Analysis of Positional Encodings for Neural Machine Translation

Jan Rosendahl, Viet Anh Khoa Tran
Weiyue Wang and Hermann Ney

Human Language Technology and Pattern Recognition Group
RWTH Aachen University, Germany
{rosendahl,vtran,wwang}@i6.informatik.rwth-aachen.de

Abstract

In this work we analyze and compare the behavior of the Transformer architecture when using different positional encoding methods. While absolute and relative positional encoding perform equally strong overall, we show that relative positional encoding is vastly superior (4.4% to 11.9% BLEU) when translating a sentence that is longer than any observed training sentence. We further propose and analyze variations of relative positional encoding and observe that the number of trainable parameters can be reduced without a performance loss, by using fixed encoding vectors or by removing some of the positional encoding vectors.

1. Introduction

In this work we analyze the performances of different methods to encode positional information in the Transformer architecture [1]. The Transformer architecture relies on self-attention layers to handle sequences of varying length. However, self-attention layers themselves provide little positional information. Keys and values in self-attention are treated as sets without an ordering and reordering the queries simply results in a reordered output. While the decoder may get some positional information from the left-to-right masking scheme, the encoder does not have access to any positional information. To solve this problem, Transformer models rely on explicit positional information, typically in form of a (absolute) positional encoding vector that is added to each embedded word.

Recent works suggest to use relative positional encodings instead. This can be done by modifying the self-attention layer, which considers each pair of tokens from the input sentence, to include a new vector that encodes the distances of the tokens [2, 3].

In this work we empirically analyze different positional encoding schemes for the use of machine translation. We compare absolute and relative positional encodings with a strong focus on their behaviour for different sentence lengths. Furthermore we propose and analyze different variations and combinations of absolute and relative positional encoding.

2. Related Work

The Transformer [1] model consists of stacked encoder and decoder layers. Since the model uses self-attention layers to handle sequences, information about the sequence order has to be included explicitly into the model. Vaswani et al. [1] introduced the Transformer using absolute positional encodings based on sinusoid functions. Absolute positional encodings are also used in convolutional systems [4]. The positional encodings are added to the input embeddings at the bottom of the encoder and decoder stacks.

Shaw et al. [2] propose an alternative approach, in which the self-attention mechanism is extended to efficiently account for representations of relative positions or distances between tokens. The relative positional information is included by adding vectors, which represent the pairwise relationship between the current position and other positions, to the projected keys and values. Our experiments are mainly based on this technique.

Chen et al. [5] use bidirectional long short-term memory (LSTM) [6] cells in a recurrent neural networks (RNN) and combine it with a self-attentive architecture, showing that recurrence can contribute to their strongest model. The proposed cascaded model involves the fine-tuning of self-attention layers stacked on pre-trained frozen LSTM layers. With this structure, the positional encoding technique in the standard Transformer model is no longer required, as the LSTM-RNN layers can embed contextual information of unlimited length. Inspired by this idea, we also try to apply LSTM layers for the purpose of positional encoding (for both encoders and decoders) and compare them with other positional encoding techniques.

Dai et al. [3] also claim that absolute positional encoding does not contain information to distinguish positional differences, which can lead to a performance loss. To avoid this the authors offer a different derivation than [2] and test their method on the task of language modeling.

3. Positional Encoding

The Transformer architecture is a sequence-to-sequence architecture, consisting of stacked encoder and decoder layers. To generate sequence representations the Transformer em-

ploys self-attention.

An encoder layer consists of a self-attention sub-layer followed by a feedforward sub-layer. Similarly, a decoder layer consists of a self-attention sub-layer, followed by an encoder-decoder attention sub-layer, which is followed by a feedforward sub-layer. The output of each sub-layer for a given time step is of dimension d_{model}. However, the self-attention mechanism ignores the sequence order, which is why positional encodings are used in the Transformer architecture.

