Proceedings of the 3rd International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/6/20230879

Comparative analysis between application of transformer and recurrent neural network in speech recognition

Zhehao Liao

University of Nottingham, Ningbo, China, 315100

kagurazakayumeno@gmail.com

Abstract. Transformer is a deep learning model applying self-attention mechanism which is widely used in solving sequence-to-sequence questions, including speech recognition. After Transformer was proposed, it has been greatly developed and made great progress in the field of speech recognition. Recurrent Neural Network (RNN) is also a model that can be used in speech recognition. Speech recognition is a kind of sequence-to-sequence question that can transform human speech into text form. Both RNN and Transformer use encoder-decoder architecture to solve sequence-to-sequence questions. However, RNN is a recurrent model, weak in parallel training, and it will not perform quite well as Transformer in sequence-to-sequence question, which is a non-recurrent model. This paper mainly analyzes the accuracy Transformer and RNN in automatic speech recognition. It shows that Transformer performs better than RNN in speech recognition area, having higher accuracy, and it therefore provides evidence that Transformer can be an efficacious approach to automatic speech recognition as well as a practical substitution for traditional ways like RNN.

Keywords: Transformer, Recurrent Neural Network, Comparative Analysis.

1. Introduction

Transformer is a non-recurrent sequence-to-sequence model introduced by Vaswani et al. [1], and it was initially used to solve the questions of machine translation. They used Transformer for 2014 English-to-France task and achieved state-of-the-art performance with little training costs. Afterwards, Transformer is also widely used in many other sequence-to-sequence questions, such that image segmentation, chat bot and text-to-speech [2-4].

Recurrent neural network (RNN) is a neural network model derived from feedforward networks, and it uses time series data to process sequential input. It obtains information from prior elements to determine the current input and output, which means that the output will depends on the prior elements of the sequence. RNN is also a common model for solving sequence-to-sequence questions. It is widely used for machine translation and writing recognition [5, 6].

Transformer applies self-attention mechanism, which will use position-pair computation to extract the global features of the sequence by positional encoding. With the applying of self-attention, it just needs to be calculated once to obtain the dependencies of the sequence between different positions. Compared with RNN, it will be calculated one by one by position-chain computation of RNN. Therefore, for sequence-to-sequence tasks, Transformer depending on self-attention mechanism can perform better than RNN [7].

DOI: 10.54254/2755-2721/6/20230879

Speech recognition is a kind of sequence-to-sequence question. It means that transferring human speech into text form, which can make people interpret what a people said easily. And automatic speech recognition is that the machine will depend on the speech features, giving the probabilities of each possible word next and therefore inferring the context. Transformer and RNN are two models that widely used in many sequence-to-sequence questions, and it seems that they can also be used in solving such questions. It is reported that both Transformer and RNN are applicable to automatic speech recognition [8, 9].

This paper mainly analyzes the accuracy Transformer and RNN in automatic speech recognition, and it may provide some evidence that transformer can be an efficacious approach to automatic speech recognition as well as a practical substitution for traditional ways like RNN.

2. Methodology

This paper is mainly aimed to compare the performance of Transformer and RNN in the field of automatic speech recognition, and the core parameters needed to study are the accuracy of recognition and their training speeds. Meanwhile, it is needed to pay attention to the principle of automatic speech recognition that the machine will make predictions by the speech features. Therefore, the conditional probability is applied for the accuracy comparison.

Meanwhile, in this paper, the corpora taken into consideration are AISHELL-1, CSJ and LiBriSpeech [10-12]. AISHELL-1 is a Mandarin corpus, and the two test sets are dev and test. CSJ is a Japanese corpus, and the three test sets are eval1, eval2 and eval3. LiBriSpeech is an English corpus, and the four test sets are dev clean, dev other, test clean and test other.

To compare the accuracy, error rate is introduced, which is calculated by the number of errors divided by the total number. And the model with lower error rate has higher accuracy. For the languages like Chinese and Japanese which consist of characters, character error rate (CER) will be applied, while for the languages like English which consist of words, word error rate (WER) will be applied.

