Analysis on the number of layers in the transformer-based model for neural machine translation

Dongxing Li, Zuying Luo*
School of Artificial Intelligence, Beijing Normal University, Beijing 100875, China

ABSTRACT

Recently, the transformer-based models have been widely used in sequence-to-sequence (seq2seq) tasks, especially neural machine translation (NMT). In the original transformer, the layer number in encoder is equal to the layer number in decoder. However, the structure is more complex and task is more difficult in decoder than those in encoder, so the layer number should not be same. In order to verify how many layer number in encoder and decoder is properly valued, we improve transformer as our model and conduct four experiments on four translation tasks of IWSLT2017. The experimental results show that the layer number in decoder should be larger than that in encoder, which can bring better translation performance.

Keywords: Transformer, encoder, decoder, layers

1. INTRODUCTION

Since the transformer architecture was proposed by Vaswani[1], it has been widely used in a variety of seq2seq tasks, such as text summarization[2], question answering (QA)[3], and NMT. In NMT tasks, its performance far exceeds those using RNN[4-6] and CNN-based[7,8] networks. Currently, there are variants of transformer architecture which improve translation performance to a certain extent. They mainly include: (1) analysis of macro-architecture, such as sandwich Transformer[9], Universal transformers[10] and reformer[11]; (2) analysis and improvement of multi-head self-attention network[12]; (3) improvement of training methods, such as autoregressive translation (AT)[1] and non-autoregressive translation (NAT)[13-16]; (4) analysis of adopted structure, such as encoder2decoder[1,2,17,18] and unified encoder and decoder[19]; (5) improvements of the training data, such as incorporating lexical, syntactic and semantic information into the input[20-22].

This paper analyses the macro structure of transformer, which employs the encoder2decoder structure. In the original transformer, encoder and decoder have the same layer number. However, the structures and tasks of encoder and decoder are different and thus it is unreasonable to use the same layer number. Compared with the structure of the encoder, the structure of the decoder is more complex. Additionally, the task of the encoder is to understand the source sequence and the task of the decoder is to generate the target sequence, so the task of the decoder is more difficult. Therefore, it is argued that more decoder layers should be used to improve the translation performance.

The original transformer is still employed as our model. The difference is the fact that we use different layers in the encoder and decoder. Through four comparative experiments on four datasets of IWSLT 2017, it is found that more layers in the decoder than those in the encoder can produce better translation performance.

2. TRANSFORMER

Transformer was proposed by Vaswani in 2017 to solve NMT problem[1]. This architecture abandons the use of recurrence and convolutions, and only employs attention mechanisms. It consists of stacked encoders and decoders.

2.1 Encoder

The encoder is essentially a language model which is used to learn the distributed representation of source language sequence. Namely, the encoder's input is a source language sequence and each token is represented as a one-hot vector, which is mapped to a distributed representation in the embedding layer. And then all the representations of the sequence tokens are entered into encoder. The encoder consists of a self-attention layer and a feed-forward layer. Its core module is self-attention layer, which completely abandons the learning method of RNN and CNN and only relies on the attention mechanism.

* luozy@bnu.edu.cn

Third International Conference on Computer Science and Communication Technology (ICCSCT 2022)
edited by Yingfa Lu, Changbo Cheng, Proc. of SPIE Vol. 12506, 1250625
 · 0277-786X · doi: 10.1117/12.2661982

If the input is $X_s = (x_1, x_2, \mathrm{L}\ , x_m)$, where m and $x_i \in ¡^{|V_s|}$ denote the length of source sequence and a one-hot representation of the *i*th token, where $|V_s|$ is the vocabulary size of the source language. The output of the sequence is $Z = (z_1, z_2, \mathrm{L}\ , z_m)$, where $z_i \in ¡^{d_z}$. First, the one-hot matrix of the source sequence is transformed to the token embedding matrix $X \in ¡^{m \times d_x}$ through embedding layer using $W_E \in ¡^{|V_S| \times d_x}$. The embedding matrix is transformed to three intermediate matrices-queries Q, keys K and values V through three linear transformations by $w^Q, w^K, w^V \in ¡^{d_x \times d_z}$, where query is the matching matrix, key is the matched matrix and value is the value matrix of key. The query and key matrices are mapping relation.