3.1. Absolute Positional Encoding

In the original Transformer architecture, an absolute positional encoding vector PE_j is added to each embedded source and target word to denote its position j. The idea is to use sinusoids of different wavelengths to encode different positions. For the position $j \in \{1, \dots, J\}$ in a sequence of length J, the absolute positional encoding vector $\text{PE}_j \in \mathbb{R}^{d_{\text{model}}}$ is defined by

$$\text{PE}_{\text{j},2\beta} = \sin(\text{j}/10000^{2\beta/d_{\text{model}}}), \tag{1}$$

$$\text{PE}_{\text{j},2\beta+1} = \cos(\text{j}/10000^{2\beta/d_{\text{model}}}) \tag{2}$$

where $\beta = 1, \dots, \lfloor \frac{d_{\text{model}}}{2} \rfloor$. Vaswani et al. [1] propose that sinusoidal positional embeddings perform equally well as learned positional encodings and hypothesize that the former are capable of generalizing to longer sequences.

3.2. Relative Positional Encoding

Absolute positional encoding does not explicitly consider distance relationships. In the framework of relative positional encoding [2] a trainable distance encoding is added to every self-attention layer.

Consider the n-th head of a self-attention layer. First, a vector $\gamma^K_{j'-j} \in \mathbb{R}^{d_k}$ is added to the projected keys when computing the energy:

$$e_{j,j'} = \frac{(h_{j'} W^Q_n)(h_j W^K_n + \boldsymbol{\gamma^K_{j'-j}})^T}{\sqrt{d_k}}, \tag{3}$$

where $W^K_n,\ W^Q_n \in \mathbb{R}^{d_{\text{model}} \times d_k}$ are the projection matrices of the n-th attention head, $\{h_j \mid j = 1, \dots, J\}$ is the set of keys and $h_{j'}$ is the j'-th query of the self-attention mechanism. Second, the projected values are shifted by $\gamma^V_{j'-j} \in \mathbb{R}^{d_v}$ when computing the weighted sum:

$$c_j = \sum_{j'=1}^{J} \alpha_{j,j'} (h_j W^V_n + \boldsymbol{\gamma^V_{j'-j}}), \tag{4}$$

where $\alpha_{j,j'} = \frac{\exp e_{j,j'}}{\sum_{\tilde{j}=1}^{J} \exp e_{j,\tilde{j}}}$ and $W^V_n \in \mathbb{R}^{d_{\text{model}} \times d_v}$ is a projection matrix. Using the two different distance terms $\gamma^K_{j'-j}$ and $\gamma^V_{j'-j}$ for the projected keys and the projected values allows to project these keys and queries onto different dimensions, i.e. $d_k \neq d_v$.

The distance terms are based on trainable parameters $r^K_{-\tau}, \dots r^K_{\tau} \in \mathbb{R}^{d_k}$ and $r^V_{-\tau}, \dots r^V_{\tau} \in \mathbb{R}^{d_v}$ respectively, where the hyperparameter τ describes the *clipping distance*. A distance term $\gamma_{j'-j}$, which encodes a distance that exceeds τ, will fall back to the value of γ_τ (or $\gamma_{-\tau}$, if $j' < j$):

$$\gamma^K_{j'-j} = r^K_{\text{clip}_\tau(j'-j)} \tag{5}$$

$$\gamma^V_{j'-j} = r^V_{\text{clip}_\tau(j'-j)} \tag{6}$$

$$\text{clip}_\tau(x) = \max(-\tau, \min(\tau, x)). \tag{7}$$

In order to facilitate computation, the distance terms are shared across attention heads of a multi-head attention layer, but not across self-attention sub-layers or Transformer layers.

A variation of relative positional encoding is to use the sinusoids from the absolute positional encoding approach instead of trained parameters (*relative sinusoidal positional encoding*):

$$\gamma^K_{j'-j} = [\text{PE}_{\text{clip}_\tau(j'-j),1}, \dots, \text{PE}_{\text{clip}_\tau(j'-j),d_k}] \tag{8}$$

$$\gamma^V_{j'-j} = [\text{PE}_{\text{clip}_\tau(j'-j),1}, \dots, \text{PE}_{\text{clip}_\tau(j'-j),d_v}]. \tag{9}$$

We also investigate a variation of relative positional encoding where the distance terms γ^V are omitted.

