Conditional Text Generation

Dr. Nurul Lubis
Dialog Systems and Machine Learning Group
25.06.2021
Today’s generative networks are trained on huge amounts of data
- Generalizability to unseen samples

Example: chit-chat systems
- Trained on data from reddit, twitter, ...

Once deployed, we have no control over the generation

<table>
<thead>
<tr>
<th>Role</th>
<th>response</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Does money buy happiness?</td>
</tr>
<tr>
<td>Bot</td>
<td>Depends how much money you spend on it.</td>
</tr>
<tr>
<td>User</td>
<td>What is the best way to buy happiness?</td>
</tr>
<tr>
<td>Bot</td>
<td>You just have to be a millionaire by your early 20s, then you can be happy.</td>
</tr>
<tr>
<td>User</td>
<td>This is so difficult!</td>
</tr>
<tr>
<td>Bot</td>
<td>You have no idea how hard it is to be a millionaire and happy. There is a reason the rich have a lot of money</td>
</tr>
</tbody>
</table>

DialoGPT (Zhang et al., 2019)
Conditional Text Generation

- Generation with one or more controllable attributes
  - Generation is conditioned on the control attribute
  - Adding adjustable “knobs” in the generation process
Conditional Text Generation

- Generation with one or more controllable attributes
  - Generation is *conditioned* on the control attribute
  - Adding adjustable “knobs” in the generation process
- Gaining popularity, many possible applications
  - Story generation
    - Ending (Peng et al., 2018) and plot (Yao et al., 2019)
  - E-mail
    - Politeness (Madaan et al., 2020)
  - Dialogue
    - Chitchat: Persona (Zhang et al., 2018; Dinan et al., 2016)
    - Task-oriented: dialogue action
- What do you think about the last football match?
  - I didn’t see it! I’m not really into sports
  - I think team A did very well!
  - Persona: dislikes sport
  - Persona: likes sport
Text Generation

- Mainly framed as a language generation task
- Recurrent NN:

What do you think...

Encoder

Decoder
Conditional Text Generation

- Where can we add control attributes in the decoding process?
  - 5 modules in the modification space (Phrabumoye et al., 2020)
    - External input $h_0$
      - Start of decoding or generation
    - Sequential input $x$
      - Control at every time step
    - Generator operation
      - Parameterizing the control attribute
    - Output $o$
      - Projection to vocabulary
    - Loss function $L$
      - Comparison to target

- Diagram:
  - $<\text{start}>$
  - $x_1 \rightarrow h_0$
  - $x_2 \rightarrow h_1$
  - $x_3 \rightarrow h_2$
  - $x_4 \rightarrow h_3$
  - $o_1 \rightarrow \text{I}$
  - $o_2 \rightarrow \text{didn’t}$
  - $o_3 \rightarrow \text{see}$
  - $L$
External input, initialization of decoder state

- Arithmetic or linear transformation
  - When the control attribute comes from another source, it can be combined into $h_0$
    - Linear layer (Hoang et al., 2016)
    - Arithmetic operation (Chandu et al., 2019)
    - Concatenation (Fu et al., 2018; Zhou et al., 2018; Dinan et al., 2018)

- Stochastic changes
  - Variational model such as VAE (Kingma and Welling, 2014)
    - Sample a latent variable to be used to initialize the decoding process (Bowman et al., 2016)

- External feedback
  - Put $h_e$ through a discriminator to detect the control signal
  - Optimize the representation
- Giving control signal at every time step
- Similar to before, the control attribute can be combined with each step of the sequential input with arithmetic or linear transformation
- Numerous works based on RNN in different domains do not show impressive results
  - Word definition: word to be defined is given as input at every time step concatenated with previously generated token (Noraset et al., 2017)
  - Article generation: hidden state of encoder is concatenated at every time step (Phrabumoye et al., 2019)
Generator operation, mapping input to output

- **Recurrent NNs (Rumelhart et al., 1986)**
  - RNN cells contain different gates and operation to better model context
    - LSTM (Hochreiter and Schmidhuber, 1997), GRU (Cho et al., 2014)
  - Similarly, operations can be added to compute the control attribute
  - Factored LSTM (Gan et al., 2017) for image captioning
    - LSTM parameters are factored into three components, each responsible for style, image, and its associated tag.
  - Semantic conditioning (Wen et al., 2015)
    - add a cell in addition to LSTM cell which takes dialogue action to perform sentence planning.

