SA-HAVE: A Self-Attention based Hierarchical VAEs Network for Abstractive Summarization

Xia Wan and Shenggen Ju
College of Computer Science, Sichuan University, Chengdu, Sichuan, 610065, China
*Corresponding author’s e-mail: wxdwlai@stu.scu.edu.cn

Abstract. The abstractive automatic summarization task is to summarize the main content of the article with short sentences, which is an important research direction in natural language generation. Most abstractive summarization models are based on sequence-to-sequence neural networks. Specifically, they encode input text sequences by Bi-directional Long Short-Term Memory (bi-LSTM), and decode summaries word-by-word by LSTM. However, existing models usually did not consider both the self-attention dependence during the encoding process using bi-LSTM, and deep potential sentence structure information for the decoding process. To tackle these limitations, we propose a Self-Attention based word embedding and Hierarchical Variational AutoEncoders (SA-HVAE) model. The model first introduces self-attention into LSTM to alleviate information decay of encoding, and accomplish summarization with deep structure information inference through hierarchical VAEs. The experimental results on the Gigaword and CNN/Daily Mail datasets validate the superior performance of SA-HVAE, and our model has a significant improvement over the baseline model.

1. Introduction
Text summarization refines integrant information from long texts to a condensed version while preserving its meaning. Summarization can be broadly classified into two categories: extractive and abstractive. Extractive summarizations select complete essential clauses from the original texts and put them together to form a coherent summary, which is more readable but contains redundant information. Abstractive methods rewrite new sentences from scratch without being constrained to reuse phrases from the original texts. Though abstractive models generate more flexible and condensed summaries, facing more challenges.

Abstractive summarization model is built on a sequence-to-sequence (seq2seq) neural network architecture. Chopra et al. [1] used Long Short-Term Memory (LSTM) to generate summaries. It can capture the temporal context information of the sequences. Rush et al. [2] introduced the attention mechanism into the summary task to capture the pivotal part of the text to improve the quality of a summary generated. See et al. [3] proposed the Pointer-Generator Network (PG-Net), which can be viewed as a balance between extractive and abstractive approaches. The above researches are adopted by the LSTM model, and the encoder uses random initialization for word embedding. However, the word embedding lacks the self-attention dependence between input words. Pilault et al. [4] and Song et al. [5] performed a novel Transformer-based [6] model on relevant information before being tasked with generating a summary. Different from LSTM-based models, Transformer can capture the self-attention relationship between input words.

We also note that the structure representation is learned by variational autoencoder (VAE) framework. There are two motivations for using VAE: First, as an unsupervised generative model, VAE has...
better potential modeling capabilities similar to ordinary neural network models; second, it can help learning low-dimensional features and infer more complex features from low-dimensional features. Li et al. [7] introduced a seq2seq framework with a deep recurrent generative decoder to capture historical latent variable dependencies. However, long-term sequential recurrent dependence will lead to the loss of previous information. Zhao et al. [8] presented a variational neural decoder that incorporates latent variables into the LSTM hidden state. However, they used only a single VAE which is insufficient to capture the deeper information of the source sentences.

To overcome these limitations, we propose the self-Attention-based word embedding mechanism and hierarchical Variational AutoEncoder (SA-HAVE) model, in which the word embedding encoder with self-attention learned by the transformer serves as the input of bi-LSTM to alleviate information decay of encoding. And then, we propose a hierarchical VAEs (HVAEs) model with a cascade gated to jointly learn the deep structure information (e.g., syntactic information) from the previous shallow level of the VAE layer and the original context presentation. This model aims to accomplish deep structure information inference and summarization in an end-to-end manner via hierarchical VAEs, and embed the latent syntactic information into the summarization. The main contributions of the paper are summarized as follows.

1) We first introduce a self-attention encoder in LSTM to obtain self-attention weight between input sequences, such that the problem of information attenuation in LSTM can be alleviated.

2) We propose the hierarchical VAEs network to learn the distribution of the deep potential structure information in source sentences and embed the information into the summarization, thereby improving the quality of the decoder in generating summaries.

3) We finally evaluate the performance of our model on Gigaword and CNN/Daily Mail datasets through experiments. The experimental results show that our model has significantly improved the ROUGE scores upon multiple abstractive summarizers.

The rest of the paper is organized as follows. Section 2 is the related work of automatic summarization. Section 3 describes them in detail. Section 4 shows the experimental results and the analysis. Section 5 concludes our paper and provides prospects for future work.