3.3. LSTM-based Positional Encoding

Since the Transformer employs self-attention layers for sequence handling it needs explicit positional information to avoid bag-of-word modeling. Bahdanau et al. [7] handle this problem by using LSTM layers that incorporate the positional information implicitly. This *LSTM-based positional encoding* can be used for the Transformer architecture. In the encoder, the embeddings are passed into a BiLSTM layer before being passed into the first encoder layer. Similarly, we introduce a single LSTM layer between the embedding layer and the first decoder layer. Note that this approach significantly increases the number of parameters (by 20% in our case), increases the model depth and removes parts of the parallelism for the Transformer architecture.

This addition of two LSTM layers is similar to the idea of a LSTM-Transformer Cascaded Encoder [5], i.e. to pass the source sequence first through an LSTM-based encoder before feeding the output representations to a regular Transformer encoder. Differently than Chen et al. [5] we only intend to retain the positional information, hence we use a single BiLSTM layer in the encoder. Furthermore Chen et al. do not consider the LSTM layers in the decoder.

4. Experimental Results

4.1. Experimental Setup

We use a 6 layer Transformer model that closely matches the 'base' configuration that was introduced together with the Transformer architecture [1]. In particular we use 8 attention heads and layer sizes of $d_{\text{model}} = 512$ and $d_{\text{ff}} = 2048$

Positional Encoding	De→En newstest2018		Zh→En newstest2019	
	BLEU	TER	BLEU	TER
Absolute	40.4	46.7	24.0	64.0
Relative	40.4	46.9	25.0	62.9

Table 1: Comparison of absolute and relative positional encoding on the WMT De→En and Zh→En news translation task.

for the feed-forward layer. For all experiments with relative positional encoding we use a clipping distance of $\tau = 16$ as suggested by Shaw et al. [2]. Source embeddings, target embeddings and target projection matrices are not shared and a warm-up period with constant or increasing learning rate was not used. All models are trained using the Adam optimizer [8] with a learning rate of 0.0003. We use a batch size of 4700 tokens and save a model checkpoint 8 times per epoch. The learning rate is scaled by a factor of 0.9 if no improvement in perplexity on the development set has been observed for 6 consecutive evaluation checkpoints. All BLEU scores reported are calculated on case-sensitive and tokenized data. TER scores are calclated using TERCom.

We train our models on the data from the De→En and the Zh→En news translation task of WMT 2019. For the De→En task we train on CommonCrawl, Europarl, News-Commentary and Rapid summing up to a total of 6M lines with `newstest2015` as development set. During preprocessing we apply byte pair encoding [9] with 50k joint merge operations. Our Zh→En systems are trained on all WMT2019 parallel data, namely NewsCommentary, the UN and CWMT corpus totaling 13.7M lines. As development set the concatenation of `newsdev2017` and `newstest2017` is used. All Zh→En data is preprocessed using the unigram language model segmentation algorithm [10] with separate vocabularies of size 32k. If not stated otherwise, all sequences longer than 100 tokens are ignored during training which make about 0.4% (De→En) and 0.2% (Zh→En) of the training data. Note that throughout the paper we determine the length of a sentence by counting the tokens on tokenized, subworded text.

Our results on Zh→En and De→En are given in Table 1. While both positional encoding schemes perform equally on the De→En task , we see a clear improvement of 1.0% BLEU for the case of Zh→En. Note that during training we discard sentence pairs if source or target exceeds 100 tokens. This is not a problem for `newstest2018` of the De→En task since no such sentence pair occurs, however for Zh→En out of the 2000 sentence pairs from `newstest2019` about 1.4% of the source sentences and 4.2% of the target sentences are longer than 100 tokens. In the following we take a closer look at the performance of different positional encoding schemes for sequences of various length.

4.2. Sequence Length Analysis

In order to get a bigger test set that provides a higher variety of sequence lengths we consider the long-standing De→En task and concatenate all old test sets `newtest2008-2018` with the exception of `newstest2015` which is used as the development set. This results in the test set `newstest_all` consisting of 28k sentence pairs. We partition this test set in 5 groups based on their source sequence length, such that each group gathers sentence pairs of similar length. Formally we define:

$$G_{\alpha:\beta} := \left\{(f_1^J, e_1^I) \in \texttt{newstest\_all} \mid \alpha \leq J \leq \beta\right\}$$

where f_1^J is the source sentence of length J and e_1^I the corresponding target sentence of length I. In the rest of this work we group all sentence pairs with a source length difference of 25, e.g. all source sentences between 0 and 25 in $G_{0:25}$.

