Research on Task-oriented Dialogue Based on Modified Transformer

ZongLi Jiang\textsuperscript{a}, Shuo Zhang\textsuperscript{b}
Faculty of Information Technology, Beijing University of Technology, Beijing, China
\textsuperscript{a}jiangzl@bjut.edu.cn, \textsuperscript{b}2131993zs@163.com

ABSTRACT
The traditional end-to-end task-oriented dialogue models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. But in the case of large amounts of data, there are many types of questions. It performs poorly when answering multiple types of questions, memory information cannot effectively record all the sentence information of the context. In view of the above this, this article uses a modified transformer model to overcome the problems mentioned in dialogue tasks. Transformer is a model constructed using attention mechanisms, which completely discards the method of RNN (recurrent neural networks), and its structure includes two sub-parts of Encoder and decoder. It uses residual network, batch normalization, and self-attention mechanism to build the model structure, uses Positional Encoding to capture sentence information, which can speed up model training convergence and capture Longer sentence information. In this paper, we modified the activation function in the transformer and use label smoothing to optimize the training to make the model's expressive ability better than previous.

CCS Concepts:
Computing methodologies → Artificial intelligence → Natural language processing → Discourse, dialogue and pragmatics

1. INTRODUCTION
The task-oriented dialogue system is an outstanding man-machine dialogue system. In recent years, it has been widely used in various industries. It can answer users’ questions by referring to resolution and query completion.

A task-oriented dialogue system helps users answer specific questions, such as schedules and customer service questions, in natural language. Traditionally, they were built with several pipeline modules: language understanding, dialog management, knowledge query, and language generation. In addition, in a task-oriented dialog system, the ability to query the external knowledge base is critical because the response is guided not only by the conversation history but also by professional guidance. Modeling the dependencies between modules is very complex. Usually a dialogue system requires a lot of manual design rules.

Recently, an end-to-end method of introducing an attention-based replication mechanism in a recurrent neural network model has shown promising results in dialog tasks. It can map plain text conversation history directly to the output response, and does not require manually labeled status tags. When unknown labels appear in the conversation history, the model is still able to produce correct and relevant entities.

However, although the mentioned above approaches were successful, they still suffer from two main problems: 1) recurrent models usually consider calculations along the symbol positions of the input and output sequences. Align the position with the steps in the calculation time, they will generate a series of
hidden states $h_t$ based on the previous input of the hidden state $ht$ and the position $t$. This inherently sequential nature prevents parallelization within training examples, which greatly reduces training speed. Although recent work has achieved significant improvements in computational efficiency through factorization tricks and conditional computation, the fundamental constraint of sequential computation, however, remains. 2) When there are many types of dialogue questions, or the amount of dialogue data is large, traditional models perform less accurately in answering multiple questions. Because models based on RNN often wait for the results of the previous step, in order to speed up training convergence and improve real-time performance, it is often not the first to consider designing models with high complexity. RNN puts a limit on the expressive power of the model.

To address these problems, we use a model modified based on Transformer, which is a model architecture that avoids recurrence, and instead relying entirely on the self-attention mechanism to draw global dependencies between inputs and outputs. In addition, this model allows for significantly more parallelization and can reach a new state of the art in dialog tasks.

2. Model Architecture

Today, most neural networks have an encoder-decoder structure. an input sequence of symbol representations is processed into memory after the encoder, the decoder then generates an output sequence of symbols one part at a time. The transformer certainly follows this completely, using stacked self-attention and point-wise, fully connected layers for the encoder and decoder, as shown in the left half and right half of Figure 1, respectively.

![Figure 1: The Transformer - model architecture.](image)

2.1 Attention

Attention mechanism was first applied in computer vision and then applied and performed well in the NLP. Its main function is to focus limited attention on key information, thereby saving computer resources and quickly obtaining the most effective information.
Compared with RNN and CNN networks, it has the following three advantages:

**Fast**: The Attention mechanism solves the problem of parallel computing. It does not depend on the calculation result of the previous step at each step of the calculation, so it can process data in parallel like CNN.