2. Related Work

There are two main different research directions of text summarization tasks. The first direction is the extractive-based model. Extractive summarizations select complete phrases from the original document as the summary. At present, the mainstream extractive summarizations are often solved as sequence labeling tasks or sentence ranking tasks [9, 10]. However, these models often generate results that contain too much redundant information. The second direction is the abstractive method. The difference is that abstractive-based models summarize the document by refactoring new phrases or sentences. The summary generated by abstractive methods is more condensed. In recent years, abstractive summarization is built on the encoder-decoder sequence generation framework [1, 2, 3, 4].

Rush et al. [2] proposed the ABS model and introduced the attention mechanism into the model. The attention mechanism can dispatch different weights to distinguish the key part from the others in the text. It can improve the quality of the summary. Chopra et al. [1] proposed a convolutional attention-based conditional recurrent neural network model based on LSTM for abstractive summarization. Gu [11] and Zeng [12] introduced a coping mechanism in the seq2seq model, which is used to locate a specific segment in the input sentence as a final output. See et al. [3] proposed the Pointer-Generator Network (PG-Net) to solve the problem of out-of-vocabulary (OOV), which can be seemed like a balance between extractive and abstractive approaches. Besides, Pilault et al. [4] performed a simple extractive step, and then used the transformer-based model to generate a summary. Song et al. [5] proposed a method for controlling the amount of verbatim copying. Wang et al. [13] proposed a novel Transformer-based model, which shares attention layers between encoder and decoder to make the most of the contextual clues. Fan et al. [14] proposed a reformulate transformer, which introduces a new layer named dynamic mask attention network (DMAN) to model localness adaptively.
In recent years, as an effective generative model, VAE [15] is often used in sequence generation tasks [7, 8, 16, 17]. Li et al. [7] proposed to incorporate a general sentence structure like “Who Action What” into the model. Zhao et al. [8] presented a variational neural decoder that incorporates only a single latent variable into the LSTM hidden state. Due to the vanishing of KL divergence, the VAE part is too weak to capture deeper structure information.

However, the aforementioned studies either did not consider incorporating the self-attention into LSTM, or they were not enough to learn the deep structure representation. In this paper, we propose the SA-HAVE model, which first adopts a self-attention-based module to present the self-attention of the input long document, and then devises a hierarchical VAEs module to learn the deep structure information.

3. The Proposed Method

3.1 Overview

In this section, we introduce our SA-HVAE model in detail. Figure 1 illustrates an overview framework of our proposed model. The SA-HVAE is a seq2seq neural network. LSTM is used as the basic sequence modeling component in the encoder and decoder. The seq2seq model can compress source text $X = \{x_1, x_2, ..., x_n\}$ into a continuous vector representation with an encoder, and then generate the summary text $Y = \{y_1, y_2, ..., y_m\}$ by a decoder. First, we use the self-attention module to map $X$ into the continuous vector, thereby obtaining the embedding. Then the embedding is feed into the bi-LSTM of the encoder. It encodes the embedding into the content vector of the text. Then, the hierarchical VAEs module uses content vectors to generate potential deep structural information of the input text. Next, we use a generate gate to weight the content vector, the latent variable of the hierarchical VAEs, and the content generated in the previous time step. Finally, the decoder generates the summaries word-by-word according to the final distribution, which is defined as $Y$.

![Figure 1. The Framework of SA-HVAE](image)
time as input at time \( t \), and the test phase takes the output words decoded at the previous time as an input, and the state vector of the decoder is \( s_t \). The calculation formula of the attention distribution in the encoder layer of the model is as follows:

\[
e^i_t = v^T \tanh(W_h h_i + W_s s_t + b), \quad i \in [1, T]
\]

\[
a^t = \text{softmax}(e^t)
\]

\( v, W_h, W_s, \) and \( b \) are learnable parameters. The attention distribution \( a^t \) can be viewed as a probability distribution of the words in the source text, that tells the decoder where to locate to produce the next word. Next, the attention distribution is used to produce a weighted sum of the encoder hidden states, known as the content vector \( c^*_t \):

\[
c^*_t = \sum_i a^t_i \times h_i
\]

After getting the content vector, combined with the state distribution of the decoder \( s_t \), the decoder calculates the word distribution \( P_{vocab} \) in the vocabulary:

\[
P_{vocab} = \text{softmax}(V'[V[s_t, c^*_t] + b] + b')
\]

\( V', V, b, \) and \( b' \) are trainable parameters.

