Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 189 (2021) 167–174

Procedia Computer Science

www.elsevier.com/locate/procedia

5th International Conference on AI in Computational Linguistics

IMPROVING ABSTRACTIVE SUMMARIZATION WITH SEGMENT-AUGMENTED AND POSITION-AWARENESS

Minh-Phuc Nguyen[a], Nhi-Thao Tran[a]

[a] *Faculty of Information and Technology, Vietnam National University, University of Science, Ho Chi Minh City, Vietnam*

Abstract

Detecting the relationship between phrases to then derive the salient information is always an art of abstractive summarizing text. In this work, we present an improved version of the extractor-abstractor system called SEGMENT, in which the extractor identifies words and phrases included in the target summary and the abstractor leverages these features into generating a fluent summary. We provide a segment embedding layer to enrich information for the abstractor that increases the cohesion among phrases. In the extractor, we combined the filtering mechanism and the position awareness to improve the quality of information selectivity. Our method demonstrates potential improvements in CNN/DM dataset and exceeds state-of-the-art 5.1% in ROUGE-1 and 5% in ROUGE-2.

Keywords: abstractive summarization; segment; recurrent network;

1. Introduction

Text summarization is about shortening a long document while retaining concise, readability, and crucial information. The two main principles for the text summarization systems are *extractive* and *abstractive* summarization. Extractive summarization aims to select subsets of words or sentences which represent a summary of the document. Besides, abstractive summarizing focuses on generating a summary that may contain new phrases not appearing in the source text.

Encoder-decoder[1][6][17] is widely used for many sequence generation tasks including text summarization. The encoder aims to compress the source input before feeding it into the decoder to generate a summary. However, this framework can not manage particular words in the source input that the model has never seen before which leads to out-of-vocabulary (OOV). This framework is then improved by a copying mechanism which was first introduced

[a] Corresponding author.
E-mail address: npminh@apcs.vn (M.P. Nguyen) , tttnhi@mso.hcmus.edu.vn (N.T. Tran)

1877-0509
10.1016/j.procs.2021.05.081

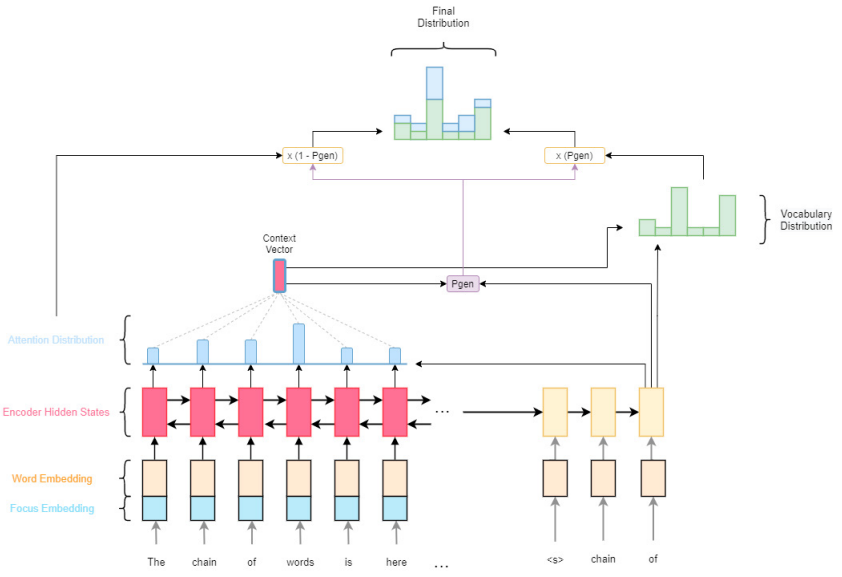


Fig. 1. Pointer Generator architecture [16] with additional focus embedding [5]

by Vinyals et al. [20] to address OOV, and it is called the Pointer network. Following previous works, See et al.[16] refined the pointer network with coverage mechanism [18] that helps model prevent repeated phrases by informing their history attention. Nallapati et al. [13] presented a joint extractive-abstractive model that can compress important sentences from source text then use these sentences to train the abstractive system.

In 2018, Chen et al. [4] introduced a system that selecting important phrases first, then uses encoder-decoder models to paraphrase them. Li et al. [10] guided the abstractive system with compressed representations of keyword extractors. To constraining copying words, German et al.[7] proposed a separated content selector that produces a binary mask to verify words are in the target source. Moreover, Cho et al. [5] inherited this content selector and presented a focus representation to feed these masks into encoder-decoder models.

