Recurrent Positional Embedding for Neural Machine Translation

Kehai Chen, Rui Wang, Masao Utiyama, and Eiichiro Sumita
National Institute of Information and Communications Technology (NICT), Kyoto, Japan
{kchen, wangrui, mutiyama, eiichiro.sumita}@nict.go.jp

Abstract

In the Transformer network architecture, positional embeddings are used to encode order dependencies into the input representation. However, this input representation only involves static order dependencies based on discrete numerical information, that is, are independent of word content. To address this issue, this work proposes a recurrent positional embedding approach based on word vector. In this approach, these recurrent positional embeddings are learned by a recurrent neural network, encoding word content-based order dependencies into the input representation. They are then integrated into the existing multi-head self-attention model as independent heads or part of each head. The experimental results revealed that the proposed approach improved translation performance over that of the state-of-the-art Transformer baseline in WMT’14 English-to-German and NIST Chinese-to-English translation tasks.

1 Introduction

Transformer translation systems (Vaswani et al., 2017), without recurrent and convolutional neural networks, rely on a positional embedding (PE) approach to encode order information into the input representation. PE is typically learned based on the position index of each word and is added to corresponding word embedding. This allows the Transformer to encode order dependencies between words in addition to the words themselves. Finally, the Transformer uses these combined vectors as the input to self-attention networks (SANs), achieving state-of-the-art translation performance with several language pairs (Vaswani et al., 2017; Dou et al., 2018; Zhang et al., 2018a; Marie et al., 2018, 2019).

In spite of their success, the input representation only involves static order dependencies based on discrete numerical information. That is, any word in the entire vocabulary has the same PE on the same position index. As a result, the dependencies encoded by the original PEs are independent of word content, which may further hinder the improvement of translation capacity. Recently, Chen et al. (2018) and Hao et al. (2019) introduced the additional source representation learned by an RNN-based encoder into Transformer to alleviate this issue, and reported improvements on the WMT’14 English-to-German translation task.

Inspired by their works (Chen et al., 2018; Hao et al., 2019), we propose an simple and efficient recurrent positional embedding approach to capture order dependencies based on word content in a sentence, thus learning a more effective sentence representation for the Transformer. In addition, we designed two simple multi-head self-attentions to introduce these learned RPEs and original input representation into the existing Transformer model for enhancing the sentence representation of Transformer. Experimental results on WMT’14 English-to-German and NIST Chinese-to-English translation tasks show that our models significantly improved translation performance over a strong Transformer baseline.

2 Background

2.1 Input Representation

In the Transformer network architecture (Vaswani et al., 2017), given a sentence of length J, the positional embedding of each word is firstly computed based on its position:

$$pe_{(j,2i)} = \sin\left(j/10000^{2i/d_{model}}\right),$$

$$pe_{(j,2i+1)} = \cos\left(j/10000^{2i/d_{model}}\right),$$

(1)
where \( j \) is the word’s numerical position index in the sentence and \( i \) is the dimension of position index. The word embedding \( x_j \) is then added with \( \text{pe}_j \) to get a combined embedding \( v_j \):

\[
v_j = x_j + \text{pe}_j. \tag{2}
\]

As a result, there is a sequence of vectors \( \{v_1, \ldots, v_J\} \) as the input to the encoder or decoder of the Transformer to learn the source or target sentence representations.

### 2.2 Self-Attention Mechanism

Following the input layer, the self-attention layer is used to learn the sentence representation for the Transformer (Vaswani et al., 2017). These combined vectors \( \{v_1, \ldots, v_J\} \) are then packed into a query matrix \( Q \), their keys and values matrices \( K \) and \( V \). The output of the self-attention layer can be computed by Eq.(3):

\[
\text{Att}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_{\text{model}}}})V. \tag{3}
\]