2.1. Description

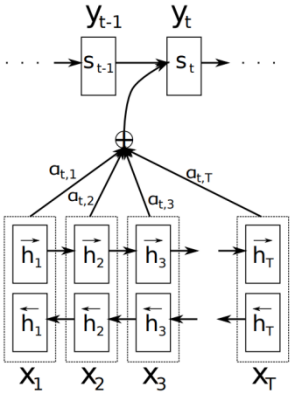

Figure 1. Model architecture of RNN [13]

Recurrent neural network (RNN) is a neural network also based on encoder-decoder architecture. According to Bahdanau, D., Cho, K. and Bengio, Y. [13], as shown in Figure 1, it consists of the hidden state h, the input sequence x and the output y produced by x. The hidden state of the network, h_t is updated at each time step t by

$$h_t = \mathrm{f}(h_{t-1}, x_t) \tag{1}$$

where f is a non-linear activation function like logistic sigmoid function. Then the context vector c_i is calculated as a weight sum

$$c_i = \sum_{j=1}^{T} a_{ij} h_j \tag{2}$$

DOI: 10.54254/2755-2721/6/20230879

where the weight a_{ij} depends on the probability model selected.

In the situation of automatic speech recognition, it can be trained to predict the next element in the sequence by learning probability distributions. At each timestep t, the conditional probability distribution of the output is $\mathrm{p}(x_t \mid x_{t-1}, \ldots, x_1)$, and which the network is needed to do is to maximize the probability.

2.1.1. Encoder. The encoder is an RNN where the input sequence is input and read by element. After input, the hidden state of the network is updated according to equation (1). At the last element of the sequence, which is often marked by a particular symbol, all the hidden states of the RNN are a summary, sent to the decoder.

2.1.2. Decoder. The decoder is another RNN, aiming at giving the maximal probability of next symbol y_t to predict and generate the output sequence y given the hidden state h_t produced in the encoder. And it is different from the encoder that both y_t and h_t are influenced by y_{t-1} and the summary of the hidden states about input sequence. Therefore, to calculate the condition probability distribution of output, the two influence factors should be taken into consideration.

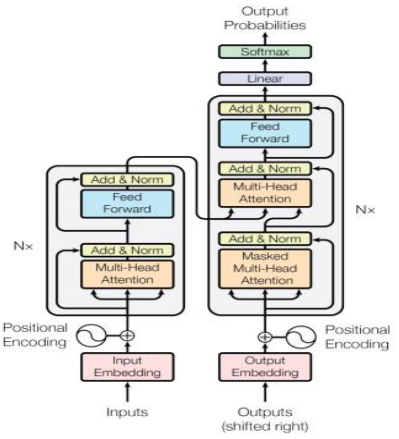

Figure 2. Model architecture of the Speech-Transformer [1]

Transformer is based on encoder-decoder architecture, like other sequence-to-sequence models (Figure 2). Its encoder and decoder both have several layers. They do the similar work that encoder layer processes the input layer by layer, and decoder layer also processes the output iteratively. The encoder layer and the decoder layer both consist of multi-head attention layer and position-wise feed-forward networks. Moreover, the part that decoder layer is more than the encoder layer is the masked multi-head attention layer.

2.2. Self-attention

Self-attention is a mechanism that measures the relevance among different positions of a sequence. It will consider all the features of input sequence. It has three inputs, query (Q), key (K) and value (V). Attention score is calculated by a designed function of both query and key, and then the output is the weighted sum of values and attention scores. As Vaswani et al. says [1], for one-sequence input, the formula of the scaled dot-product attention is

$$Attention(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = softmax\left(\frac{\mathrm{QK}^T}{\sqrt{d_k}}\right)V \tag{3}$$

where $K, V \in R^{d_k \times d_m}$, $Q \in R^{d_q \times d_m}$, d_k is the length of keys and values and d_q is the length of queries and d_m is the feature number. Scaling the result by $\frac{1}{\sqrt{d_m}}$ is to prevent the vanishing gradient problem when d_m is large.