**Fewer parameters**: Compared with CNN and RNN, the complexity of the model is smaller and the parameters are fewer. Therefore, it requires less computing power.

**Work well**: RNN has a weak memory of long-distance sentence information, just like people can’t remember long time ago. Attention is aimed at key information records, which can effectively record important information.

Attention mechanism is to calculate the weight of value. First calculate the query and key similarity to get a temporary weight, then normalize the temporary weight to get a new weight, at last perform weighted summation on the value. Different types of Attention mechanisms calculate weights differently. The attention mechanism in our model will be described in detail in the next section.

### 2.2 Self-Attention

The attention function can be described as mapping a query and a set of key-value pairs to the output, and the query, key, value, and output are all vectors. The output is calculated as weighted sum values, where the weight assigned to each value is queried by using the corresponding key.

As we all know, "Scaled Dot-Product Attention" can compute the dot products of the query with all keys, and divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values. The input consists of queries and keys of dimension $d_k$, and values of dimension $d_v$, the matrix of outputs as:

$$Attention(Q, K, V) = \text{soft max}(\frac{QK^T}{\sqrt{d_k}})V$$

Multi-Head Attention is a mosaic of eight Scaled Dot-Product Attentions. The approach of the model here is to let the eight attentions learn different features in their respective training processes, and then combine these features together. In our work, we use $d_k = d_v = d_{\text{model}}/h=64$ and employ $h=8$ parallel attention layers so that the output has the same dimensionality:

$$MultiHead(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_n)W^o$$

where $\text{head}_i = Attention(QW^Q_i, KW^K_i, VW^V_i)$
2.3 Encoder and Decoder Stacks
The encoder consists of a stack of N = 6 identical layers. Each layer has two sub-layers. They are a combination of a multi-head self-attention and a fully connected feed-forward network. We use a residual network connection around each of the two sublayers, and then implement layer normalization. In other words, the output of each sub-layer is LayerNorm (x + Sublayer (x)).

The decoder also consists of a stack of N = 6 identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer that pays multiple attention to the output of the encoder stack. Similar to the encoder, the decoder also uses residual connections to sub-layers, and then follows layer normalization. Author of transformer also modified the self-attention sub-layer in the decoder stack to prevent locations from focusing on subsequent locations. Because we want to ensure that the prediction of position i can only depend on known entities smaller than position i.

2.4 Residual network
Modern neural networks are generally optimized by gradient-based BP algorithms. For feed-forward neural networks, it is generally necessary to propagate the input signal forward, then back-propagate the error and use gradient update methods to update the parameters. The update of a parameter of the first layer needs to calculate the gradient of the loss ε, which is calculated according to the chain rule as follows:

$$\delta^{(l)} = \frac{\partial z^{(l+1)}(x)}{\partial z^{(l)}} \cdot \delta^{(l+1)}$$

if define

$$\gamma^{(l)} = \left\| \frac{\partial z^{(l+1)}}{\partial z^{(l)}} \right\|$$

then $$\delta^{(l)} = \gamma^{(l)} \delta^{(l+1)}$$

When $$\gamma^{(l)} < 1$$, the error term of the first layer is smaller than that of the latter layer. If this is the case in many layers, the gradient will gradually disappear in the back propagation, and the underlying parameters cannot be updated effectively. This is the vanishing gradient problem. When $$\gamma^{(l)} > 1$$, the gradient will increase exponentially, causing a gradient explosion problem.
Residual network is here to solve the problem of gradient dispersion. Its unit is implemented in the form of layer-hopping connection. That is, the input of the unit is directly added to the output of the unit and then activated. Therefore, the residual network can easily use in mainstream. Deep learning framework is implemented by directly updating parameters using the BP algorithm. In the BP process, it is guaranteed that the gradient has at least one value 1.