Moreover, self-attention can be further refined as multi-head self-attention to jointly attend to the information from different representation subspaces at different positions. Specifically, \( Q, K, \) and \( V \) are linearly projected \( H \) times with different, learned linear projections to \( d_k \) and \( d_v \) dimensions, respectively. On each of these projected versions of queries, keys, and values, the attention function is performed in parallel, yielding \( d_v \)-dimensional output values. Take a single head as an example, the output of the \( h \)-th head \( O_h \) is computed by Eq.(4):

\[
O_h = \text{Att}(QW_h^Q, KW_h^K, VW_h^V), \tag{4}
\]

where the parameter matrices are \( W_h^Q \in \mathbb{R}^{d_{\text{model}} \times d_k} \), \( W_h^K \in \mathbb{R}^{d_{\text{model}} \times d_k} \), and \( W_h^V \in \mathbb{R}^{d_{\text{model}} \times d_v} \). For example, if there are \( H=8 \) heads and \( d_{\text{model}}=512 \), \( d_k=d_v=512/8=64 \). Finally, the outputs of \( H \) heads are concatenated to serve as the sentence representation \( S \):

\[
S = \text{Concat}(O_1, \ldots, O_{H-1}, O_H)W^O, \tag{5}
\]

where \( W^O \in \mathbb{R}^{Hd_v \times d_{\text{model}}} \) is a parameter matrix.

### 3 Recurrent Positional Embedding

We propose a recurrent positional embedding approach based on part of word embedding instead of numerical indices of words to capture order dependencies between words in a sentence. Specifically, the embedding of each word \( x_j \) is divided into two parts \( x_j^p \) and \( x_j^r \), and their dimensions are \( d_p \) and \( d_r \) (\( d_{\text{model}}=d_p+d_r \)), respectively. As a result, the sequence of word vectors \( \{x_1, \ldots, x_J\} \) are spited into \( \{x_j^p, \ldots, x_j^r\} \) and \( \{x_j^1, \ldots, x_j^J\} \). An RNN with an nonlinear projection layer is then designed to learn its recurrent state \( r_j \) for each word over \( \{x_j^1, \ldots, x_j^J\} \):

\[
r_j = \text{Tanh}(f_{\text{RNN}}(x_j^r, r_{j-1})W^r + b^r), \tag{6}
\]

where \( W^r \in \mathbb{R}^{d_r \times d_r} \) is a parameter matrix and \( b^r \in \mathbb{R}^{d_r} \) is a bias item.\(^1\) Note that the \( x_j^r \) is derived from part of the word embedding \( x_j \). Finally, there is a sequence \( R=\{r_1, \ldots, r_J\} \) called as recurrent positional embeddings (RPEs). In this work, a bidirectional RNN and a forward RNN (Bahdanau et al., 2015) are used to learn source RPEs and target RPEs, respectively. Noth that the RNN is also replaced by other neural networks for learning dependency information, such as GRU (Cho et al., 2014), and SRU (Li et al., 2018a).

In addition, other sub-sequence \( \{x_j^1, \ldots, x_j^J\} \) is used to gain the reduced dimension input representation \( P=\{p_1, \ldots, p_J\} \) according to the Section 2.1. Both of \( R \) and \( P \) will be together as the input to the encoder (or decoder) to learn a more effective source (or target) representation for the Transformer.

### 4 Neural Machine Translation with RPE

To make use of these learned RPEs, we propose two simple methods: RPE head (RPEHead) self-attention and mixed positional representation head (MPRHead) self-attention. Both of RPEHead and MPRHead can utilize RPEs to learn sentence representation for the Transformer.

#### 4.1 RPEHead Self-Attention

For the RPEHead self-Attention, these learned RPEs are integrated into multi-head self-attention (Fig 1(a)) as several independent heads to learn the sentence representation, as shown in Fig 1(b).