However, the standard content selector extracts words that are in the target without any contextual and positional information. Due to this problem, a large number of misunderstood words by the wrong context might be selected. To address this problem, we propose a SEGMENT, the simple and effective framework that increases the cohesion among phrases. This helps the extractor aware of phrases in the target source while maintaining a concentrate on single words. We also modify the extractor architecture with the positional context that enriches the area features and a Temporal Convolution Network[2](TCN) layer to softly filter encoding features. Our SEGMENT overview architecture is illustrated in Figure 3.

2. Related Work

The challenges in summarization are identifying and selecting the salient information before producing a comprehensive output. Therefore, the **SELECTOR** aims to select important tokens while the **Pointer Generator** generates the summary.

Pointer Generator Network(PG) is published by See et al. [16] to overcome the OOV problem. The neural-network based approach has a fixed vocabulary that is built from the most popular words. This leads to some particular words that appear in the input source but do not include in the fixed vocabulary would not be considered to generate. Therefore, an extended vocabulary is produced for each source text by examining words in the input. In the decoding step, a generation probability aims to decide to copy words from the source text or choose a word in the vocabulary. The pointer generator idea is described in Figure 1.

Given sequence of tokens in the input source document $x = [x_1, ..., x_N]$ and a decoder then generates the target summary $y = [y_1, ..., y_M]$ modeling the conditional probability $p(y_1, ..., y_M|x_1, ..., x_N)$ where N is the input length and M is the summary length. PG takes the attention output which is attention score e^t to produce attention distribution a^t before constructs context vector h_t^* at each time step t. Attention score e^t computation is illustrated in Equation 1 where v, W_h, W_s, b_{attn} are learnable parameters. The context vector h_t^* is formed by attention distribution a^t and encode hidden states h, which is showed in Equation 3.

$$e_i^t = v^T tanh(W_h h_i + W_s s_t + b_{attn}) \qquad i = 1; 2...; N \tag{1}$$

$$a^t = softmax(e^t) \tag{2}$$

$$h_t^* = \sum_i a_i^t h_i \tag{3}$$

In the next step, context vectors h_t^* and the current state s_t are used to calculate vocabulary distribution P_{vocab} where V and V' are learnable matrix . Beside, the generation probability $P_{gen} \in [0; 1]$ at time step t provided in Equation 5 is calculated from context vectors h_t^*, decoder state s_t, decoder input x_t with learnable parameters w_h, w_s, and w_x.

$$P_{vocab} = softmax(V'(V[s_t, h_t^*])) \tag{4}$$

$$P_{gen} = \sigma(w_h^T h_t^* + w_s^T s_t + w_x^T x_t) \tag{5}$$

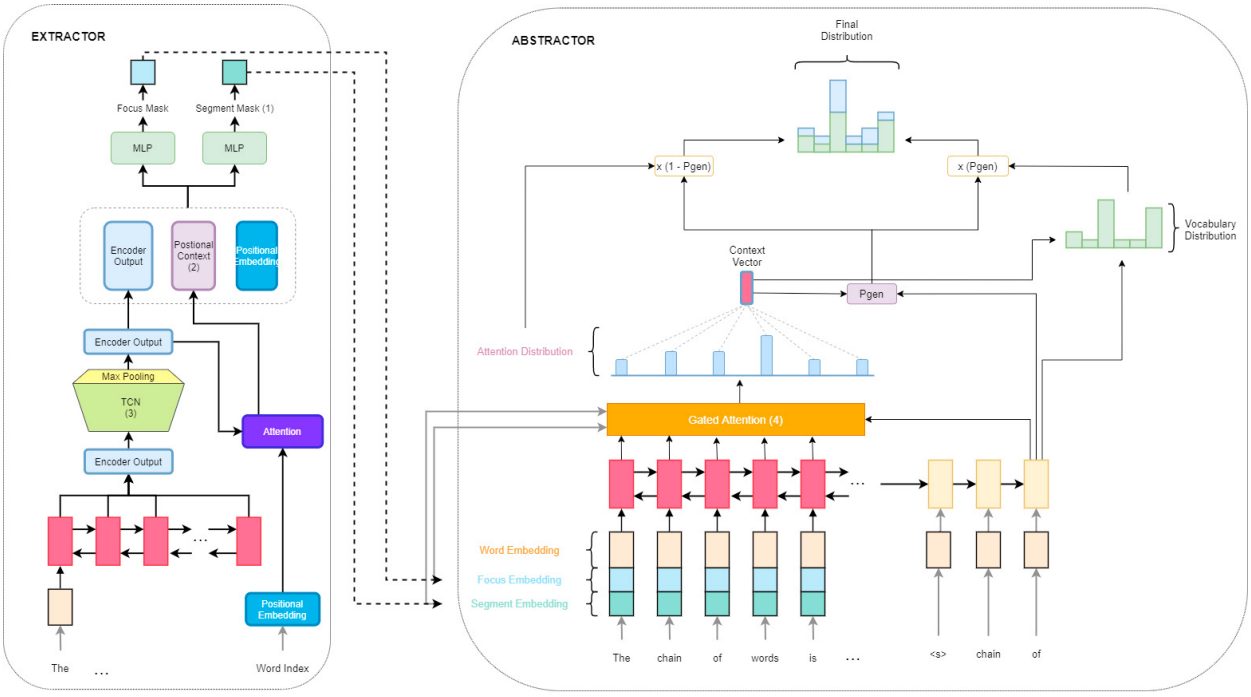
Final distribution $P(w)$ leverages P_{gen} as a soft switch to choose between generating a word from the vocabulary by sampling from P_{vocab}, or copying a word from the input sequence by sampling from the attention distribution a^t.