- **Transformers (Vaswani et al., 2017)**
  - Rely on attention mechanisms to draw global dependencies between input and output
  - Hierarchical disentangled attention for graph-based conditioning (Chen et al., 2019)

- **Pre-trained models**
  - Fine-tuning pre-trained models for downstream tasks has shown good results
  - Plug and Play LM (Dathathri et al., 2019) combines pre-trained LM with classifiers to guide the generation process.
  - Changes made should not interfere with pre-trained weights otherwise retraining will be necessary
Output

- Attention
  - Guiding the generation process by attending to the source sequence (Bahdanau et al., 2015)
    - Find a context vector at each time step which captures the information needed from the source
  - Attention for controlled generation
    - Add style control attribute to input sequence (Sudhakar et al., 2019)
    - Use attention to attend over external document as additional source (Dinan et al., 2018)
    - Use the representation of agent persona to compute attention weights (Zhang et al., 2018)
      - Attention weights is recomputed according to the control attribute

- External feedback
  - The output space can be modified with adversarial loss
    - An adversary tries to distinguish sentence whose style has been transferred or sentence from the true target distribution (Shen et al., 2017)
  - Use reward to optimize the output
    - Style reward, semantic reward, and fluency reward (Gong et al., 2019)
  - Arithmetic or linear transform
    - Combine the output with control attribute through a linear layer, addition, or concatenation (Hoang et al., 2016)
      - The transformed output is then used to predict the target token
Loss function

- In addition to typical loss for matching to target, we can add loss to optimize for the control attribute
- KL loss
  - Measure of divergence between two distributions
  - Variational models.
    - If prior for the (latent) control variable is known, we can use KL loss to minimize the distance between the learned posterior distribution and the prior (Lubis et al., 2020)
- Classifier loss
  - Ensure that the generated sequence comply with the control attribute
    - Which style is this sequence? (Hu et al., 2017; Phrabumoye et al., 2018; Sudhakar et al., 2019)
    - Does this sequence belong to style A? (Chandu et al., 2019)
    - Does this sequence have the same style as sequence sampled from a particular style? (Yang et al., 2018)

- Custom task specific loss
Controlled Generation in Dialogue Systems
Semantic action as control

We would like to generate natural language response conditioned on the dialogue action
Semantically-conditioned LSTM (Wen et al., 2015)

- **LSTM cell**
  - input, forget, and output gates
  - Input is current token and previous hidden state

- **Dialogue action (DA) cell below LSTM cell**
  - Input is dialogue action, modifies the dialogue act
    - at each time step $t$ the DA cell *reading gate* decides what information should be retained for future time steps and discards the others
    - \[
    \mathbf{r}_t = \sigma(\mathbf{W}_{wr}\mathbf{w}_t + \alpha\mathbf{W}_{hr}\mathbf{h}_{t-1}) \quad (7)
    \]
    - \[
    \mathbf{d}_t = \mathbf{r}_t \odot \mathbf{d}_{t-1} \quad (8)
    \]

- **Performs sentence planning**
  - Manipulates the LSTM value based on dialogue act features
    - \[
    \mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t + \tanh(\mathbf{W}_{dc}\mathbf{d}_t) \quad (9)
    \]

- **Skip connection, backwards reranking**
Semantically-conditioned LSTM (Wen et al., 2015)

