Document Level Hierarchical Transformer

Najam Zaidi (Monash University), Trevor Cohn (University of Melbourne),
Gholamreza Haffari (Monash University)

Abstract

Generating long and coherent text is an important and challenging task encompassing many application areas such as summarization, document level machine translation and story generation. Despite the success in modeling intra-sentence coherence, existing long text generation models (e.g., BART and GPT-3) still struggle to maintain a coherent event sequence throughout the generated text. We conjecture that this is because of the difficulty for the model to revise, replace or revoke any part that has been generated by the model.

In this paper, we present a novel semi-autoregressive document generation model capable of revising and editing the generated text. Building on some recent semi-autoregressive models, we propose document generation as a hierarchical Markov decision process with a two level hierarchy, where the high and low level editing programs generate and refine the document. We train our model using imitation learning and introduce roll-in policy such that each policy learns on the output of applying the previous action. Experiments applying the proposed approach convey various insights on the problems of long text generation using our model. We suggest various remedies such as using distilled dataset, designing better attention mechanisms and using autoregressive models as a low level program.

Problem Formulation

We cast document generation and refinement as a hierarchical Markov decision process (HMDP) with a two level hierarchy. The high level program is defined by the tuple $(\mathscr{D}, \mathscr{A}_{\mathscr{H}}, \mathscr{E}, \mathscr{R}, \mathbf{d_0})$ where a state $\mathbf{d} \in \mathscr{D}$ corresponds to a set of sequences $\mathbf{d} = (\mathbf{s_1}, \mathbf{s_2}, ..., \mathbf{s_L})$ up to length L, and $\mathbf{d_0} \in \mathscr{D}$ is the initial document. The low level program corresponds to the tuple $(\mathscr{S}, \mathscr{A}_{\mathscr{L}}, \mathscr{E}, \mathscr{R}, \mathbf{s_0})$ where a state $\mathbf{s} \in \mathscr{S}$ corresponds to a sequence of tokens $\mathbf{s} = (w_1, w_2, ..., w_n)$ from the vocabulary V up to length n, and $\mathbf{s_0} \in \mathscr{S}$ is the initial sequence.

Policies

High Level Actions work at the sentence level:

- **Reposition**: The reposition policy reads in the document consisting of set of sequences. For every sentence, the reposition policy makes a categorical decision between 0 and L+1 where L is the number of sequences in the document.
- **Insert**: The insertion policy reads the input document consisting of set of sequences and for every possible slot, the insertion policy makes a binary decision which is 1 (insert here) or 0 (do not insert).
- **Update**: The update policy reads the input document, consisting of set of sequences, and for every sequence position, the update policy makes a binary decision which is 1 (update this sentence) or 0 (do not update)

Low Level Actions work at the word level:

- Reposition and Insert: Similar to sentence level but works at word level

Training and Generation

Algorithm 1 Generation in HMDP

Require: Initial document $\mathbf{d_0}$, policy: π_{θ_H}

$\mathbf{d} \leftarrow \mathbf{d_0}$
while Termination condition is not met **do**
  $\mathbf{rep\_index} \leftarrow \text{argmax}_{\mathbf{r}} \sum_{\mathbf{s}_i \in \mathbf{d}} \log \pi^{rep}_{\theta_H}(r_i|\mathbf{s}_i, \mathbf{d})$ ▷ Do reposition
  $\mathbf{d} \leftarrow \mathscr{E}(\mathbf{d}, \mathbf{rep\_index})$
  $\mathbf{ins\_index} \leftarrow \text{argmax}_{\mathbf{p}} \sum_{\mathbf{s}_i, \mathbf{s}_{i+1} \in \mathbf{d}} \log \pi^{ins}_{\theta_H}(p_i|\mathbf{s}_i, \mathbf{s}_{i+1}, \mathbf{d})$ ▷ Do insertion
  $\mathbf{d} \leftarrow \mathscr{E}(\mathbf{d}, \mathbf{ins\_index})$ ▷ Call to Low level MDP
  $\mathbf{upd\_index} \leftarrow \text{argmax}_{\mathbf{u}} \sum_{\mathbf{s}_i \in \mathbf{d}} \log \pi^{upd}_{\theta_H}(u_i|\mathbf{s}_i, \mathbf{d})$ ▷ Do update
  $\mathbf{d} \leftarrow \mathscr{E}(\mathbf{d}, \mathbf{upd\_index})$ ▷ Call to Low level MDP
end while