$$P(w) = P_{gen} * P_{vocab}(w) + (1 - P_{gen}) \sum_{i:w_i=w} a_i^t \tag{6}$$

SELECTOR is a framework proposed by Cho et al. [5] with two separate modules which are selector and generator. The selector is a bidirectional Gated Recurrent Unit [6] (Bi-GRU) to produce focus mask $m = [m_1, ..., m_N]$ which is defined as words in target source while the generator is PG network. To attend these focus masks at the decoding step, Cho et al. provided them an embedding space and concatenated them with the input embedding of the source sequence which is blue blocks in Figure 1.

Our work does not use expert numbers to generate multiple inputs like Cho et al.[5] because we focus on each input has multiple outputs. This leads to our extractor architecture being different from the author which is described in detail in Section 3.1. First, the input is fed into the extractor to produce focus logit o_t^f. Then the focus logit o_t^f would be transformed into the binary mask m_t as Equation 7 where ϵ_m is focus mask threshold. Those focus masks are fed to the PG as additional information. The PG with additional focus embedding is our base model which is described in Figure 1.

$$p(m_t|x) = \begin{cases} 1 & o_t^f > \epsilon_m \\ 0 & ow \end{cases} \tag{7}$$



3. Method

3.1. Extractor

We provide a multi-task learner Extractor that outputs two binary masks which are focus masks and segment masks responsible for words in the target source and words in phrases included in the summary. This enables our framework to capture all the words that are on target and increase the cohesion among phrases. Moreover, we take the advantage of position awareness by adding positional embedding and computing positional context.

3.1.1. Positional Context

Adding anchor points while combining two separated features space has been applied in image captioning [11]. Motivated by that work, we provide the positional context h^p which is a joint feature between encoder outputs and positional embedding $z^p = [z_1^p, ..., z_N^p]$ following Equation 10. The positional context is the summation of attention distribution and encoder outputs. Attention distribution e^{ex} is computed by a transformed attention score between Extractor's encoder outputs h^{ex} and positional embedding z^p as Equation 8 where $W_{h^{ex}}, W_p$ are learnable matrices.

$$e_i^{ex^t} = v^T tanh(W_{h^{ex}} h_i^{ex} + W_p z_i^p + b_{pos-attn}) \qquad i = 1; 2...; N \tag{8}$$

$$a^t = softmax(e^{ex^t}) \tag{9}$$

$$h_t^p = \sum_i a_i^t h_i \tag{10}$$

We hypothesis that with a particular positional context, the model can be affected differently. The model with word order information would "understand" the context more specifically. While given the relation among sentences order, we believe that the model can learn the relations among sentences to generate a fluent summary. In terms of the word's position awareness, we hypothesis that a learned representation would give better performance than an absolute embedding. Regarding absolute positional embedding, we believe a constant embedding would give the