![Figure 3: Residual Network](image)

### 2.5 Batch Normalization

Batch Normalization is calculated on the basis of mini-batch. It can drag the distribution of the data to a distribution with a mean value of 0 and a variance of 1. It slows down the internal Covariate Shift and accelerates the model training convergence. It is independent normalization of each feature. In actual operation, it is added between the neural network and the activation function.

Formula calculation process:

\[
Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}
\]

\[
\mu = \frac{1}{m} \sum_{i=1}^{m} z^{[l](i)}
\]

\[
\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} (z^{[l](i)} - \mu)^2
\]

\[
\gamma \cdot \frac{Z^{[l]} - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \beta
\]

\[
A^{[l]} = g^{[l]}(Z^{[l]})
\]

### 2.6 Feed-Forward Networks

Each layer in the encoder and decoder contains a fully connected feedforward network, which is applied to each location separately and identically. Different from the original transformer, we use leaky ReLU instead of ReLU activation to connect the transformation between two linear functions.

\[
FFN(x) = \max(xW_1 + b_1, xW_2 + b_2)W_3 + b_3
\]

As shown above, when our data passes through different intervals, different results can be calculated with different parameters, thereby preventing the activation function from disappearing in areas less than 0.
Leaky ReLU is a variant of ReLU. Although the model here replaces ReLU, it does not indicate that Leaky ReLU is better than ReLU completely. The feature of ReLU in the negative half of 0 can provide sparseness of feature selection. We choose Leaky ReLU in our model, because we hope to retain all the information and try to preserve the characteristics of the data. In actual results, the score of the model using Leaky ReLU is slightly improved.

2.7 Position embedding
Recurrent neural network is a network design based on sequence prediction. The output of each step needs to use the hidden state and output of the previous step to predict, and the time cost will be much larger. Here, in order to improve the convergence speed of the model, we use position embedding to capture the position information. It uses a sine (or cosine) function to record the position information of embedding. Position embedding has the same dimensions as embeddings, so that two embedding can be summed. In our work, we use sine functions:

$$PE_{(pos, 2i)} = \sin(pos/10000^{2id_{model}})$$

‘pos’ is the position of word and ‘i’ is the dimension of embedding. You can see the wavelengths form a geometric progression from $2\pi$ to $10000 \cdot 2\pi$, it allow the model to easily learn to attend by relative positions. And then we add this to embeddings at the input for encoder and decoder stacks.

2.8 Label smoothing
Label smoothing is a very simple but useful technique, and its role has been affirmed in many NLP tasks. Its principle is similar to the idea of preventing overfitting, including some random setting of sample error labels, which all belong to the idea of preventing overfitting.

In the actual training process, we hope that the labels for data fitting are not labels like one-hot. Label smoothing is just like its name. It prevents the model from overfitting by reducing True label value and slightly increasing all other false label values. It converts the original extreme one-hot form into a smoother form:

$$y_{k}^{LS} = y_{k} (1 - \alpha) + \alpha / K$$

3. Implementation
In this part we describe the parameters of the model, the training process and the evaluation method.

3.1 Data
Because there are many model parameters, here we merge the two data and use: the bAbI dialog (Bordes and Weston, 2017) and DSTC2 (Henderson et al., 2014). These two data are the dialogue for restaurant ordering. The test set is divided into two parts, which are divided into out-of-vocabulary (OOV) and common test sets. You will see the test results of the model in the result section.

3.2 Evaluation Methodology
We use three scoring mechanisms to evaluate the results of the model, and evaluate the accuracy, recall, and accuracy of the model.

**Dialog Accuracy:** It is correct only if the generated response is exactly the same as the golden response. A dialog is correct only if each of the responses it generates is correct, which can be considered the task completion rate.