In Table 2 Lines 1-3 we report the performance of the three positional encoding schemes presented in Section 3 on $G_{0:25}, \ldots, G_{101:\infty}$ as well as the size of the resulting groups. Note that in this section the only relative positional encoding considered is the one proposed by [2] with trainable γ^K and γ^V. See Line 1 for a baseline using absolute positional encoding which is the default setting of many Transformer implementations. Note that the performance is very similar for different length source sentences up to 75 tokens ($G_{0:25}$ to $G_{51:75}$) and drops about 2% BLEU when source sentences of length 76-100 are considered. The translation of source sentences of length higher than 100 tokens however results in a BLEU score of 21.5% i.e. a drop of 7.8% BLEU and an increase of 7.0% TER. Compared to this the model with relative positional encoding (Line 2) performs equally in the cases $G_{0:25}, \ldots, G_{76:100}$ but does not drop in performance when the source sequences extend beyond length 100.

If we use an LSTM-based positional encoding the performance for sequences up to length 100 is equal to both absolute and relative positional encoding but in-between the two approaches for sequence longer than that. Since LSTM-based positional encodings are weaker than pure relative positional encoding we ignore them for our further investigation.

We observe that absolute and relative positional encoding perform extremely similar in most cases, however for sequences longer than 100 tokens relative positional encoding establishes a very strong lead of around 8% BLEU. We would like to point out that in our setup the maximum source length observed in training is also 100 tokens (see Section 4.1). This raises the question whether absolute positional suffers a) when generalizing to unseen sequence lengths or b) on long sequences in general.

To answer the above question we train two models using absolute respectively relative positional encoding with the same setups as before but restricting the sequence length to 75 tokens in training. This removes 71k sentence pairs, i.e. 1.2% of the training data. Comparing the two versions of the absolute positional encoding model (Table 2 Line 1

	Encoding	max seq.len. in train	Source Length J: $J \in [0, 25]$, $\lvert G_{0:25} \rvert = 14641$ BLEU	TER	$J \in [26, 50]$, $\lvert G_{26:50} \rvert = 11024$ BLEU	TER	$J \in [51, 75]$, $\lvert G_{51:75} \rvert = 2109$ BLEU	TER	$J \in [76, 100]$, $\lvert G_{76:100} \rvert = 261$ BLEU	TER	$J > 100$, $\lvert G_{101:\infty} \rvert = 40$ BLEU	TER
1	Absolute	100	31.7	57.9	32.0	57.3	31.4	58.5	29.3	62.3	21.5	69.3
2	Relative	100	31.8	58.1	32.2	57.1	31.8	58.1	29.3	62.7	29.3	62.7
3	LSTM	100	31.4	58.6	31.9	57.5	31.4	58.4	28.7	63.9	25.5	66.6
4	Absolute	75	31.7	58.1	32.0	57.3	31.2	58.6	25.9	65.0	15.3	74.5
5	Relative	75	31.7	58.0	31.9	57.4	31.5	58.5	29.6	62.2	27.2	66.3

Table 2: Comparison of different positional encoding schemes by sequence length on the De→En task. Results are reported on the concatenation of `newstest2008-2018` without the development set `newstest2015`. Sentences are grouped by their source length J before translation and the number of available sentences per group is given by $|G|$.

vs 4) we notice that they have equal performance for for all sequence lengths up to 75. For sequences with length between 76 and 100 the absolute positional encoding model loses 3.4% BLEU if it does not see these lengths in training. For sequences longer than 100 tokens this difference grows to 6.2% BLEU. Note that this drop in BLEU can not be explained by a loss of training data since relative positional encoding systems on the same training data do not suffer from the same performance loss (Table 2 Line 5). These systems perform equally good as the absolute positional encoding baseline (Line 1) up to source lengths of 100 even though it is not trained to deal with them. For sequences longer than 100 tokens the relative positional encoding system with reduced training data outperforms the baseline by 5.8% BLEU but stays 2.1% BLEU behind its counterpart on the full data (Line 2).

