’ve just seen
Learning (I): Auxiliary tasks

• “Knowledge as target”
  • Create learning targets (labels) based on the knowledge
  • Use the targets to supervise the training of the model
Learning (I): Auxiliary tasks

(1) Combine the **auxiliary tasks** with **standard text generation task**
- Lead to a *multi-task* learning paradigm
- Ex: dialog generation

Dinan et al., “Wizard of Wikipedia: Knowledge-Powered Conversational agents”
Learning (I): Auxiliary tasks

(1) Combine the **auxiliary tasks** with **standard text generation task**
    • Lead to a *multi-task* learning paradigm

(2) The auxiliary tasks provide direct supervision for the text generation task
    • Lead to a *weakly-supervised* learning paradigm
    • Ex: aspect-based summarization

---

**Knowledge sources**

- *ConceptNet*
- *Wikipedia*

---

**Coronavirus daily news**

- **Aspect**: Delta variant
  **Summary**: The highly contagious delta variant is surging ...

- **Aspect**: Vaccination
  **Summary**: The statewide vaccine mandate for ...

- **Aspect**: Travel
  **Summary**: the CDC reinstated its highest travel advisory tier ...

---

Learning (II): Reinforcement learning

- “Knowledge as reward”
  - Knowledge-informed reward function evaluates the quality of generation
  - Model is trained to maximize the reward using reinforcement learning:
    - Policy gradient, (Soft) Q-learning, etc.

- Ex: Learning to generate prompts for topic-controllable generation

Han et al., “Text Generation with Efficient (Soft) Q-Learning”
Learning (III): Learning with knowledge constraints

• “Knowledge as constraints”
  • Impose knowledge-informed constraints on the NLG training objective
  • Model is trained to optimize the objective subject to the constraints

• Methods: posterior regularization, constraint-driven learning, integer linear programming, …
  • Posterior regularization:

\[
\min_{\theta, q, \xi \geq 0} \mathcal{L}(\theta) + \text{KL}(q(y|x) \parallel p_{\theta}(y|x)) + ||\xi||_b
\]

s.t. \( \mathbb{E}_q [f(x,y)] \leq \xi \)  

Minimize KL divergence: encourage model \( p_{\theta} \) to stay close to auxiliary distribution \( q \)

Impose constraints on \( q \)

Solve with an EM-style procedure
Learning (III): Learning with knowledge constraints

• “Knowledge as constraints”
  • Impose knowledge-informed constraints on the NLG training objective
  • Model is trained to optimize the objective subject to the constraints

• Methods: posterior regularization, constraint-driven learning, integer linear programming, ...
  • Posterior regularization
  • Ex:
This part: General principles and methodologies for integrating knowledge into NLG

Overview:

• Knowledge-enhanced model architectures
  • Attention/copy mechanisms
  • Graph neural models

• Knowledge-enhanced learning
  • Auxiliary loss/tasks
  • Reinforcement learning with knowledge-informed rewards
  • Learning with knowledge constraints

• Knowledge-enhanced inference
  • Steered decoding
  • Prompts
Knowledge-enhanced inference

• Integrate knowledge during the text decoding process

• Can be applied to pretrained language models (e.g., GPT-2/3, T5) for knowledge-enhanced NLG
Inference (I): Steered decoding

• Guide the decoding by changing the generation distribution

• TODO: PPLM, GeDi, DeLorean
Inference (II): Prompts

• Guide the decoding by changing the generation distribution

Pretrained LM (e.g., GPT3)

Generate a story about cat: once upon a time, ...

prompt input continuation