First, we concatenate \( P=\{p_1, \ldots, p_J\} \) and \( R=\{r_1, \ldots, r_J\} \) as a new combined input representation \( \mathcal{T} \). To perform the attention function in Eq.(3) over the combined input representation \( \mathcal{T} \), \( d_r \) and \( d_p \) are set to \( t^*d_{\text{model}}/H \)

\(^1\)The \( r_0 \) is set as a zero vector.
and \(d_{\text{model}}\cdot d_{r}\) in the process of learning RPEs, respectively. This guarantees that there are two types of heads: one type of head only contains RPE and other type of head only contains the original reduced dimension input representation. Second, the \(T\) is mapped to a new query matrix \(Q^T\), and their keys and values matrices \(K^T\) and \(V^T\). According to Eq.(4), the output of each head is computed by Eq.(7):

\[
O_h^T = \text{Att}(Q^T W_h^Q, K^T W_h^K, V^T W_h^V). \tag{7}
\]

Therefore, the final sentence representation \(S^T\) is formally represented as:

\[
S^T = \text{Concat}(O_1^T, \ldots, O_{H-1}^T, O_H^T) W^O. \tag{8}
\]

### 4.2 MPRHead Self-Attention

Compared with the RPEHead model, the MPRHead model applies the multi-head mechanism to the RPEs. In other words, RPEs are encoded into the sentence representation from different vector sub-spaces, as shown in Fig 1(c).

To this end, each vector of \(P\) is divided into the \(H\) heads \(\{p_j^1, \ldots, p_j^H\}\). Similarly, each vector of \(R\) is divided into the \(H\) heads \(\{r_j^1, \ldots, r_j^H\}\). The corresponding heads for \(p_j\) and \(r_j\) are then concatenated as a combined sequence \(p_{rj}\), in turn:

\[
p_{rj} = \{p_j^1 : r_j^1, \ldots, p_j^H : r_j^H\}, \tag{9}
\]

where “:" is the concatenation operation. All heads in \(p_{rj}\) are further concatenated as a mixed embedding \(m_j \in \mathbb{R}^{d_{\text{model}}}\) in turn. As a result, there is a new sequence of mixed embeddings:

\[
M = \{m_1, \ldots, m_j\}. \tag{10}
\]

The \(M\) is mapped to a query matrix \(Q^M\), and their keys and values matrices \(K^M\) and \(V^M\).

According to Eq.(4) and Eq.(5), the final sentence representation \(S^M\) is represented as:

\[
O_h^M = \text{Att}(Q_h^M W_h^Q, K_h^K W_h^K, V_h^V W_h^V),
\]

\[
S^M = \text{Concat}(O_1^M, \ldots, O_{H-1}^M, O_H^M) W^O. \tag{11}
\]

Note that for both RPEHead and MPRHead models, the RPEs will be jointly learned with the existing Transformer architecture.

## 5 Experiments

### 5.1 Experimental Setup

The proposed methods were evaluated on the WMT’14 English to German (EN-DE) and NIST Chinese-to-English (ZH-EN) translation tasks. The ZH-EN training set includes 1.28 million bilingual sentence pairs from the LDC corpora, where the NIST06 and the NIST02/NIST03/NIST04 data sets were used as the development and test sets, respectively. The EN-DE training set includes 4.43 million bilingual sentence pairs of the WMT’14 corpora, where the newstest2013 and newstest2014 data sets were used as the development and test sets, respectively.

The BPE (Sennrich et al., 2016) was adopted and the vocabulary size was set as 32K. The dimension of all input and output layers was set to 512, and that of the inner feedforward neural network layer was set to 2048. The total heads of all multi-head modules were set to 8 in both encoder and decoder layers. In each training batch, there was a set of sentence pairs containing approximately 4096*4 source tokens and 4096*4 target tokens. For the other setting not mentioned, we followed the setting in Vaswani et al. (2017).