In total we conclude that absolute and relative positional encoding are equally strong for sequence lengths observed in training. Since these make up the majority of the test data that all five systems in Table 2 have a very similar absolute performance with BLEU scores ranging from 39.9% to 40.4% on the `newstest2018`. However relative positional encoding is vastly superior if applied to unseen sequence lengths.

A problem of this kind of analysis is the lack of long sentences available. While $G_{0:25}, G_{26:50}$ and $G_{51:75}$ all contain more than 2k sentence pairs note that we just end up with 40 sequences longer than 100 tokens even when combining 9 test sets. This is obviously a very thin basis to draw conclusions from. While we could extract long sentence pairs from the training data it is quite possible that this results in a very biased selection, e.g. by selecting sentences with very uncommon words which are split heavily by a subword segmentation.

In order to increase the amount of good-quality long sentences we take consecutive sentence pairs from `newstest_all` and concatenate their source respectively target sides. This yields a new test set `concat_seq` with 14k sentence pairs which we group again into $\tilde{G}_{0:25}, \ldots, \tilde{G}_{101:\infty}$ which are translated with the above models. The smallest group contains $|\tilde{G}_{101:\infty}| = 731$ sentence pairs which allows for more reliable observations. This creates a small asymmetry between training (where the models learns to translate one source to one target sentence) and translation (where the models now translate two source to two target sentences). Since all models are equally exposed to this we do not believe that this changes the relative performance difference between the systems.

Table 3 shows that all four models drop quite a bit in performance, however their relative behaviour remains the same: Up to sequences of length 75, all 4 models are equally strong, but for sequence lengths between 75 and 100 only the model using absolute positional encoding with maximum sequence length 75 in training drops in performance. Similar to Table 2 this is a very significant drop by about 4% BLEU. For sequences longer than 100 tokens both relative positional encoding models perform equally well beating the absolute positional encoding baseline by 4% BLEU.

This confirms our observation that relative positional encoding is superior to absolute positional encoding in terms of generalization to unseen sequence lengths.

4.3. Further analysis of relative positional encoding

Since relative positional encoding has proven to be beneficial for sequences of unseen lengths, we further analyze three variations of this method:

- **Relative sinusoidal positional encoding** does not use trainable parameters for γ vectors but instead uses the sinusoids from absolute positional encoding as described in Section 3.2.
- **Relative γ^K-only positional encoding** by using the relative positional information $\gamma^K_{j'-j}$ for the self-attention key but omitting $\gamma^V_{j'-j}$ for the value.
- **Relative and absolute positional encoding** by applying both encoding approaches in the same model.

We report the results for all 3 systems trained with maximum sequence length of 100 tokens on $\tilde{G}_{0:25}, \ldots, \tilde{G}_{101:\infty}$

	Encoding	max seq.len. in train	Source Length J: $J \in [0,25]$, $\lvert\tilde{G}_{0:25}\rvert = 940$ BLEU	TER	$J \in [26,50]$, $\lvert\tilde{G}_{26:50}\rvert = 5903$ BLEU	TER	$J \in [51,75]$, $\lvert\tilde{G}_{51:75}\rvert = 4709$ BLEU	TER	$J \in [76,100]$, $\lvert\tilde{G}_{76:100}\rvert = 1755$ BLEU	TER	$J > 100$, $\lvert\tilde{G}_{101:\infty}\rvert = 731$ BLEU	TER
1	Absolute	100	26.9	63.8	27.1	62.3	27.2	62.4	27.4	62.5	20.1	71.0
2	Relative	100	27.0	63.5	26.9	62.7	27.2	62.9	27.3	63.2	24.5	67.0
3	Absolute	75	26.7	63.0	26.9	62.4	26.9	62.5	22.7	66.3	14.9	74.6
4	Relative	75	26.8	63.0	27.1	62.3	27.3	62.3	27.0	62.8	24.2	66.2

Table 3: Comparison of different positional encoding schemes on the De→En task when translating two consecutive sentences. Results are reported on `concat_seq`. Sentences are grouped by their source length before translation and the number of available sentences per group is given by $|\tilde{G}_i|$.