Then, the mutual attention score between any pair of tokens in the source sequence is calculated and normalized to the attention weight, and finally multiplied by the values matrix to obtain the representation matrix of the sequence. It is calculated by:

$$\begin{aligned} &Attention\_score = QK^T \\ &Attention\_weight = soft\max(\frac{Attention\_score}{\sqrt{d_K}}) \cdot \\ &Attention = Attention\_weight \cdot V \end{aligned} \tag{1}$$

Feedforward neural network consists of two linear transforms and one Relu nonlinear transformation.

$$FFN(x) = \mathrm{Re}lu(x \cdot w_1 + b_1) \cdot w_2 + b_2 . \tag{2}$$

In equation (2), ***x*** represents an output feature matrix of previous layer. A short-cut connection layer and a normalization layer are connected behind the two sub-layers, respectively.

2.2 Decoder

The decoder has similar structure to the encoder. However, there are two differences. First, it uses masked self-attention layer instead of self-attention layer because of the AT property and adds an encoder-decoder attention layer after this layer. Residual and normalization operations are also performed after these three sublayers respectively.

Masked Self-attention Layer. The structure and its computation method of this layer are similar to the corresponding encoder's sub-layers. The interesting differences involve that: each token of encoder is visible and there is an attention relationship between the current token and all words; the current token of the decoder only focus on the generated token subsequence, while ignoring the future target tokens which means the attention weight between the current token and the future target tokens is 0. That's what "Masked" means. The self-attention relationship is still an asymmetric square matrix. In order to achieve the effect of "Masked", the position above the diagonal of the attention square is set to *–inf* because the exponential operation is required to calculate the attention using the softmax function.

Encoder-decoder Attention Layer. This layer is also implemented in a similar way to the self-attention layer in encoder. The shape of attention matrix is $n \times m$ where n and m denote the lengths of the target and source sequences and is subjected to $\forall 1 \le i \le n, \sum_{j=1}^{m} A_{ij} = 1$. If the length of source language sequence is equal to the length of target language sequence, namely $m = n$, the matrix is an asymmetric square matrix. In addition, when computing attention scores, the Q matrix is obtained from the masked self-attention layer in decoder, and the K, V matrices are obtained from the latent variable matrix of last layer in encoder.

3. MODEL ARCHITECTURE

As is shown in Figure 1, the original transformer model is revised as our model. In the original transformer model, the layer number in encoder is same as that in decoder, namely $m = n = 6$, where m and n denote the layer number in encoder and decoder, respectively. The main purposes of this model are: (1) to discriminate whether the original transformer model is appropriate when choosing the layer number as 6 in encoder and decoder; (2) to judge whether the layer number in encoder and decoder must be equal. Since the layer structure of decoder is relatively more complex, its generation task is more difficult and meanwhile layer number is an important hyper-parameter, so we mainly analyse the

impact of layer number on the translation performance. To ensure comparative experiments, all other hyper-parameters and training methods remain unchanged. The comparison experiments of the models include: (1) the layer number in encoder is same as that in decoder and varies from 1 to 12; (2) the layer number in decoder is fixed as 6 and the layer number in encoder varies 1 to 12; (3) the layer number in encoder is fixed as 6 and the layer number in decoder varies from 1 to 12; (4) the layer number in encoder is fixed as 3, and the layer number in decoder varies from 7 to 12.