Baseline systems included a vanilla Transformer (Vaswani et al., 2017), Relative PEs (Shaw et al., 2018), and directional SAN (DiSAN) (Shen...
### System Architecture

**Existing NMT systems**

<table>
<thead>
<tr>
<th>System</th>
<th>Architecture</th>
<th>newstest2014</th>
<th>#Param</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vaswani et al. (2017)</td>
<td>Transformer (base)</td>
<td>27.3</td>
<td>65M</td>
</tr>
<tr>
<td></td>
<td>Transformer (big)</td>
<td>28.4</td>
<td>213M</td>
</tr>
<tr>
<td>Chen et al. (2018)</td>
<td>RNMT+SAN</td>
<td>28.39</td>
<td>378.9M</td>
</tr>
<tr>
<td>Hao et al. (2019)</td>
<td>Transformer (base)+BiARN</td>
<td>28.21</td>
<td>97.4M</td>
</tr>
<tr>
<td></td>
<td>Transformer (big)+BiARN</td>
<td>28.98</td>
<td>323.5M</td>
</tr>
</tbody>
</table>

**Our NMT systems**

<table>
<thead>
<tr>
<th>This work</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer (base)</td>
<td>27.25</td>
<td>97.35M</td>
<td></td>
</tr>
<tr>
<td>+Relative PE</td>
<td>27.60</td>
<td>97.42M</td>
<td></td>
</tr>
<tr>
<td>+DiSAN</td>
<td>27.66</td>
<td>97.39M</td>
<td></td>
</tr>
<tr>
<td>+RPEHead</td>
<td>28.11*</td>
<td>97.84M</td>
<td></td>
</tr>
<tr>
<td>+MPRHead</td>
<td>28.35*</td>
<td>97.72M</td>
<td></td>
</tr>
<tr>
<td>Transformer (big)</td>
<td>28.22</td>
<td>272.6M</td>
<td></td>
</tr>
<tr>
<td>+MPRHead</td>
<td>29.11*</td>
<td>289.1M</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Results for EN-DE translation task. The mark “*” after scores indicates that the model was significantly better than the baseline Transformer (base or big) at the significance level \( p < 0.01 \) (Collins et al., 2005).

![Figure 2: The BLEU scores on the different \( d_r \).](image)

5.2 **Effect of RPEs**

In this work, we extracted \( d_r \) dimensions of each word vector to learn recurrent embeddings. To explore the relation between \( d_r \) and translation performance, Figure 2 shows the translation performance on the different \( d_r \). For +RPEHead (or +MPRHead), with the increasing in the dimension of \( d_r \), BLEU scores are gradually increasing, but BLEU scores begin to decrease when \( d_r \) is more than 320 (or 256). In particular, +RPEHead and +MPRHead achieve the highest BLEU scores at \( d_r=320 \) and \( d_r=256 \), respectively. This means that the original partial input representation and our RPE can complement each other to improve translation performance.

5.3 **Main Results**

According to the results of Fig 2, \( d_r \) of RPEHead is set to 320 and \( d_r \) of MPRHead is set to 256. The main translation results are shown in Table 1.

1) For the proposed methods, both RPEHead (base) and MPRHead (base) outperformed Transformer (base), especially are better than +RPE and +DiSAN. This indicates that the learned RPEs are beneficial for the Transformer system.

2) Moreover, +MPRHead (base) performed better than RPEHead/RPE (base). The reason may be that adding RPEs into variant heads can encode order dependencies based word vectors from variant vector sub-spaces, which is one of the advantages of the multi-head mechanism. In particular, +MPRHead (base/big) is slightly better than Transformer (base/big)+BiARN. This denotes that the RPEs can more effective improve the performance of Transformer.

3) MPRHead (big) was superior to Transformer (big) significantly. MPRHead (base) achieved a comparable performance compared to Transformer (big) which contains approximately three times parameters, indicating that the proposed RPE is efficient.

In addition, Table 2 shows that the proposed models also gave similar improvements over the baseline system and the compared methods on the NIST ZH-EN task. These results indicate that our
approach is a universal method for improving the translation of other language pairs.