### 3.2 Self-Attention based Encoder

The input is a variable-length sequence \( X = \{x_1, x_2, ..., x_n\} \) to represent the source text. The output is a sequence \( Y = \{y_1, y_2, ..., y_m\} \) which represents the summary. \( X \) is first input to the self-attention word embedding module, which uses a Transformer-like [16] self-attention model to represent the input text as a word embedding vector, as shown in Figure 2. \( Q \) represents the query vector, the key and value are represented by the vectors \( K \) and \( V \). The self-attention method is used here, which is generated by inputting one's embedding, \( X_{\text{embedding}} \), that is, \( Q, K, V \) are generated by linear mapping:

\[
Q = X_{\text{embedding}} \cdot W_Q
\]

\[
K = X_{\text{embedding}} \cdot W_K
\]

\[
V = X_{\text{embedding}} \cdot W_V
\]

\( W_Q, W_K, \) and \( W_V \) are trainable parameters. The calculation of the self-attention score of the text is shown in formula (1):

\[
\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V
\]

\( d_k \) represents the dimension of the vector \( K \), and \( 1/\sqrt{d_k} \) represents the normalization factor of the result. Divide \( Q, K, \) and \( V \) into smaller vectors to obtain multi-head self-attention. Each head focuses on a different part of the source. The final multi-head self-attention is shown in (6):

\[
\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, ..., head_h)W^O
\]

\[
head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)
\]


\[ W^O, W^Q, W^K, \text{ and } W^V \] are learnable parameters. After getting the attention score, it is fed into add & norm sub-layer, which contains two main operations. The first one is the residual connection, i.e., \( X_{\text{embedding}} + \text{MultiHead}(Q, K, V) \). The second is layer normalization, which normalizes the hidden layer in the neural network to a standard normal distribution, thereby accelerating convergence. Denote by \( X_{\text{hidden}} \) the result, where the value of \( X_{\text{hidden}} \) is determined through the ReLU activation function, as the feedforward network is a simple two-layer linear mapping, i.e.,

\[
X_{\text{hidden}} = \text{Relu}(W_1 \cdot W_2 \cdot X_{\text{hidden}}). 
\]

(7)

After all the above operations, we get a matrix that contains not only the self-attention of the input text, but also self-attention between the words of the input \( X \). It then is used as the embedding to input into the bi-LSTM of the encoder.

3.3 Hierarchical VAEs with Cascade Gate

Hierarchical VAEs contain multiple layers and feed forward in adjacent layers. For example, Figure 3 depicts the flow of a two layer hierarchical VAEs. As we have known, the VAE mainly contains two-step generation process, as shown in the left part in Figure 3: first sampling the latent variable, and then reconstructing the structure of the text.

![Figure 3. Flow of hierarchical VAEs](image)

In the sampling stage, the variational encoder maps the content from source text \( X \), observed variable \( y_{<t} \) and hidden structure information \( z_{<t} \) to the posterior probability distribution of the structure variable as \( p_{\theta}(z_t | y_{<t}, z_{<t}, X) \), where \( z_0 \) is defined as the zero vector. It can be seen that this is a recurrent inference process, in which \( z_t \) contains not only the information from the source text but also the historical latent structure of \( z_{<t} \).

We use \( c_t^* \) from the encoder to present the input \( X \), and the state vector \( s_t \) of the decoder to present observed \( Y \) to calculate the sentence structure information at time \( t \), expressed as \( h_t^d \). The most bottom layer of the latent variable is calculated by the formula is as follows:

\[
h_t^d = \sigma(w_h^d \cdot c_t^* + w_s^d \cdot s_{t-1} + w_z^d \cdot z_t^d + b_{\text{ptr}}) 
\]

(8)

\( w_h, w_s, w_z, \) and \( b_{\text{ptr}} \) are parameters that can be learned by the model. \( \sigma(\cdot) \) is the sigmoid function. \( z_0^* \) is zero vector. For the shallow VAE, it only pays attention to the original input, however, the deep layer capture more complicated structured information from the front layer by a cascade gate, the gate decides which part is more important:

\[
h_t^d = \sigma(w_h^d \cdot c_{t-1}^* + w_s^d \cdot s_{t-1} + w_z^d \cdot z_t^d + w_z^d \cdot z_{t-1}^d + b_{\text{ptr}}) 
\]