Encoding	Source Length J: $J \in [0,25]$, $\lvert\tilde{G}_{0:25}\rvert = 940$ BLEU	TER	$J \in [26,50]$, $\lvert\tilde{G}_{26:50}\rvert = 5903$ BLEU	TER	$J \in [51,75]$, $\lvert\tilde{G}_{51:75}\rvert = 4709$ BLEU	TER	$J \in [76,100]$, $\lvert\tilde{G}_{76:100}\rvert = 1755$ BLEU	TER	$J > 100$, $\lvert\tilde{G}_{101:\infty}\rvert = 731$ BLEU	TER
Relative	27.0	63.5	26.9	62.7	27.2	62.9	27.3	63.2	24.5	67.0
Relative sinusoidal	26.4	63.9	26.9	62.8	27.1	63.2	27.0	63.3	24.5	66.5
Relative γ^K-only	27.1	63.0	27.1	62.1	27.1	62.5	27.1	62.9	24.4	66.8
Relative and absolute	26.2	63.9	26.7	62.7	27.0	62.5	26.7	63.5	22.5	67.3

Table 4: Comparison of modifications of absolution positional encoding on the De→En task when translating two consecutive sentences. Results are reported on `concat_seq`. Sentences are grouped by their source length J before translation and the number of available sentences per group is given by $|\tilde{G}_i|$.

of `concat_seq` in Table 4. Surprisingly all variations show a performance very similar to the pure relative positional baseline. Relative sinusoidal positional encodings and the combination of relative and absolute positional encoding are slightly weaker for short sentences. However in further experiments on `newstest_all` we do not observe the same pattern and conclude that it is most likely noise. More meaningful is that the combination of relative and absolute positional encoding lacks behind 2.0% BLEU on $\tilde{G}_{101:\infty}$. We observe similar behaviour on $G_{101:\infty}$ of `newstest_all` where the combination is 4.5% BLEU weaker than pure relative positional encoding.

This indicates that half the parameter added by relative positional encoding can be omitted since the addition of $\gamma^K_{j'-j}$ alone yields equal performance and length generalization.

5. Conclusion

We analyze the behaviour of the Transformer architecture with absolute and relative positional encoding and show that relative positional encoding is strongly superior when translating a source sequence that is longer than any observed training sequence. By excluding long sequences from the training we verify that this gap of 4.4% to 11.9% BLEU is an effect of generalization not of absolute sentence length. We further analyze variations of relative positional encoding and observe that the number of trainable parameters can be reduced by using fixed positional encodings or by removing the weight vectors γ^K without a performance loss.

Because long sequences are rare in test sets, the described effects are often not visible when total BLEU or TER performance is considered. For a strong and stable general purpose system however, these differences are crucial.

We restrict our analysis to the relative positional encoding presented by Shaw et al. [2] and further research should include alternatives such as the one presented by Dai et al. [3]. It remains open for investigation whether the results carry over to other task such as language modeling or automatic speech recognition.

6. Acknowledgements

This work has received funding from the European Research Council (ERC) (under the European Union's Horizon 2020 research and innovation programme, grant agreement No 694537, project "SEQCLAS") and the Deutsche Forschungsgemeinschaft (DFG; grant agreement NE 572/8-1, project "CoreTec"). The work reflects only the authors' views and none of the funding agencies is responsible for any use that may be made of the information it contains.

7. References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin,

"Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[2] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-Attention with Relative Position Representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, Jun. 2018, pp. 464–468.

[3] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2978–2988. [Online]. Available: https://www.aclweb.org/anthology/P19-1285

[4] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1243–1252.

[5] M. X. Chen, O. Firat, A. Bapna, M. Johnson, W. Macherey, G. Foster, L. Jones, M. Schuster, N. Shazeer, N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, Z. Chen, Y. Wu, and M. Hughes, "The best of both worlds: Combining recent advances in neural machine translation," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, Australia, July 2018, pp. 76–86.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[7] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. [Online]. Available: http://arxiv.org/abs/1409.0473

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[9] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2016, pp. 1715–1725.

[10] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, Australia, July 2018, pp. 66–75.