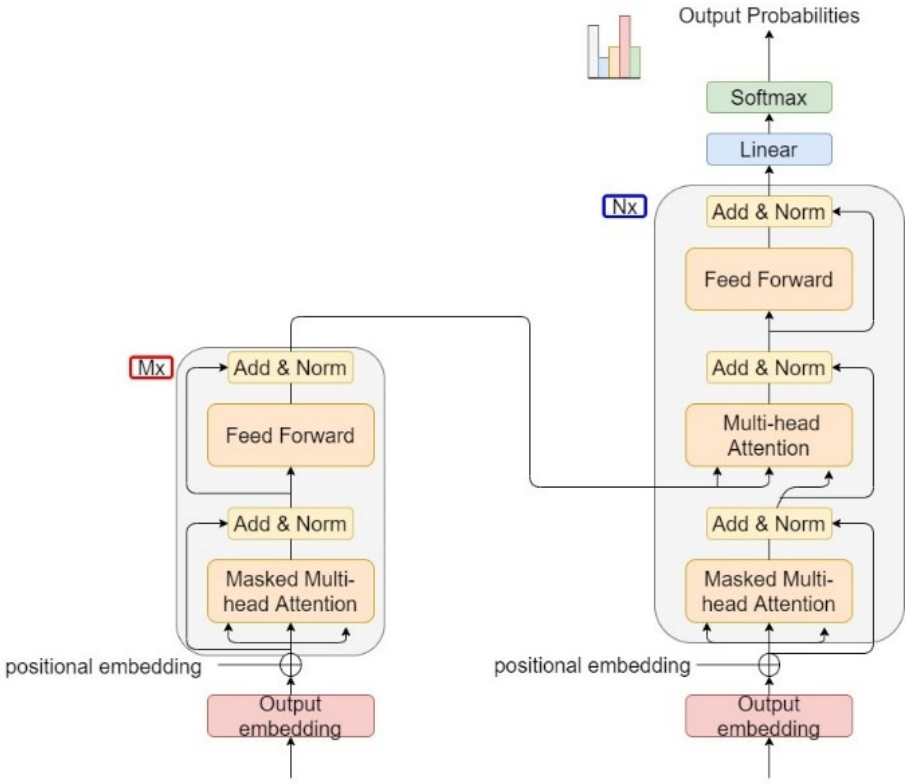

Figure 1. The variance of Transformer model.

4. EXPERIMENT

4.1 Datasets

The IWSLT2017 dataset, consisting of the DE->NL, DE->EN, NL->EN, and EN->RO datasets, is used as experimental benchmark. The training set of all datasets contains about 20,000 sequence pairs, and dev2010 and test2010 are used as validation and test sets, respectively. Table 1 summarizes the statistics of all datasets. Four datasets are preprocessed by Moses, a tool for statistical machine translation. Firstly, all sequence pairs are tokenized to deal with the punctuations. Then, each dataset is truncated. Finally, to increase the granularity of words and learn the relationship of affixes, all the sequences are encoded with bytes pair encoding (BPE), which can be implemented with subword-nmt (https://github.com/rsennrich/subword-nmt). Additionally, a respective dictionary is learned for all preprocessed datasets, respectively.

Table 1. The training and test datasets for our model.

IWSLT2017	DE->EN	DE->NL	NL->EN	EN->RO
Training	206,112	213,628	237,240	220,538
Dev2010	888	1,001	1,003	914
Test2010	1,568	1,779	1,777	1,678

4.2 Setting

For translation evaluation metric, BLEU (Bilingual Evaluation Understudy) is chosen for our tasks, which has been currently one of the most popular evaluation metrics for NMT. It indicates the precision between the reference and translation using the N-gram tokens matching. The learning rate is determined by the warm-up strategy[1]. When the model is being trained, all dropouts are set to 0.3. The dimension of the hidden state for the linear transformation in each encoder layer is 2,048 and model dimension is set to 512. The number of attention heads is 8. All experiments are conducted using Tensorflow 1.12.0 with the reference project (https://github.com/Kyubyong/transformer). All experiments are implemented using one NVIDIA Tesla V100 GPU with 32-GB memory. The transformer-base model is chosen as our baseline, which has 6 layers for the encoder and decoder, respectively.