- Tested on restaurant and hotel domains
  - 8 shared system dialogue act
  - 12 slots per domain with some overlap
  - 248 distinct DA for restaurant domain, 164 for hotel
- Objective and subjective evaluations compared to RNN
  - More natural and informative
  - More preferred by humans
Graph-based generation (Chen et al., 2019)

- Multi-domain large scale ontology requires scalability
  - Graph-based representation exploits commonalities between domains, e.g. intents and slots
- Hierarchical disentangled self-attention
  - Different attention heads focus on different nodes in the graph
    - Hierarchical: Layers in the graph
    - Domain -> action -> slot
  - Allows efficient combinations of nodes

Diagram:
- What type and color of taxi do you want to take?
- Disentangled Self-Attention
- Hierarchical Disentangled SA
- Dialog Act Graph
Disentangled self-attention (DSA)

- Transformers concatenate vectors from multiple heads into the final vector.
- While the method to compute attention in each head is identical to transformer, DSA employs binary vector which acts as a switch:
  - Only vectors from “active” heads are considered.
  - The active vectors are summed together.
- Multiple DSA layers are stacked to better handle complexity.
Graph-based generation (Chen et al., 2019)

- MultiWOZ data (Budzianowski et al., 2018)

- Training
  - Dialogue act is used to activate corresponding heads, creating a path inside the graph.
  - The generated response is matched to the target to compute loss for optimization

- Few-shot experiment
  - Graph-act representation helps generalization in few-shot cases
  - HDSA further exploit the hierarchical structure of the dialogue action space
Semantically-conditioned GPT (Peng et al., 2020)

- Leveraging the fluency of large pre-trained models
  - Pre-trained GPT2 Medium, 354M parameters
  - Fine-tune with dialogue NLG task
- Training: Dialogue act followed by system response
- Inference: Dialogue act followed by [BOS] token
Few-shot study on SC-GPT (Peng et al., 2020)

- **FewShotWOZ:**
  - A modified RNNLM (Wen et al., 2015) and MultiWOZ (Budzianowski et al., 2018) corpus with smaller training set, more domains, and smaller overlap between train and test sets.

<table>
<thead>
<tr>
<th>Model</th>
<th>Restaurant BLEU ↑ ERR ↓</th>
<th>Laptop BLEU ↑ ERR ↓</th>
<th>Hotel BLEU ↑ ERR ↓</th>
<th>TV BLEU ↑ ERR ↓</th>
<th>Attraction BLEU ↑ ERR ↓</th>
<th>Train BLEU ↑ ERR ↓</th>
<th>Taxi BLEU ↑ ERR ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC-LSTM</td>
<td>15.90 48.02</td>
<td>21.98 80.48</td>
<td>31.30 31.54</td>
<td>22.39 64.62</td>
<td>7.76 367.12</td>
<td>6.08 189.88</td>
<td>11.61 61.45</td>
</tr>
<tr>
<td>SC-GPT</td>
<td>38.08 3.89</td>
<td>32.73 3.39</td>
<td>38.28 2.75</td>
<td>32.95 3.38</td>
<td>20.69 12.72</td>
<td>17.21 7.74</td>
<td>19.70 3.57</td>
</tr>
</tbody>
</table>

Table 3: Performance of different methods on FewShotWOZ

<table>
<thead>
<tr>
<th>Model</th>
<th>Entity F1</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC-LSTM (Wen et al., 2015b)</td>
<td>80.42</td>
<td>21.6</td>
</tr>
<tr>
<td>HDSA (Chen et al., 2019)</td>
<td>87.30</td>
<td>26.48</td>
</tr>
<tr>
<td>GPT-2</td>
<td>87.70</td>
<td>30.71</td>
</tr>
<tr>
<td>SC-GPT</td>
<td>88.37</td>
<td>30.76</td>
</tr>
</tbody>
</table>