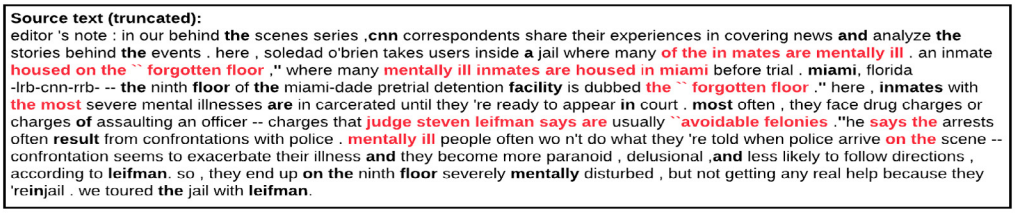


Fig. 2. Example of segment mask. The bold texts show the words included in the target source. The red highlights present the segment mask.

model the most explicit information about the position to make a clear context. We use the sinusoidal function same as positional encoding in Transformer [19] for absolute embedding. To produce sentence order, we provide a sentence indexing for each word based on the sentence order in the source text. Concerning no position awareness, this model would not use positional context.

3.1.2. Segment

Because the focus mask can select lots of individual words without considering the context and this creates noise information for the model. Therefore, a segment plays an essential role in filtering out the individual words and compacting the segment information to yield more salient information. Segment are phrases that are included in the target source which are red highlights in Figure 2. Based on grammar, words in segments usually stand not in the individual but in a chain which makes the sentence smoothly and grammatically. Moreover, the target source is compressed of striking information which are noun phrases and verb phrases, so the segment inherits this property to produce salient points of the given text. We therefore select all the words that are in phrases as long as appearing in the target source. In Figure 3, segment mask is described at the top of the Extractor module. The focus segment logit is calculated from the current hidden state h_t^{ex}, first and last hidden state h_1^{ex}, h_N^{ex}, positional context h^p and positional embedding z^p as Equation 12 where FC^s are fully-connected layers. During training, we treat segment masks $z^s = [z_1^s, ..., z_N^s]$ as the same as focus masks but use different Multi-layer Perceptron(MLP) weights to generate the focus segment logit o^s.

$$(h_1^{ex}...h_N^{ex}) = BiGRU(x) \tag{11}$$

$$o_t^s = FC^s([h_t^{ex}; h_1^{ex}; h_N^{ex}; h^p, z^p]) \tag{12}$$

To transform into a segmented mask, we reuse Equation 7 with a different constant threshold. Then these masks are embedded as segment embedding and concatenated with focus embedding and word embedding to feed into Abstractor as Figure 2. In terms of making the segment label, we first generate the focus mask which is a binary vector with the length of N that represents words inside the target summary. Given the binary focus mask that has been extracted from the source text, we generate a binary focus segment mask following a for loop along N in three strategies: the first word, the last word and (two special cases), the general case. The word is marked 1 if it is inside the segment, otherwise is 0. For two special cases, we guarantee that the word next to it has the same label and vice versa. For the general case, we simply ensure the at least one neighbor has the same label as the current word in the for loop.

3.1.3. Temporal Convolutional Network(TCN)

Our Extractor improves upon TCN [2] by adding a max-pooling layer to select significant features. TCN is residual connection blocks where each block contains residual dilated causal convolutions [14]. Due to the dilated convolutions [21], the model can widen the kernel size and handle sequence input parallel. Moreover, by combining dilated convolutions and residual connection, TCN enables to control of the model's memory size and stable gradients. Therefore, we hypothesis that TCN can help our model extracts significant features along large kernel size which is similar to select prominent features of phrases. According to the Figure 3, TCN is the green block that is connected with

Max-pooling layer in Extractor module. We decide to leverage the power of TCN to filter out unnecessary features of the encoder outputs. Extractor's encoder outputs e^{ex} are feed into the TCN layers to expanse the feature size. Then the expanding features are filtered out by the max-pooling layer to select salient features.

3.2. Abstractor

3.2.1. Gated Attention

Instead of using the Extractor's output at inference time only [7] which is a hard binary mask to constrain copying words, we want to leverage its information to make the model more flexible while generating summaries. We hypothesis that with the help of focus information, the model can avoid a significant amount of unnecessary attention on unattended elements, and allows the model to have more concentration on important parts of the sequence. Such as when the model knows the lasted predicted word is in both the input source and the segment mask, it can enable to have higher attention with the neighbor's word of the predicted one in the input source. The main idea is to utilize focus and segment representation to re-ranking the attention score at the decoding step.