4.3 Results

(1) The layer number in encoder is same as that in decoder and varies from 1 to 12.

As is shown in Figure 2, when the layer number in encoder and decoder are equal, the translation performance curve is concave. Specifically, the inflection point of curve occurs when the layer number is three or four. When the layer number is greater than 6, the translation performance drops sharply.

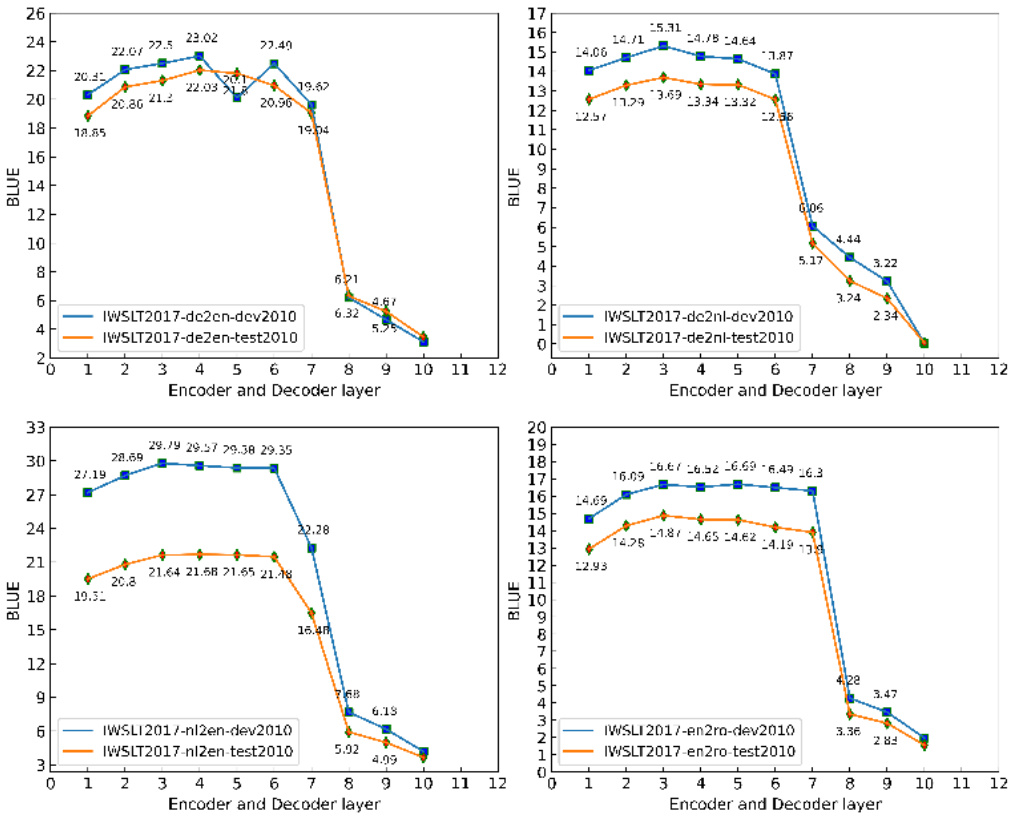

Figure 2. Translation performance curves with the same layer number in encoder and decoder.

(2) The layer number in decoder is fixed as 6 and that in encoder varies from 1 to 12.

As is shown in Figure 3, when the layer number in encoder increases from 1 to 6, the translation performance changes slightly. However, when it increases from 6 to 12, the translation performance drops sharply. Simultaneously, it shows that: (1) The layer number in encoder cannot exceed that in decoder. When the layer number in decoder is greater than 6, the translation performance shows a linear downward trend. (2) The translation performance peaks when the layer number in encoder is 3 instead of 6.