(9)

After obtaining \( h_t^d \), according to the distribution results of the potential structure information, we further calculate the mean and variance corresponding to the distribution, then use two linear layers to generate the corresponding mean \( \mu_t \) and variance \( \sigma_t^2 \). For the convenience of calculation, we usually use the form of logarithm \( \log(\sigma_t^2) \). The formula is as follows:
\[ \mu_t = W_{h\mu}^d h_t^d + b_{h\mu}^d \]
\[ \log(\sigma_z^2) = W_{h\sigma}^d h_t^d + b_{h\sigma}^d \]

Equations 10 and 11 define the parameters \( W_{h\mu}, b_{h\mu}, W_{h\sigma}, \) and \( b_{h\sigma} \) of the VAE model. These learnable parameters are used to calculate the latent variable \( z_t^d \) as follows:
\[ z_t^d = \mu_t + \sigma_t \otimes \epsilon_t \sim \mathcal{N}(0,1) \]

Equation 12 shows the resampling technique used in the VAE model to generate each new word in the abstract. The probability of generating each word is defined as \( p_\theta(y_t | z_t) \). The goal is to maximize this probability when generating each word.

In the generation phase, the latent variable is fed into the generation module to generate the word sequence. The probability is defined using the model parameters \( \phi \) and \( \theta \). The goal is to maximize the probability of generating each word in the abstract.

Equation 13 shows the generation process in the variational decoder as:
\[ p_\theta(Y) = \prod_{t=1}^{T} p_\theta(y_t | z_t) p_\theta(z_t) dz_t \]

The conditional probability distribution \( p_\theta(y_t | z_t) \) is unknown and difficult to predict directly. Therefore, we assume a probability function with a known distribution to simulate the true posterior distribution.

In the above formula, the conditional probability distribution \( p_\theta(y_t | z_t) \) is unknown and difficult to predict directly. Therefore, we assume a probability function with a known distribution to simulate the true posterior distribution.

Equation 14 shows the calculation of the KL divergence.

Where \( \phi \) is the model used to fit the true probability distribution \( p_\theta \). The loss function of the VAE module is as follows:
\[ l_{vae} = E_{q_\phi(y_t | y_{<t}, z_{<t})} \left\{ \sum_{t=1}^{T} \log p_\theta(y_t | z_t) - D_{KL}[q_\phi(z_t | y_{<t}, z_{<t}) || p_\theta(z_t)] \right\} \]

### 3.4 Decoder with Hierarchical VAEs

After the decoder obtains the initialization vector representation from the output of the encoder, the words combined from the summary at the previous moment act as input at time \( t \). Then, the output words generated by decoding at the previous moment act as input in the test phase. Finally, the decoder generates the corresponding hidden state vector \( s_t \) and content vector one by one. The attention distribution \( a_t \) can be viewed as a probability distribution of the words in the source text that tells the decoder where to locate to produce the next word. Next, the attention distribution is used to produce a weighted sum of the encoder hidden states, known as the content vector \( c_t^* \).

The content vector is not only used to measure the main content of the input sequence, but also an important guiding factor for the probability distribution of the final generated vocabulary by the decoder. At present, the calculation process of the content vector does not consider the potential sentence structure information of the input sequence. To tackle the problem, this article uses a pointer to the original text in the decoder by drawing on the idea of pointer-generator network. And we use the deep structural information to assist in calculating the distribution of original words. The generate gate, denoted as \( P_{gen} \), appends these offline words to the end of the vocabulary. \( P_{gen} \) is calculated as follows:
\[ P_{gen} = \sigma(w_{h1}^T c_t^* + w_{h2}^T s_t + w_{h3}^T z_t + b_{ptr}) \]

And the final probability of generating a word is:
The loss function of the model consists of two parts: one part is the negative number of the log-likelihood estimate, which represents the error of the true result of the generated result, and the other part is the variational lower bound, which is measured by ELBO. Finally, the loss function of the model is expressed as:

$$\mathcal{L} = \frac{1}{D} \sum_{d=1}^{D} \sum_{t=1}^{T} \left\{ -\log p(y_t^{(d)} | y_{<t}^{(d)}, x) + D_{KL}[q_\phi(z_t^{(d)} | y_{<t}^{(d)}, z_{<t}^{(d)}) || p_\psi(z_t^{(d)})] \right\} \quad (18)$$

4. Experiments

4.1 Datasets and Evaluation

The model in this paper experiments on the Gigaword and CMM/DM both of them are English news dataset. The Gigaword dataset consists of the first sentence of the article and the title combined with heuristic rules. And CNN/DM is an online news text obtained from the news interface of CNN and Dailymail websites.