Table 5: Performance on MultiWOZ

<table>
<thead>
<tr>
<th>Model</th>
<th>Seen BLEU ↑ ERR ↓</th>
<th>Unseen BLEU ↑ ERR ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC-LSTM</td>
<td>23.05 40.82</td>
<td>12.83 51.98</td>
</tr>
<tr>
<td>GPT-2</td>
<td>30.43 3.26</td>
<td>27.92 17.36</td>
</tr>
<tr>
<td>SC-GPT</td>
<td>40.28 1.09</td>
<td>36.69 4.96</td>
</tr>
</tbody>
</table>

Table 9: Performance of different methods on seen DAs and unseen DAs in restaurant domain.
Plug and Play Conversational Models (Madotto et al., 2020)

- **Chit-chat system**
  - Trained on large amounts of data
  - Benefits from large pre-trained models
    - DialoGPT (Zhang et al., 2019)

- **Plug and Play Conversational Models**
  - Residual Adapters (Houlsby et al., 2019; Bapna and Firat, 2019) within DialoGPT
    - Added on top of each transformer layer
    - Steers the output distribution without modifying the pre-trained weights
    
    \[
    f_{\theta_i}(x) = \text{ReLU}\left(\text{LN}(x) \cdot W_i^E\right) \cdot W_i^D,
    \]
    \[
    \text{Adapter}(o_{i:t}) = f_{\theta_i}(o_{i:t}) + o_{i:t},
    \]
    - One adapter per control attribute

- **Fine-tuning**
  - Given a set of dialogues with certain attribute \( a \), optimize the parameter of the residual adapters to minimize the NLL over the dataset
  - The parameters of the large pre-trained model is untouched
  - The data used for fine-tuning need not be dialogue data, and can be artificially generated

---

<table>
<thead>
<tr>
<th></th>
<th>( \downarrow ) Ppl.</th>
<th>( \uparrow ) Dist 1/2/3</th>
<th>Discrim.</th>
<th>Score</th>
<th>Posit.</th>
<th>Nega.</th>
<th>Busin.</th>
<th>Sci/Tech</th>
<th>Sport</th>
</tr>
</thead>
<tbody>
<tr>
<td>DG</td>
<td>39.60</td>
<td>0.22/0.64/0.77</td>
<td>46.48</td>
<td>32.91</td>
<td>65.67</td>
<td>19.40</td>
<td>17.41</td>
<td>91.04</td>
<td>27.86</td>
</tr>
<tr>
<td>WD</td>
<td>53.03</td>
<td>0.25/0.74/0.84</td>
<td>50.18</td>
<td>34.54</td>
<td>58.21</td>
<td>28.86</td>
<td>19.40</td>
<td>91.04</td>
<td>36.82</td>
</tr>
<tr>
<td>PP</td>
<td>45.86</td>
<td>0.24/0.67/0.79</td>
<td>73.28</td>
<td>49.54</td>
<td>75.12</td>
<td>51.74</td>
<td>47.26</td>
<td>93.03</td>
<td>59.20</td>
</tr>
<tr>
<td>AD</td>
<td>41.57</td>
<td>0.17/0.58/0.77</td>
<td>96.52</td>
<td>70.01</td>
<td>93.03</td>
<td>73.13</td>
<td>68.66</td>
<td>99.00</td>
<td>83.08</td>
</tr>
</tbody>
</table>

**Score by Attribute**

Table 3: Automatic evaluation results. In all the metrics higher is better except for Perplexity (Ppl.), and Discrim. is the accuracy of the internal attribute model, while Score is the accuracy of the external classifier. All the results, are averaged among the six attribute models.
Conclusion

- Conditional generation is gaining attention
  - Many ways to Rome
    - Conditioning can be done at various places in the generation process
  - Wide range of applications
    - Dialogue systems are natural benefactors

- Generalizability
  - How we represent the control attributes
  - Parameter sharing

- Benefiting from large pre-trained models
  - How to minimize re-training of pre-trained parameters
  - There may be some tradeoff with fluency
    - How to maintain fluency while perturbing the generation
Thank you!