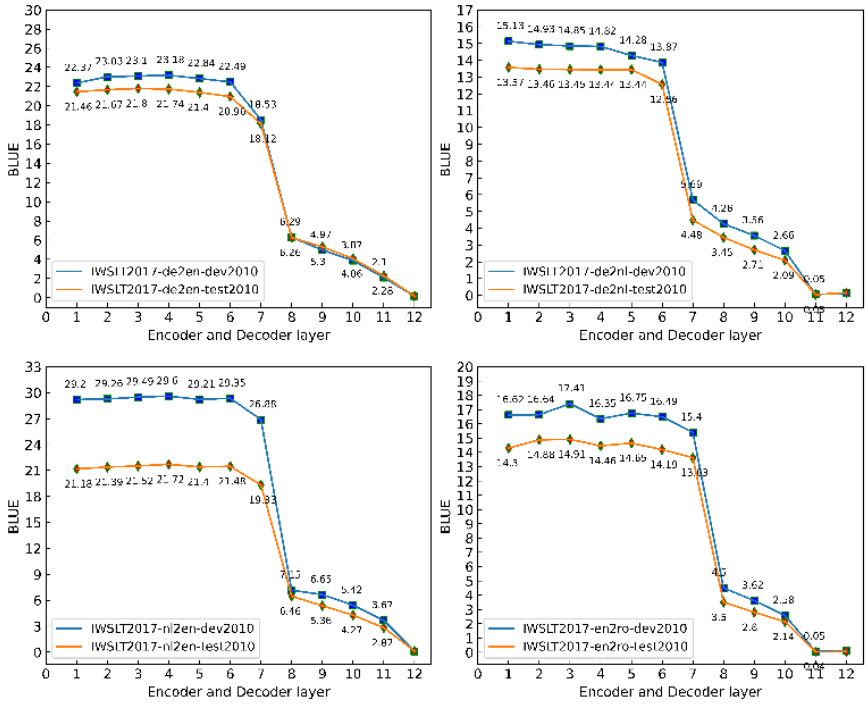

Figure 3. Translation performance curves with the fixed layer number in decoder and changing layer number in encoder.

(3) The layer number in encoder is fixed as 6 and that in decoder varies from 1 to 12.

As is shown in Figure 4, the translation performance is positively correlated with the layer number in decoder. Specifically, when the layer number in decoder varies from 1 to 6, the growth trend is more obvious, and when it increases from 6 to 12, the translation performance is improved slightly.

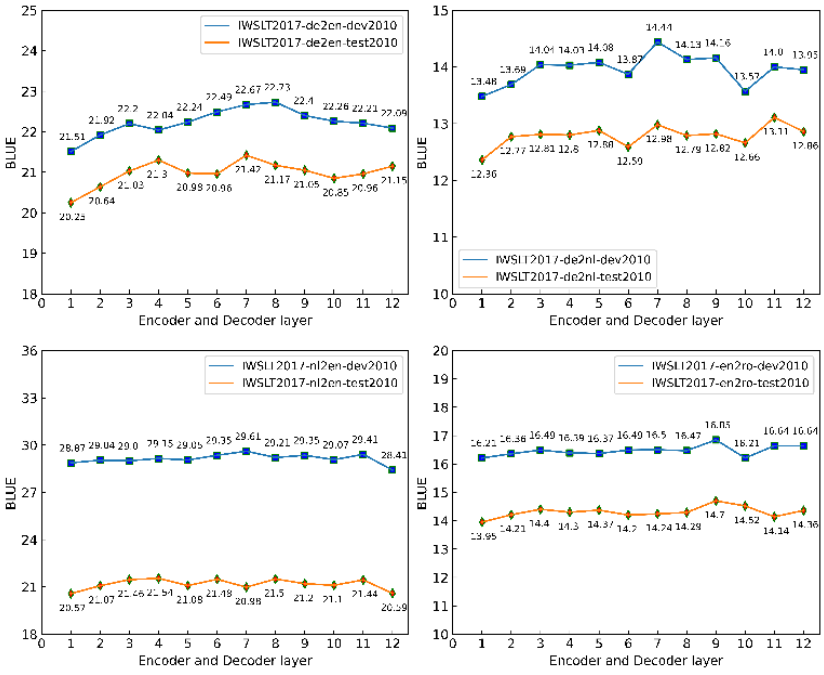

Figure 4. Translation performance curves with the fixed layer number in encoder and changing layer number in decoder.