The model uses the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) method proposed by Lin et al. [18] for performance evaluation, which is a recall-oriented generative model evaluation index. In the experimental results, the F-score of ROUGE-1 (R-1), ROUGE-2 (R-2), ROUGE-L (R-L) is used to evaluate our proposed model and baseline models.

4.2 Implementation Details

Our experiments are based on PyTorch and done on an NVIDIA 2080Ti GPU. The batch size set during model training is 32. As for optimizer, we utilize Adam optimizer [19] with learning rate 1e-4 to update parameters in our model. The vocabulary size is set to 50k for both CNN/DM and Gigaword. And the hierarchical VAEs contains 3 layers, which achieves the optional results by ablation studies in section 4.5.

4.3 Overall Result

We conducted experiments on Gigaword and CNN/DM respectively. From the table, we can figure out that SA-HVAE achieves better results as compared to other baselines. As described in Section 3, the SA-HAVE we propose in this paper, which can obtain the self-attention between words in the input texts, and the deep sentence structure information. Therefore the context of the source document can be understood as much as possible, and it’s able to improve the accuracy of summary results. Specially, the experimental results of Gigaword are shown in Table 1, and the experimental results of CNN/DM are shown in Table 2.

As shown in Table 1, the results on the Gigaword dataset our model exceeds 2.44 and 0.52 on ROUGE-1 and ROUGR-L respectively, show that SA-HAVE achieves significant improvements over DRGD [7], with 5.36, 2.18, and 3.59 on ROUGE-1/2/L higher. Compared to the prior state-of-the-art model, i.e., Transformer+Rep [20], our model can still be better.

| Models                        | R-1  | R-2  | R-L  |
|-------------------------------|------|------|------|
| ABS+ [2]                      | 29.80| 11.90| 27.00|
| PG [3]                        | 34.19| 16.92| 31.81|
| DRGD [7]                      | 36.27| 17.57| 33.62|
| Mask Attention Network        | 38.28| 19.46| 35.46|
| [14]                          | 39.00| 19.40| 35.30|
As we can see the results in Table 2, compared with PG [3], our model improves ROUGE-1/2/L by 3.22, 2.02, and 2.95 respectively. This shows the effectiveness of the SA-HVAE model. We believe that self-attention can reduce the information attenuation for the long input document, and the hierarchical VAEs are beneficial to decoding processing.

Table 2. ROUGE F1 scores on the test set of CNN/DM

| Models                      | R-1 | R-2 | R-L |
|-----------------------------|-----|-----|-----|
| PG [3]                      | 39.53 | 17.28 | 36.38 |
| PG+SA [21]                  | 40.02 | 17.88 | 36.71 |
| Subformer-base [22]         | 40.90 | 18.30 | 37.70 |
| Mask Attention Network [14] | 40.98 | 18.29 | 37.88 |
| Selector & Pointer-Genera-  | 41.72 | 18.74 | 38.79 |
| tor [23]                    |     |     |     |
| SA-HVAE                     | 42.75 | 19.30 | 39.33 |

4.4 Ablation studies

To explore the influence of key components in our model, we conducted a series of ablation experiments on CNN/DM. The results of the ablation experiments are illustrated in Table 3.

Table 3. Results of ablation experiments

| Model                      | R-1 | R-2 | R-L |
|----------------------------|-----|-----|-----|
| SA-HVAE                    | 42.75 | 19.30 | 39.33 |
| - HVAEs                    | 40.29 | 17.80 | 36.97 |
| - SA-Embedding             | 39.87 | 17.43 | 36.44 |

From the ablation study results, we have the following observations:

Firstly, reducing the hierarchical VAEs component, the R-1 values of CNN/DM lost 2.46. And then, after removing the self-attention word embedding module, they lost 0.42. These show that reducing these two modules will lose the accuracy of generation.

4.5 Results analysis

Figure 4 shows the effect of different VAE layers on the experimental results, the multiple layers of the hierarchical VAEs achieve better results than the single-layer VAE. As mentioned before, hierarchical VAEs capture deeper information. And when setting the layers as 3 of the VAEs, we get the best performance. Hierarchical VAEs can improve the ability of variational derivation, and the information of VAE is used to enhance the generation ability of the decoder. However, when the hierarchical VAEs have too many layers, there is no distinction in terms of more complicated information among the sentences of the input documents.