DOI: 10.54254/2755-2721/6/20230879

For multi-sequence input, multi-head attention is needed instead of one-head attention, and the head number is the number of the input sequence, denoted by d_h. Multi-head attention is the core structure of Transformer, and it is used for processing multi-sequence input to get all features. It calculates attention d_h times like the scaled dot-product attention. The independent outputs are concatenated and changed into the corresponding dimension.

$$MultiHead(Q,K,V) = Concat(head_1, \dots, head_{d^h})W_O \tag{4}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{5}$$

where $W_i^Q, W_i^K, W_i^V \in R^{d_m \times d_k}$, $W_O \in R^{d_h d_k \times d_m}$, d_h is the number of attentions in the layer.

2.2.1. Encoder. To obtain the final outputs, the sum of input encoding and positional encoding are inputted to the encoder block, and the block has two layers. First layer is multi-head attention layer, and its queries (Q), keys (K) and values (V) are from the previous outputs. Second layer is position-wise feed-forward networks, and the dimension of inner layer is denoted by d_i. Layer normalization and residual connection are also applied for more effective training.

2.2.2. Decoder. Initially, output embedding is used to transform the sequence into the output with appropriate dimension, and then it is added with the positional encoding. Afterwards, the summation of the result is inputted to the decoder block. Compared with the encoder, the decoder block has three layers. First layer is a masked multi-head attention layer. Like normal multi-head attention layer, it has the similar queries, keys and values, and the mask means that the predictions of outputs can only depend on the known information and avoid being influenced by the 'future' information. Second and third layer resemble the encoder block with a multi-head attention layer and position-wise feed-forward networks. Moreover, layer normalization and residual connection are also introduced to each layer in the decoder block. Finally, through the decoder block, the fully connected linear layer is applied to transform the outputs into the probabilities.

2.2.3. Positional encoding. To know the sequence order information, positional encoding can be added to the input representation by relative or absolute positional information. Positional encoding can be defined as:

$$PE_{i,\ j} = \begin{cases} \sin\left(\frac{i}{10000^{\frac{j}{d}}}\right) & if\ j\ is\ even \\ \cos\left(\frac{i}{10000^{\frac{j}{d}}}\right) & if\ j\ is\ odd \end{cases} \tag{6}$$

where PE represents the position, and (i, j) are the row and column.

The positional encoding works because PE is linear. Namely, for arbitrary k, linear function of $PE_{i,\ j}$ can be substituted for $PE_{i,\ j+k}$.

3. Analysis

For RNN, we followed the architecture constructed for the speech recognition from Zeyer A. et al. [8] and Hori T., Cho J. and Watanabe S. [14]. And the Adadelta optimizer is used for training [15]. The same architecture is applied for Transformer in [4] for AISHELL-1 and CSJ (e = 12, d = 6, di = 2048, dh = 4, dm = 256) and LibriSpeech (e = 12, d = 6, di = 2048 dh = 8, dm = 512). And according to Dong L., Xu S. and Xu B. [7], Adam optimizer is used for training with $\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^{-9}$, and its learning rate varies as the formula:

$$lr = k \times d_m^{-0.5} \times \min(n^{-0.5}, n \times n_w^{-1.5}) \tag{7}$$

DOI: 10.54254/2755-2721/6/20230879

where n is the step number and k is a tunable number and n_w is the number of first training steps where the learning rate increases. After n_w steps, learning rate begins to decrease in inverse proportion to $n^{-0.5}$. This study sets $n_w = 25000, k = 10$ and decrease k to 1 when converging. In addition, the residual and attention dropout are both set as 0.1.