(4) More difference of the layer number in encoder and decoder.

It can be seen from experiment 3 that fixing layer number in encoder and increasing layer number in decoder can improve translation performance. Therefore, we test to increase more difference of layer number in encoder and decoder to discriminate whether to improve the translation performance. As is shown in Figure 5, the experimental results show that the translation results are not actually improved.

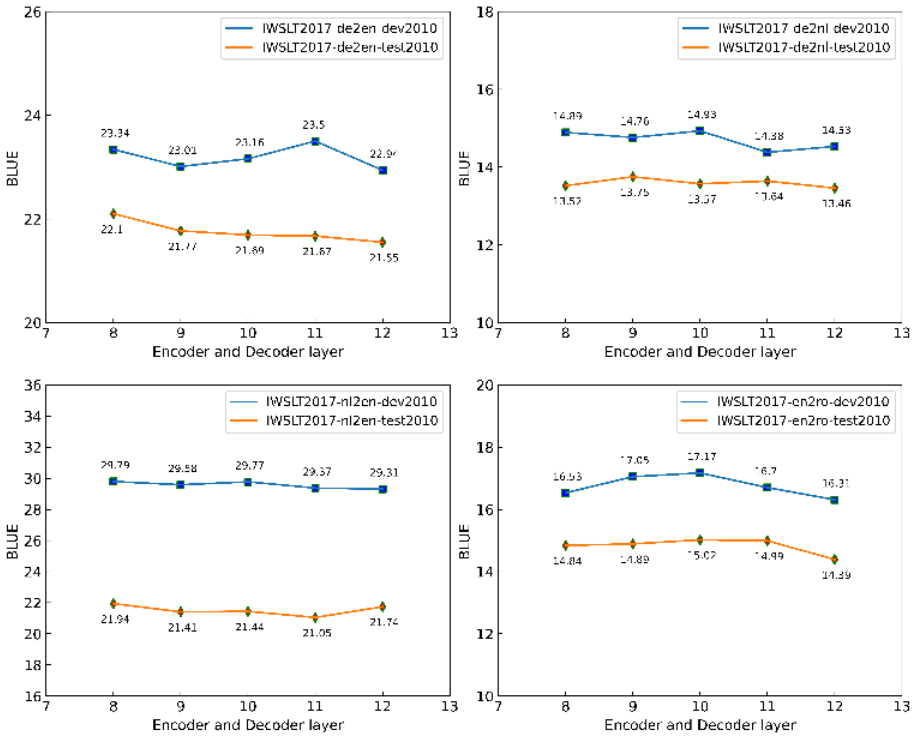

Figure 5. Translation performance curves with the fixed layer (3) in the encoder and changing layers {7, 8, 9, 10, 11, 12} in the encoder.

5. CONCLUSION

In this paper, the only difference between our proposed model and the original transformer model is the setting of layer number in encoder and decoder. It is proved that the layers in the decoder should be different from those in the encoder. Specifically, in Section 4.3, the experiment (1) shows that a bigger layer number brings worse results with the same layer number in encoder and decoder and when the layer number is slightly less than 6 brings a peak translation performance; the experiments (2) and (3) show that the layers in decoder should be more than those in encoder; the experiment (4) shows that the more difference of layers between encoder and decoder brings better translation performance.

In conclusion, the results of four experiments show that: (1) when the layer number in encoder and decoder is same as 6, the result is not optimal but {3, 4, 5} is a better choice set; (2) due to the complex structure and difficult generation task of decoder, its layer number is preferably greater than that in encoder, and the larger number leads to better result.