**BLEU:** This is a scoring method derived from machine translation tasks. BLEU score is a relevant measure in task-oriented dialog as there is not a large variance between the generated answers, unlike open domain generation.
**Entity F1**: The entities in each gold System response are selected through a predefined list of entities. This metric evaluates the ability to generate related entities from the provided knowledge base and capture dialog semantics.

### 3.3 Train

Here are some parameters that need to be set. If your dimension of encoder or decoder is 512, so your dimension of feedforward layer is 2048. The number of encoder or decoder blocks can be 6 or 12 as required, and the number of attention heads we know is 8 from previous research. After setting the parameters, you can start training your own model.

### 3.4 Results

Two test sets are used here, one is an OOV with a different data distribution, and the other is a regular test set with the same data distribution as the training set. Therefore, you can view the result from the table.

| Test     | Dial.Acc | BLEU  | Ent.F1   |
|----------|----------|-------|----------|
| Mem2Seq  | 0.436488 | 82.14 | 0.711533 |
| Transformer | 0.486636 | 70.64 | 0.748307205678 |

| OOV      | Dial.Acc | BLEU  | Ent.F1   |
|----------|----------|-------|----------|
| Mem2Seq  | 0.2818   | 84.08 | 0.726013 |
| Transformer | 0.511484 | 72.17 | 0.752003 |

The results show that the transformer has achieved a very high improvement in the accuracy of the dialogue. The F1 score has improved slightly, but the BLEU score is more than 10 points lower than Mem2Seq. However, this is also related to the BLEU scoring mechanism. The BLEU score is calculated from a reference file. This will not calculate the score for synonyms. The converter model has more powerful expression capabilities, so the output results are more diverse.

### 4. Conclusion and Future Works

When considering the model, there are many optional techniques we can choose. For example, Maxout can also achieve the same or better results as leaky ReLU, but Maxout requires specific detailed design and requires greater computational costs. For training efficiency, we choose leaky ReLU.

Some of the modifications we made in Transformer can generally be classified into the two directions: saving data diversity and preventing overfitting. Leaky ReLU activation functions, stacked encoders and decoders, and Position embedding are all to ensure the diversity of model output results. Residual network, batch normalization, and label smoothing are all to prevent overfitting of the model. When these technologies are combined, we can see satisfactory results from this model.

The Transformer model is a popular and powerful model recently. Since its birth, more than ten tasks in the NLP field have achieved good results, especially when applied to problems with large amounts of data and many types of data. It shows a stronger model expression ability than based on recurrent neural networks. Of course, a single transformer will have some shortcomings in capturing position information, but the Bert model in the future can solve this problem by connecting the transformers bidirectionally. In future research directions, we will continue technical research and continuously improve our model in the field of dialogue.

### ACKNOWLEDGMENTS

Our thanks to ACM SIGCHI for allowing us to modify templates they had developed.
REFERENCES

[1] Ashish aswani, Noam hazeer, Niki armar, Jakob Uszkoreit, Llion ones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. arXiv preprint arXiv :1706.03762

[2] Andrea Madotto, Chien-Sheng Wu, Pascale Fung, Mem2Seq: Effectively Incorporating Knowledge Bases into End-to-End Task-Oriented Dialog Systems.arXiv:1804.08217

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv:1607.06450, 2016

[5] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. arXiv preprint arXiv:1601.06733, 2016.

[6] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. CoRR, abs/1406.1078, 2014.

[7] Junyoung Chung, Çaglar Gülcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR, abs/1412.3555, 2014.

[8] Alex Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.

[9] Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. International Conference on Learning Representations, abs/1605.07683.

[10] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. arXiv preprint arXiv:1703.03130, 2017.

[11] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. International Conference on Learning Representations.

[12] Cheongjae Lee, Sangkeun Jung, Seokhwan Kim, and Gary Geunbae Lee. 2009. Example-based dialog modeling for practical multi-domain dialog system. Speech Communication, 51(5):466–484.

[13] Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Query-reduction networks for question answering. International Conference on Learning Representations.