Figure 4. Results on CNN/DM of different layers VAE
5. Conclusion
Abstractive summarization is an important research topic in natural language generation. It needs to compress and reconstruct new sentences after understanding the source document. Existing studies neglected the correlation between the words in the source document and the deep potential sentence structure information which causes biased results. Therefore, we proposed the SA-HAVE model, where the self-attention based word embedding module is applied to learn the correlation between words of the input article, and the hierarchical VAEs are applied to generate corresponding deep latent sentence structure information according to the source text. The experimental results proved the significant effect of our model with respect to both Gigaword and CNN/Daily Mail datasets. In the future, we will focus on improving the performance of self-attention in our model, which is imperative for encoding the document.

References
[1] Sumit Chopra, Michael Auli, and Alexander M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. In Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 93–98 (2016).
[2] Rush, Alexander M., Sumit Chopra, and Jason Weston. "A neural attention model for abstractive sentence summarization." arXiv preprint arXiv:1509.00685 (2015).
[3] See, Abigail, Peter J. Liu, and Christopher D. Manning. "Get to the point: Summarization with pointer-generator networks." arXiv preprint arXiv:1704.04368 (2017).
[4] Pilault, Jonathan, et al. "On Extractive and Abstractive Neural Document Summarization with Transformer Language Models." Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2020.
[5] Song, Kaiqiang, et al. "Controlling the amount of verbatim copying in abstractive summarization." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. No. 05. 2020.
[6] Vaswani, Ashish, et al. "Attention is all you need." arXiv preprint arXiv:1706.03762 (2017).
[7] Li, Piji, et al. "Deep recurrent generative decoder for abstractive text summarization." arXiv preprint arXiv:1708.00625 (2017).
[8] Zhao, Huan, et al. "Variational neural decoder for abstractive text summarization." Computer Science and Information Systems 00 (2020): 12-12.
[9] Padmakumar, Aishwarya, and Akanksha Saran. "Unsupervised text summarization using sentence embeddings." Dept. Comput. Sci., Univ. Texas, Austin, USA, Tech. Rep. (2016).
[10] Nallapati, Ramesh, Feifei Zhai, and Bowen Zhou. "Summarunner: A recurrent neural network based sequence model for extractive summarization of documents." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 31. No. 1. 2017.
[11] Gu, Jiatao, et al. "Incorporating copying mechanism in sequence-to-sequence learning." arXiv preprint arXiv:1603.06393 (2016).
[12] Zeng, Wenyuan, et al. "Efficient summarization with read-again and copy mechanism." arXiv preprint arXiv:1611.03382 (2016).
[13] Wang, Tianming, and Xiaojun Wan. "T-CVAE: Transformer-Based Conditioned Variational Autoencoder for Story Completion." IJCAI. 2019.
[14] Fan, Zhihao, et al. "Mask Attention Networks: Rethinking and Strengthen Transformer." arXiv preprint arXiv:2103.13597 (2021).
[15] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013).
[16] Bowman, Samuel R., et al. "Generating sentences from a continuous space." arXiv preprint arXiv:1511.06349 (2015).
[17] Wang, Wenlin, et al. "Topic-guided variational autoencoders for text generation." arXiv preprint arXiv:1903.07137 (2019).
[18] Lin, Chin-Yew. "Rouge: A package for automatic evaluation of summaries." Text summarization branches out. 2004.
[19] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).
[20] Takase, Sho, and Shun Kiyono. "Rethinking Perturbations in Encoder-Decoders for Fast Training." arXiv preprint arXiv:2104.01853 (2021).
[21] Chowdhury, Tanya, Sachin Kumar, and Tanmoy Chakraborty. "Neural abstractive summarization with structural attention." arXiv preprint arXiv:2004.09739 (2020).
[22] Reid, Machel, Edison Marrese-Taylor, and Yutaka Matsuo. "Subformer: Exploring Weight Sharing for Parameter Efficiency in Generative Transformers." arXiv preprint arXiv:2101.00234 (2021).
[23] Cho, Jaemin, Minjoon Seo, and Hannaneh Hajishirzi. "Mixture content selection for diverse sequence generation." arXiv preprint arXiv:1909.01953 (2019).