At each time step t, we compute a gated mask g^t then element-wise multiply it with attention score e^t. Gated mask at time step t is first computed by the concatenation of attention distribution e^t , focus embedding z^f and segment embedding z^s as Equation 13 where W^g is learnable matrix. Then this multiplication is fed into layer normalization before it becomes a gated masking sampling by a sigmoid function.

$$g^t = \sigma(LayerNorm(W_g[e^t, z^f, z^s])) \tag{13}$$

$$a^t = softmax(e_t \odot g^t) \tag{14}$$

The gated attention is illustrated in Figure 3 as an orange block in Abstractor module.

4. Experiments

4.1. Dataset

We use CNN/DailyMail dataset [8][13] which contains online news articles. This dataset is divided into the following configuration: 287,113 train samples, 13,368 validation samples, and 11,490 test samples respectively. On average, there are approximately 28 sentences per document in the training set and 3-4 sentences in the reference summaries. The average word count per document of the training set is 802. Our work experiments in non-anonymized version as same as See et al[16].

4.2. Implementation Details

In all experiments, we followed the configuration by See et al. [16]. We set the maximum input length to 400 tokens, maximum of 100 tokens for the output summary for training and 120 tokens at test time, and 50,000 for the vocabulary size. We used Adam [9] optimizer and set 256 dimensions for hidden states in GRU for both Extractor and Abstractor. For Extractor, we set 0.15 for focus segment threshold and 0.1 for segment threshold. For TCN, we used Bai et al.[2] implementation and set 3 for kernel size, [256,512] for dimensions of 2 channel numbers. For Abstractor, we tied the weights [15] of the word embedding in both encoder and decoder, and the decoder output layers. Moreover, Extractor and Abstractor are trained end-to-end similar to Chen et al.[4] without freezing.

4.3. Metric

We evaluated our models by ROUGE score [12]. ROUGE calculates the similarity of two paragraphs based on the word-overlapping, bigram-overlapping, and the longest common subsequence, which is presented as ROUGE-1, ROUGE-2, ROUGE-L points, respectively. The higher the ROUGE points are, the closer between reference and predicted summary.

4.4. Result

Table 1. Results of abstractive summarizers on the CNN/DM dataset. The first section shows state-of-the-art abstractive summarization methods on CNN/DM. The second section shows our base and described methods in this work. R stands for ROUGE. R-AVG is the average score of R1, R2, and RL.

Model	R1	R2	RL	R-AVG
PG	39.53	17.28	36.38	31.06
Bottom-Up	41.22	18.68	38.34	32.75
SELECTOR	41.72	18.74	38.79	33.08
DCA [3]	41.69	**19.47**	37.92	33.11
SEGMENT	42.10	19.24	38.80	33.38
SEGMENT + gated attn	**42.20**	19.11	38.70	33.33
SEGMENT + TCN	**42.20**	19.25	**38.86**	**33.43**
SEGMENT + gated attn + TCN	42.08	19.31	**38.87**	33.42

Our results are described in Table 1. All the SEGMENT variations show consistent improvement results in ROUGE-1 and ROUGE-2 due to the support of additional features from the segment mask. We observe that our SEGMENT shows the competitive result by the increase in ROUGE-1 and ROUGE-L are +0.38 and +0.50 respectively. However, SEGMENT variations have no significant difference in ROUGE-L compared to baseline. The model still outputs the summary whose arrangement order of phrases is still incorrect because of the lack of document-level context.

Among these variations, SEGMENT with TCN has the highest result in ROUGE-1, ROUGE-2 and this model also outperforms the state-of-the-art in R-average. With the help of TCN, SEGMENT can filter out unnecessary parts and focus on needed components that increase ROUGE-1. SEGMENT with TCN outperforms the MLE state-of-the-art model by +0.48 ROUGE-1, +0.51 ROUGE-2, and +0.35 ROUGE-AVG respectively. Compared to reinforcement-learning based state-of-the-art, SEGMENT with TCN also beats that model in ROUGE-1, ROUGE-L, and ROUGE-AVERAGE. While our model result is less than DCA in ROUGE-2, but the consistent increase hints that the fluency of the predicted summary is specifically better.

In terms of gated attention, the SEGMENT with gated mechanism shows the tradeoff in ROUGE score while ROUGE-1 is marginally increasing but ROUGE-2 and ROUGE-L are slightly decreasing. SEGMENT with gated attention and TCN achieves the best ROUGE-2 by +0.57 increasing score but the ROUGE-1 moderately decreased. Although the SEGMENT with gated attention variations show the improvement result which means this approach can potentially benefit the model, there is a slight decrease in other scores that leads to making this combination incomprehensive. This tradeoff is sufficient to demonstrate that it still a challenge to apply flexible masking during training to produce a consistent result.