5.4 Ablation Experiments
To further explore the effect of position information, PEs and RPEs were applied on the encoder (Enc) and decoder (Dec) side, respectively. Table 3 shows the results of ablation experiments on the EN-DE newstest2014 test set:

<table>
<thead>
<tr>
<th>System</th>
<th>Architecture</th>
<th>NIST02</th>
<th>NIST03</th>
<th>NIST04</th>
<th>#Param</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing NMT systems</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Liu et al. (2019)</td>
<td>Share-private Embeddings</td>
<td>43.73</td>
<td>41.99</td>
<td>44.53</td>
<td>N/A</td>
</tr>
<tr>
<td>Chen et al. (2019)</td>
<td>Reordering Embeddings</td>
<td>47.54</td>
<td>46.56</td>
<td>47.27</td>
<td>78.3M</td>
</tr>
<tr>
<td>Our NMT systems</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transformer (base)</td>
<td></td>
<td>46.62</td>
<td>45.90</td>
<td>45.95</td>
<td>77.94M</td>
</tr>
<tr>
<td>+Relative PEs (Shaw et al., 2018)</td>
<td></td>
<td>46.91</td>
<td>46.47</td>
<td>45.96</td>
<td>77.01M</td>
</tr>
<tr>
<td>+RPEHead</td>
<td></td>
<td>47.06</td>
<td>46.21</td>
<td>46.32</td>
<td>78.04M</td>
</tr>
<tr>
<td>+MPRHead</td>
<td></td>
<td>47.69*</td>
<td>47.27*</td>
<td>47.31*</td>
<td>78.04M</td>
</tr>
<tr>
<td>Transformer (big)</td>
<td></td>
<td>47.88</td>
<td>47.58</td>
<td>47.37</td>
<td>243.7M</td>
</tr>
<tr>
<td>+RPEHead</td>
<td></td>
<td>48.73*</td>
<td>48.61*</td>
<td>48.21*</td>
<td>245.5M</td>
</tr>
</tbody>
</table>

Table 2: Results for ZH-EN translation Task. The mark “*” after scores indicates that the model was significantly better than the baseline Transformer (base or big) at the significance level $p < 0.01$ (Collins et al., 2005).

1) For Enc/Dec/Enc&Dec, the proposed RPEHead/MPRHead outperformed the Transformer in corresponding ablation settings, indicating that our methods worked well on both the encoder and decoder.

2) For PE/RPEs, performance became slightly lower when PE was removed from the decoder. However, when PE was removed from the encoder, the translation performance drastically decreased. We think that the source sentence representation is more sensitive to position information than the target sentence representation. It is possible that each hidden state in the decoder takes the previous hidden state into consideration, which would be somewhat similar to the proposed RPEs. In contrast, the encoder would not be expected to contain this mechanism. Therefore, the position information would be more important in the encoder than the decoder.

3) The training speeds of the proposed RPEHead (base) and MPRHead (base) were just slightly slower than those of the vanilla Transformer (base), because we nearly did not introduce additional (less than 2%) parameters.

6 Conclusion and Future Work
In this paper, we presented a recurrent embedding to capture the order dependencies in a sentence. Empirical results show that this method can improve the performance of NMT efficiently. In future work, we plan to extend this method to unsupervised NMT (Sun et al., 2019) and other natural language processing tasks, such as dependency parsing (Li et al., 2018b) and reading comprehension (Zhang et al., 2018b).

Acknowledgments
We are grateful to the anonymous reviewers and the area chair for their insightful comments and suggestions. This work was partially conducted under the program “Research and Development of Enhanced Multilingual and Multipurpose Speech Translation Systems” of the Ministry of Internal Affairs and Communications (MIC), Japan. Masao Utiyama is partly supported by JSPS KAKENHI Grant Number 19H05660. Rui Wang was partially supported by JSPS grant-in-aid for early-career scientists (19K20354): “Unsupervised Neural Machine Translation in Universal Scenarios” and NICT tenure-track researcher startup fund “Toward Intelligent Machine Translation”.

1365
References