Algorithm 2 Low Level MDP

Require: Document $\mathbf{d}$, policy: π_{θ_L}, Hi Level MDP action: $\mathbf{H}$

while Termination condition is not met **do**
  $\mathbf{s_0} \leftarrow \text{buildFrame}(\mathbf{d}, \mathbf{H})$
  if $\mathbf{s_0}$ is empty **then**
    $\mathbf{s} \leftarrow \mathbf{s_0}$ ▷ Skip reposition
  else
    $\mathbf{rep\_index} \leftarrow \text{argmax}_{\mathbf{r}} \sum_{w_i \in \mathbf{s}} \log \pi^{rep}_{\theta_L}(r_i|w_i, \mathbf{s}, \mathbf{d})$ ▷ Do reposition
    $\mathbf{d} \leftarrow \mathscr{E}(\mathbf{s}, \mathbf{rep\_index})$
  end if
  $\mathbf{plh\_index} \leftarrow \text{argmax}_{\mathbf{p}} \sum_{w_i, w_{i+1} \in \mathbf{s}} \log \pi^{ins}_{\theta_L}(p_i|w_i, w_{i+1}, \mathbf{s}, \mathbf{d})$ ▷ Insert placeholders
  $\mathbf{s} \leftarrow \mathscr{E}(\mathbf{s}, \mathbf{plh\_index})$
  $\mathbf{tok\_index} \leftarrow \text{argmax}_{\mathbf{t}} \sum_{w_i \in \mathbf{s}, w_i == <mask>} \log \pi^{tok}_{\theta_L}(t_i|w_i, \mathbf{s}, \mathbf{d})$ ▷ Fill placeholders
  $\mathbf{s} \leftarrow \mathscr{E}(\mathbf{s}, \mathbf{tok\_index})$
end while
$\mathbf{d} \leftarrow \text{documentUpdate}(\mathbf{d}, \mathbf{s})$

Algorithm 3 Training for Hierarchical Levenshtein Transformer

Require: Training data $\mathscr{T}$, Model policy: π_θ, Expert policy: π_*

while Maximum training steps reached **do**
  $(\mathbf{d}, \mathbf{d}_*) \sim \mathscr{T}$ ▷ Sample a training pair
  $\mathbf{repH^*, insH^*, updH^*} \leftarrow \pi^H_*(\mathbf{d}, \mathbf{d}_*)$ ▷ Get oracle actions
  $\mathbf{repL1^*, insL1^*, tokL1^*, repL2^*, insL2^*, tokL2^*} \leftarrow \pi^L_*(\mathbf{d}, \mathbf{d}_*)$

  $\mathscr{L}^{rep}_{\theta_H} \leftarrow -\sum_{\mathbf{s} \in \mathbf{d}} \log \pi^{rep}_{\theta_H}(repH^*_i|\mathbf{s}_i, \mathbf{d})$
  $\mathbf{d} \leftarrow \text{applyAction}(\mathbf{d}, \mathbf{repH^*})$

  $\mathscr{L}^{ins}_{\theta_H} \leftarrow -\sum_{\mathbf{s}_i, \mathbf{s}_{i+1} \in \mathbf{d}} \log \pi^{ins}_{\theta_H}(insH^*_i|\mathbf{s}_i, \mathbf{s}_{i+1}, \mathbf{d})$
  $\mathbf{s} \leftarrow \text{buildFrame}(\mathbf{insH^*}, \mathbf{d})$

  $\mathscr{L}^{rep1}_{\theta_L} \leftarrow -\sum_{w_i \in \mathbf{s}} \log \pi^{rep}_{\theta_L}(repL1^*_i|w_i, \mathbf{s}, \mathbf{d})$ ▷ Low Level
  $\mathbf{s} \leftarrow \text{applyAction}(\mathbf{s}, \mathbf{repL1^*})$
  $\mathscr{L}^{ins1}_{\theta_L} \leftarrow -\sum_{w_i, w_{i+1} \in \mathbf{s}} \log \pi^{ins}_{\theta_L}(insL1^*_i|w_i, w_{i+1}, \mathbf{s}, \mathbf{d})$
  $\mathbf{s} \leftarrow \text{applyAction}(\mathbf{s}, \mathbf{insL1^*})$
  $\mathscr{L}^{tok1}_{\theta_L} \leftarrow -\sum_{w_i \in \mathbf{s}, w_i == <mask>} \log \pi^{tok}_{\theta_L}(tokL1^*_i|w_i, \mathbf{s}, \mathbf{d})$
  $\mathbf{d} \leftarrow \text{applyAction}(\mathbf{d}, \mathbf{insH^*})$