Table 2. Experimental results of SEGMENT with different positional embedding on the CNN/DM dataset.

Model	R1	R2	RL
SEGMENT w/o POS	41.95	**19.24**	38.69
SEGMENT	**42.10**	**19.24**	**38.80**
SEGMENT with SENT POS	41.85	19.10	38.63
SEGMENT with ABSOLUTE POS	41.92	19.15	38.61

We conducted some experiments to compare between the positional embedding model***(SEGMENT)***, the sentence embedding***(SENT POS)***, the absolute positional embedding***(ABSOLUTE POS)*** and no positional embedding one which is showed in Table 2. Overall, the positional embedding shows the best performance in the ROUGE score. Due to the help of position awareness, SEGMENT can capture long-range dependencies which are demonstrated in

the increasing of all results. SEGMENT with sentence embedding shows no enhancement in the result because the relation among order, context, and quantity of sentences are always changed, this leads to the noise representation for the model. In terms of SEGMENT with absolute embedding, the fixed positional embedding produces a non-flexible representation that performs very similar to the SEGMENT with sentence embedding.

5. Conclusion

This work presents a simple and efficient extractor architecture for the abstractive summarization system that remarkably affects the selection of words and phrases included in the summary. A comparison among different positional embeds showed that the learned positional embedding potentially makes the model aware of the positional context. Besides, the combination of SEGMENT and TCN leads to an astonishing improvement in ROUGE scores on CNN/DM dataset. The results proved that segment-augmented and TCN are the pivotal elements that increase the cohesion among phrases and filter unnecessary parts in the source input. In future work, attending self-supervised learning to SEGMENT would raise a potential approach that makes the model notice the intrinsic structure to capture long-term dependencies better.

Acknowledgements

This research is supported by research funding from Advanced Program in Computer Science, University of Science, Vietnam National University - Ho Chi Minh City.

References

[1] Bahdanau, D., Cho, K., Bengio, Y., 2016. Neural machine translation by jointly learning to align and translate.
[2] Bai, S., Kolter, J.Z., Koltun, V., 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling.
[3] Celikyilmaz, A., Bosselut, A., He, X., Choi, Y., 2018. Deep communicating agents for abstractive summarization.
[4] Chen, Y.C., Bansal, M., 2018. Fast abstractive summarization with reinforce-selected sentence rewriting.
[5] Cho, J., Seo, M., Hajishirzi, H., 2019. Mixture content selection for diverse sequence generation.
[6] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation.
[7] Gehrmann, S., Deng, Y., Rush, A.M., 2018. Bottom-up abstractive summarization.
[8] Hermann, K.M., Kočiský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., Blunsom, P., 2015. Teaching machines to read and comprehend.
[9] Kingma, D.P., Ba, J., 2017. Adam: A method for stochastic optimization.
[10] Li, C., Xu, W., Li, S., Gao, S., 2018. Guiding generation for abstractive text summarization based on key information guide network, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), Association for Computational Linguistics, New Orleans, Louisiana. pp. 55–60.
[11] Li, X., Yin, X., Li, C., Zhang, P., Hu, X., Zhang, L., Wang, L., Hu, H., Dong, L., Wei, F., Choi, Y., Gao, J., 2020. Oscar: Object-semantics aligned pre-training for vision-language tasks.
[12] Lin, C.Y., 2004. ROUGE: A package for automatic evaluation of summaries, in: Text Summarization Branches Out, Association for Computational Linguistics, Barcelona, Spain. pp. 74–81.
[13] Nallapati, R., Zhai, F., Zhou, B., 2016. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents.
[14] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K., 2016. Wavenet: A generative model for raw audio.
[15] Press, O., Wolf, L., 2017. Using the output embedding to improve language models, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, Association for Computational Linguistics, Valencia, Spain. pp. 157–163.
[16] See, A., Liu, P.J., Manning, C.D., 2017. Get to the point: Summarization with pointer-generator networks, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Vancouver, Canada.
[17] Sutskever, I., Vinyals, O., Le, Q.V., 2014. Sequence to sequence learning with neural networks.
[18] Tu, Z., Lu, Z., Liu, Y., Liu, X., Li, H., 2016. Modeling coverage for neural machine translation.
[19] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention is all you need.
[20] Vinyals, O., Fortunato, M., Jaitly, N., 2017. Pointer networks.
[21] Yu, F., Koltun, V., 2016. Multi-scale context aggregation by dilated convolutions.