Evaluating the Transformer With Relative Position Representations on the 2017 IWSLT Dataset

Jin Li
University of Chicago
jinli11@uchicago.edu

Abstract

The transformer architecture achieved state of the art results for machine translation without explicitly modeling positional information in its structure. Shaw et al. were able to improve the translation performance by presenting a learned approach to representations of relative positions. We compared both the base transformer and the transformer with relative position representation to the submitted models in the IWSLT 2017 Evaluation Campaign, and achieved improvements of 0.27 BLEU score on Italian to English translations.

1 Introduction

Before the transformer architecture has been introduced, recurrent neural networks (such as long term memory and gated recurrent neural networks) have been the go-to state of the art approaches in neural translation models (Hochreiter and Schmidhuber, 1997). The transformer was the first neural model that relies solely on an attention mechanism, achieving state of the art translation quality with reduced training time (Vaswani et al., 2017). Compared to the former models, the transformer does not explicitly model relative or absolute absolute position information in its structure, but rather adds representations of absolute positions to its inputs. Shaw et al. was able to improve the performance of the transformer model by extending the self-attention mechanism to consider distances between sequence elements (Shaw et al., 2018).

In this paper, we trained the transformer with and without relative position representations on the 2017 IWSLT Ted Talk dataset, a corpus of high quality translations of multiple languages (Cettolo et al., 2018). We trained the model on the Italian to English dataset and evaluated it with BLEU score metric.

2 Background

In this section, we will briefly explain the components of the transformer architecture. Next, we discuss how relative position is incorporated in the transformer, replacing the explicit representations of position information.

2.1 Transformer

The transformer model is a encoder-decoder structure, consisting of stacked encoder and decoder layers. The encoder maps an input sequence $a = (a_1...a_n)$ to a sequence of continuous representations $b = (b_1...b_n)$. The decoder network acts as the language model, using b and an item from the output sequence $c = (c_1...c_m)$ to predict the next output in that sequence. For example, inputting c_{i-1} along with b will be used to predict c_i.

In this architecture, a sentence is split into words, each of which is mapped to a word embedding that is learned along with the model. Next, position encodings based on a sinusoid function of varying frequency are added to the input embeddings. The equations used for positional representations are:

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

$$PE_{(pos,2i+1)} = \cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$

pos is the position of the word in the sentence, and i is an index from the word embedding.

The authors of the transformer state that these positional representations help the model generalize to unseen sequence lengths unseen during training (Vaswani et al., 2017).

Each layer of the encoder network consists of two sublayers, a multi-head attention followed by a feed-forward layer (two linear transformations with ReLU activation in between), both of which employs residual connections from the previous

layers after layer normalization and dropout. The residual connections help information propagate position information to higher layers. The output of the encoder network is transformed and then fed into each layer of the decoder network. Each layer of the decoder consists of 3 sub-layers: multi-head attention followed by encoder-decoder attention, followed by a position-wise feed-forward layer. Just like the encoder structure, each sub-layer employs residual connections followed by layer normalization and dropout.

The output of the decoder structure is then fed through a linear transformation, then through a softmax to predict a probability distribution over the target words.

2.2 Self-Attention

Each self-attention sub-layer has h attention heads. Each attention head takes in an input sequence $X = (x_1, ..., x_n)$ of n elements, where $x_i \in R^{d_x}$, and produces an output sequence $Z = (z_1, ..., z_n)$ where $z_i \in R^{d_z}$.

Each output z_i is computed by summing all input projections (the values), weighted by some attention function (calculated through the queries and keys).

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V) \tag{1}$$

The attention weight is computed with a softmax normalization:

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{n} \exp e_{ik}} \tag{2}$$

$$e_{ij} = \frac{(x_i W^Q)(x_j W^K)^T}{\sqrt{d_z}} \tag{3}$$

$W^Q, W^K, W^V \in R^{d_x \times d_z}$ are parameters trained throughout the model that are not shared throughout the layers or attention heads in the encoder-decoder architecture. The outputs of each head are then concatenated together and multiplied by a weight matrix W^O to produce the output.

For the decoder architecture, the outputs of the top encoder is transformed into attention vectors K and V, which act as the key and value for the middle decoder layer. There is also masking in the decoder stack to ensure that predictions for position i can only depend on known outputs at positions less than i.