  $\mathscr{L}^{upd}_{\theta_H} \leftarrow -\sum_{\mathbf{s} \in \mathbf{d}} \log \pi^{upd}_{\theta_H}(updH^*_i|\mathbf{s}_i, \mathbf{d})$
  $\mathbf{s} \leftarrow \text{buildFrame}(\mathbf{updH^*}, \mathbf{d})$

  $\mathscr{L}^{rep2}_{\theta_L} \leftarrow -\sum_{w_i \in \mathbf{s}} \log \pi^{rep}_{\theta_L}(repL2^*_i|w_i, \mathbf{s}, \mathbf{d})$ ▷ Low Level
  $\mathbf{s} \leftarrow \text{applyAction}(\mathbf{s}, \mathbf{repL2^*})$
  $\mathscr{L}^{ins2}_{\theta_L} \leftarrow -\sum_{w_i, w_{i+1} \in \mathbf{s}} \log \pi^{ins}_{\theta_L}(insL2^*_i|w_i, w_{i+1}, \mathbf{s}, \mathbf{d})$
  $\mathbf{s} \leftarrow \text{applyAction}(\mathbf{s}, \mathbf{insL2^*})$
  $\mathscr{L}^{tok2}_{\theta_L} \leftarrow -\sum_{w_i \in \mathbf{s}, w_i == <mask>} \log \pi^{tok}_{\theta_L}(tokL2^*_i|w_i, \mathbf{s}, \mathbf{d})$

  $\theta \leftarrow \theta - \lambda \nabla [\mathscr{L}^{rep}_{\theta_H} + \mathscr{L}^{ins}_{\theta_H} + \mathscr{L}^{upd}_{\theta_H} + \mathscr{L}^{rep1}_{\theta_L} + \mathscr{L}^{ins1}_{\theta_L} + \mathscr{L}^{tok1}_{\theta_L} + \mathscr{L}^{rep2}_{\theta_L} + \mathscr{L}^{ins2}_{\theta_L} + \mathscr{L}^{tok2}_{\theta_L}]$
end while

Results

	Multi-News			DUC-2004		
	R-1	R-2	R-L	R-1	R-2	R-L
Copy	42.32	13.28	37.86	36.30	8.47	32.52
Transformer	40.62	12.42	36.37	35.4	7.78	31.71
LevT	25.93	8.59	28.95	23.45	4.89	25.12
Editor	25.56	8.13	28.33	23.17	4.21	25.01
Ours	21.67	5.89	24.03	18.22	2.17	20.87

Table 1: Experiment Results on Multi-News and DUC2004 dataset

	Synthetic	ROC-Stories
Copy	23.59	28.82
Transformer	30.17	35.72
LevT	22.42	25.29
Editor	22.78	25.89
Ours	20.63	23.10

Table 2: Experiment Results on Synthetic and ROC-stories dataset. We report the BLEU score in the table.

The main results for summarization are shown in table1. The best result is obtained by copy across both dataset indicating that post editing of long sequences may hurt its quality. Copy consist of output from SummPip system. SummPip uses graph clustering to find relevant sentences which are then used to generate the summary. Among other models, the Vanilla transformer performed better showing a strong bias present in the languages for autoregressive monotone generation. Levenshtein and the Editor transformer performed comparably whereas as our model showed no improvement over the baselines. We see similar performance in Synthetic and ROC-stories dataset in table \ref{tab:results2} with Vanilla transformer performing better then the other models.

Conclusion and future directions

.We present a hierarchical document generation model, that is capable of revising and editing its generated text thus bringing it closer to human-level intelligence. Although results showed that our approach lags behind the baselines, it did shed light into various problems present in semi-
autoregressive models and long document generation. In the future, we will be incorporating

- Distilled Dataset
- Better training regimes
- Use of autoregressive model as low level programmer
- Attention mechanism

Acknowledgements

The Authors would like to acknowledge reviewers for their constructive comments.