According to Karita, S. et al. [4], for the two test sets, dev and test in AISHALL, the CER of RNN is 6.8% and 8.0%, while that of Transformer is 6.0% and 6.7%. For the three test sets, eval1, eval2 and eval3 in CSJ, the CER of RNN is 6.6%, 4.8% and 5.0%, while that of Transformer is 5.7%, 4.1% and 4.5%. For the four test sets, dev clean, dev other, test clean and test other in LiBriSpeech is 3.1%, 9.9%, 3.3%, 10.8%, while that of Transformer is 2.2%, 5.6%, 2.6%, 5.7%. All these data illustrate that Transformer outstrips RNN on these corpora. And it shows that Transformer can somewhat appositely substitute for RNN in the field of speech recognition.

4. Conclusion

This paper mainly focused the following part: (1) The rudimentary construction of RNN and Transformation as well as the background of speech recognition; (2) The training architecture and optimizer applied in the learning of RNN and Transformation (3) Comparison with the accuracy between RNN and Transformation among the three datasets AISHELL-1, CSJ and LiBriSpeech.

In order to improve this research, first is to improve the study of efficiency between RNN and Transformer, including convergent speed, for further comparison. If Transformer reaches the convergence earlier than RNN, it means that the training speed of Transformer is also faster than RNN, and therefore, the conclusion can be that regarding training efficiency, Transformer also outperforms RNN. Second is that more corpora about one language can be taken into consideration to show that Transformer performs better more credibly. There are also plenty of corpora in Chinese, Mandarin, English which can be used for training. And their application can somewhat avoid the contingency that Transformer is only suited to the three corpora but does not perform well in other corpora.

5. Acknowledgement

Thanks to Dr. Ming Yang from University of Macau and Prof. Shlomo Ta'asan from Carnegie Mellon University.

References

[1] Vaswani A. et al. (2017) Attention is all you need, Advances in Neural Information Processing Systems, 30, pp. 5998–6008.

[2] Zhang L. et al. (2020) Generalizing Deep Learning for Medical Image Segmentation to Unseen Domains via Deep Stacked Transformation, IEEE transactions on medical imaging, 39(7), pp. 2531–2540.

[3] Yan R. (2018). Chitty-Chitty-Chat Bot, Deep Learning for Conversational AI. In IJCAI, 18, pp. 5520-5526.

[4] Karita S., et al. (2019) A Comparative Study on Transformer vs RNN in Speech Applications, 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 449-456.

[5] Vathsala M.K. and Holi G. (2020) RNN based machine translation and transliteration for Twitter data, International journal of speech technology, 23(3), pp. 499–504.

[6] Sun L. et al. (2017) GMU: A Novel RNN Neuron and Its Application to Handwriting Recognition, 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 1, pp. 1062–1067.

[7] Dong L., Xu S. and Xu B. (2018) Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition, 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 5884–5888.

[8] Zeyer A. et al. (2018) Improved training of end-to-end attention models for speech recognition,

DOI: 10.54254/2755-2721/6/20230879

arXiv preprint, arXiv:1805.03294.
[9] Chen N. et al. (2021) Non-Autoregressive Transformer for Speech Recognition, IEEE signal processing letters, 28, pp. 121–125.
[10] Bu H. et al. (2017) AISHELL-1: An Open-Source Mandarin Speech Corpus and A Speech Recognition Baseline, 2017 20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA), pp. 1–5.
[11] Maekawa K. et al. (2000) Spontaneous Speech Corpus of Japanese, LREC, 6, pp. 1-5.
[12] Panayotov V. et al. (2015) Librispeech: An ASR corpus based on public domain audio books, 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5206–5210.
[13] Bahdanau D., Cho K. and Bengio Y. (2014) Neural Machine Translation by Jointly Learning to Align and Translate, arXiv preprint, arXiv:1409.0473.
[14] Hori T., Cho J. and Watanabe S. (2018) End-to-end Speech Recognition with Word-based RNN Language Models, 2018 IEEE Spoken Language Technology Workshop (SLT), pp. 389-396.
[15] Zeiler M.D. (2012) ADADELTA: An Adaptive Learning Rate Method, arXiv preprint, arXiv:1212.5701.