2.3 Relation Aware Self-Attention

Shaw et al. proposed an extension to self-attention by learning pairwise relationships between input elements as an alternative to the explicit sinosoid positional representation (Shaw et al., 2018). They represent the edge between input elements x_i and x_j by vectors $a_{ij}^V, a_{ij}^K \in R^{d_z}$. These representations are shared across attention heads to reduce computations. Modifying the previous attention equations (2) (3), they proposed:

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V + a_{ij}^V) \tag{4}$$

$$e_{ij} = \frac{(x_i W^Q)(x_j W^K + a_{ij}^K)^T}{\sqrt{d_z}} \tag{5}$$

These changes allows edge information to propagate to the sub-layer output. In addition, the authors clipped the maximum relative distance considered to a maximum absolute value of k, as they claimed relative position information is not useful beyond a certain distance. They also state clipping allows the model to generalize to sequence lengths not seen during the training (Shaw et al., 2018).

$$a_{ij}^K = w_{clip(j-i,k}^K$$

$$a_{ij}^V = w_{clip(j-i,k}^V$$

$$clip(x, k) = max(-k, min(k, v))$$

Then relative position representations $w^K = (w_{-k}^K ... w_k^K)$ and $w^V = (w_{-k}^V ... w_k^V)$ are learned, where $w_i^K, w_i^V \in R^{d_a}$.

3 Experimental Setup

In this paper, we used the Italian and English translations of the 2017 IWSLT TED talk corpus, which contains approximately 250,000 sentences. The dataset is already split into train, development, and a test set. We preprocessed the dataset by removing all lines that start with $<$ and then using the SpaCy library to parse through sentences to tokenize the words.

We used the tensor2tensor library to help train and evaluate the model (Vaswani et al., 2018). To speed up training, sentences with similar lengths are batched together (with maximum input and output tokens per batch of 4096) and padded appropriately.

Models	EN-IT	IT-EN
FBK	29.60	34.24
GTCT	32.84	37.84
KIT	32.04	36.30
KYOTO	30.79	34.73
UDSDFKI	29.62	33.77
Trans	32.81	37.82
Rel Trans	33.11	37.88

Table 1: BLEU Score Model Comparison

k	IT-EN
0	16.81
1	32.54
2	32.89
4	32.76
16	33.11
64	32.94

Table 2: Experimental Results for Varying Clipping Distance

For the base transformer model, we used 6 encoders and decoder layers, $d_x = 512$, $d_z = 64$, 8 attention heads, 1024 feed forward inner-layer dimensions, and $P_{dropout} = 0.1$. For the transformer model with relative position encoding, we used a clipping distance of $k = 16$ and used unique edge representations per layer, but shared them across heads. We trained for 50,000 steps on 4 Nvidia K80 GPUs.

We used the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$ (Kingma and Ba, 2015). We used the same learning rate function proposed by Vaswani et al:

$$lr = d_{model}^{-0.5} \cdot min(step_n^{-0.5}, step_n \cdot warmup^{-1.5}) \quad (6)$$

The warmup steps was set to 4000.

During training, the labels were smoothed with value $\epsilon_{ls} = 0.1$, which improves accuracy and BLEU score while worsening perplexity (Vaswani et al., 2017). During evaluation, we used beam search with a beam size of 4 and length penalty $\alpha = 0.6$ (Wu et al., 2016).

4 Results and Analysis

We compare our BLEU results of Italian and English translations with the submitted models in the IWSLT 2017 Evaluation Campaign (Cettolo et al., 2018). We observed a performance improvement of the relative base transformer by 0.27 BLEU in English to Italian translation and 0.04 BLEU in Italian to English translation relative to the top submissions.

We also varied the different clipping distances k. We found that when $k \geq 2$, there is no noticeable differences in BLEU scores. This indicates that beyond a certain threshold, relative positional representations become less relevant to the model. When $k = 0$, we have that the BLEU score dropped dramatically, as all edges end up with the same weights, causing relative positional representation to misdirect the model.

5 Conclusions

In this paper, we compared the transformer model with and without relative position representations to the other translation models in the IWSLT 2017 Evaluation Campaign. We found that the transformer with relative position outperformed all the other models.

For future work, we plan to train the model over the other languages in the TED Talk dataset and compare the BLEU scores. We are also interested in finding a way to use relative position while reducing the memory complexity.

Acknowledgments

We are grateful for Kevin Gimpel for providing fruitful feedback and suggestions for this paper.

References

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuhito Sudoh, Koichi Yoshino, and Christian Federmann. 2018. Overview of the iwslt 2017 evaluation campaign.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *NAACL-HLT*.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Franois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Lukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob

Uszkoreit. 2018. Tensor2tensor for neural machine translation. In *AMTA*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.