ACKNOWLEDGMENTS

We are grateful for the supports by projects of the National Natural Science Foundation of China (61977009) and the Ministry of Education, P.R.C.-Chinese Mobile Research Foundation (MCM20200404).

REFERENCES

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I., "Attention is all you need," Advances in Neural Information Processing Systems 30, (2017).

[2] Song, K., Tan, X., Qin, T., Lu, J. and Liu, T. Y., "Mass: Masked sequence to sequence pre-training for language generation," Preprint arXiv:1905.02450, (2019).

[3] Rajpurkar, P., Zhang, J., Lopyrev, K. and Liang, P., 2016 "Squad: 100,000+ questions for machine comprehension of text," Preprint arXiv:1606.05250, (2016).

[4] Bahdanau, D., Cho, K. and Bengio, Y., "Neural machine translation by jointly learning to align and translate," Preprint arXiv:1409.0473, (2014).

[5] Cho, K., Van Merriënboer, B., Bahdanau, D. and Bengio, Y., "On the properties of neural machine translation: Encoder-decoder approaches," Preprint arXiv:1409.1259, (2014).

[6] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," Preprint arXiv:1406.1078, (2014).

[7] Gehring, J., Auli, M., Grangier, D. and Dauphin, Y. N., "A convolutional encoder model for neural machine translation," Preprint arXiv:1611.02344, (2016).

[8] Gehring, J., Auli, M., Grangier, D., Yarats, D. and Dauphin, Y. N., "Convolutional sequence to sequence learning," Proc. of the 34th Inter. Conf. on Machine Learning, 1243-1252, (2017).

[9] Press, O., Smith, N. A. and Levy, O., "Improving transformer models by reordering their sublayers," Preprint arXiv:1911.03864, (2019).

[10] Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J. and Kaiser, Ł., 2018 "Universal transformers," Preprint arXiv:1807.03819, .

[11] Kitaev, N., Kaiser, Ł. and Levskaya, A., 2020 "Reformer: The efficient transformer," Preprint arXiv:2001.04451.

[12] Tay, Y., Bahri, D., Metzler, D., Juan, D. C., Zhao, Z. and Zheng, C., 2021 "Synthesizer: Rethinking self-attention for transformer models," Proceedings of the 38th Inter. Conf. on Machine Learning PMLR, 10183-10192.

[13] Gu, J., Bradbury, J., Xiong, C., Li, V. O. and Socher, R., 2017 "Non-autoregressive neural machine translation," Preprint arXiv:1711.02281.

[14] Kim, Y. and Rush, A. M., 2016 "Sequence-level knowledge distillation," preprint arXiv:1606.07947.

[15] Lee, J., Mansimov, E. and Cho, K., "Deterministic non-autoregressive neural sequence modeling by iterative refinement," Preprint arXiv:1802.06901, (2018).

[16] Ghazvininejad, M., Levy, O., Liu, Y. and Zettlemoyer, L., "Mask-predict: Parallel decoding of conditional masked language models," Preprint arXiv:1904.09324, (2019).

[17] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. and Zettlemoyer, L., "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," Preprint arXiv:1910.13461, (2019).

[18] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y. Q. and Liu, P. J., "Exploring the limits of transfer learning with a unified text-to-text transformer," Preprint arXiv:1910.10683, (2019).

[19] Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M. and Hon, H. W., "Unified language model pre-training for natural language understanding and generation," Preprint arXiv:1905.03197, (2019).

[20] Hao, J., Wang, X., Shi, S., Zhang, J. and Tu, Z., "Multi-granularity self-attention for neural machine translation," Preprint arXiv:1909.02222, (2019).

[21] Hao, J., Wang, X., Shi, S., Zhang, J. and Tu, Z., "Towards better modeling hierarchical structure for self-attention with ordered neurons," Preprint arXiv:1909.01562, (2019).

[22] Song, L., Gildea, D., Zhang, Y., Wang, Z. and Su, J., "Semantic neural machine translation using AMR," Conf. on Transactions of the Association for Computational Linguistics, 19-31 (2019).