On Scalar Embedding of Relative Positions in Attention Models

Junshuang Wu,[1,2] **Richong Zhang,**[1,2*] **Yongyi Mao,**[3] **Junfan Chen**[1,2]
[1]Beijing Advanced Institution for Big Data and Brain Computing, Beihang University, Beijing, China
[2]SKLSDE, School of Computer Science and Engineering, Beihang University, Beijing, China
[3]School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada
wujs@act.buaa.edu.cn, zhangrc@act.buaa.edu.cn, ymao@uottawa.ca, chenjf@act.buaa.edu.cn

Abstract

Attention with positional encoding has been demonstrated as a powerful component in modern neural network models, such as transformers. However, why positional encoding works well in attention models remains largely unanswered. In this paper, we study the scalar relative positional encoding (SRPE) proposed in the T5 transformer. Such an encoding method has two features. First, it uses a scalar to embed relative positions. Second, the relative positions are bucketized using a fixed heuristic algorithm, and positions in the same bucket share the same embedding. In this work, we show that SRPE in attention has an elegant probabilistic interpretation. More specifically, the positional encoding serves to produce a prior distribution for the attended positions. The resulting attentive distribution can be viewed as a posterior distribution of the attended position given the observed input sequence. Furthermore, we propose a new SRPE (AT5) that adopts a learnable bucketization protocol and automatically adapts to the dependency range specific to the learning task. Empirical studies show that the AT5 achieves superior performance than the T5's SRPE.

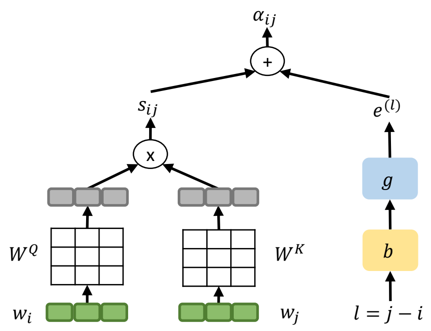


Figure 1: The framework of SRPE. b is the bucketing function and g is the bucket-embedding function. s_{ij} is the context score computed by the dot-product of the transformed word embedding pair; e^l is the scalar embedding for relative position l. α_{ij} is the final attentive score, a summation of the scalar embedding and the context score.

Introduction

Self-attention (SA) has been demonstrated as a key and powerful module for building modern neural networks, such as the Transformer. It has been used in many NLP tasks, including machine translation (Vaswani et al. 2017), natural language inference (Guo, Zhang, and Liu 2019) and text classification (Dai, Li, and Xu 2020). However, without relying on the recurrent structure or the convolutional module, the operations in SA are agnostic to the word order in the input sequence. Thus positional encoding plays a crucial role in SA to capture the temporal information of the text. In general, to incorporate the positional information, previous researchers have proposed varied positional encoding approaches. These approaches can be divided into two types: absolute positional encoding and relative positional encoding.

The absolute positional encoding represents the *absolute position* of the word in an input sentence as embedding vector, with dimension usually taken the same as the token embedding so that the two embeddings can be summed as the word representation. The absolute positional encoding includes Trigonometric Position Embedding (TPE) (Vaswani et al. 2017) and the learnable position embedding (Devlin et al. 2019).

The relative positional encoding (RPE) represents the *relative position* of the word in an input sentence either as a vector (VRPE) (Shaw, Uszkoreit, and Vaswani 2018; Dai et al. 2019) or as a scalar (SRPE) (Raffel et al. 2019). In VRPE, the embedding vectors of relative positions have the same dimension with the word embedding. The SRPE approach used in the T5 Transformer (Raffel et al. 2019), the focus of study in this paper, consists of bucketing function and bucket embedding (Figure 1). The bucketing function assigns the relative positions into different buckets through a fixed heuristic algorithm, and the bucket embedding component maps each bucket to a learnable scalar value; this "scalar embedding" is then added to the corresponding context score to obtain the final attentive score, giving rise to the self-attention weights.

Despite its power demonstrated in practice, how positional encoding functions remains largely a mystery and lacks principled explanation. This research is driven by a curiosity to understand how positional encoding takes effect in SA, where focus on the relative positional encoding approach designed in the T5 transformer. To that end, we arrive at an elegant *probabilistic interpretation* of the positional encoding in the

*Corresponding author: zhangrc@act.buaa.edu.cn
Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

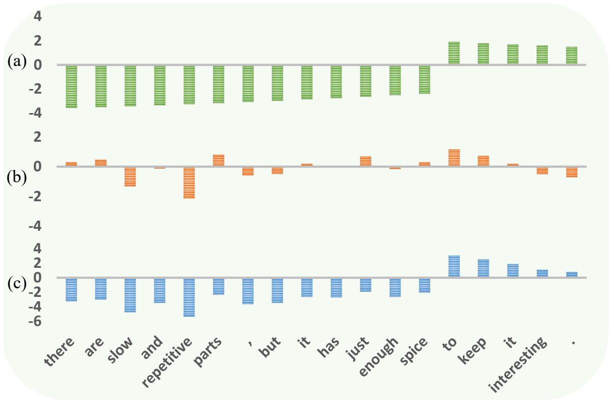


Figure 2: Scores of each token attending to the current central word *spice* of one attention head in SRPE (Scalar Relative Positional Encoding). (a): Relative position scores (b): Context scores (c): Final attentive scores.

SA of T5. Furthermore, we show that, under an appropriate probability model, the scalar positional embedding in T5 can be viewed as defining a prior distribution over the relative positions and that the final attentive distribution can be seen as the posterior given the observed input sequence.

Specifically, the final attentive scores which give rise to the posterior distribution via a softmax transformation are obtained by summing the scalar position embeddings (corresponding to the prior distribution after the softmax transformation) with their respective context scores. A concrete example is shown in Figure 2 to explain the role of SRPE in shaping the attentive scores (hence the posterior distribution). In Figure 2, we expect to gain the attentive scores for computing the self-attention weights when the "query" word is *spice* in sentence *there are slow and repetitive parts, but it has just enough spice to keep it interesting.* We note that in this example, the sentence has a positive sentiment. But the word "interesting" gets a negative context score in (b). Its final attentive score is however flipped to positive (c), which is due to the strong positive signal given by the relative position embedding in (a).

This understanding also enables us to identify two shortfalls of SPRE in T5. First, the bucketing function in T5 is a *fixed* heuristics, incapable of adapting to different tasks or datasets. Second, the scalar embeddings for the T5 buckets are learned independently without being constrained by each other. Arguably the first problem limits the adaptivity of the model, making it risk underfitting in some applications, while the second endows the model with an unnecessarily high capacity, making it risk overfitting.

This paper proposes a revised version of T5 with a new scalar relative positional encoding model, Adaptive T5 (or AT5) to overcome these two limitations. More precisely, we replace the heuristic bucketing function with a learnable function, which can automatically adjust for different tasks. We also impose constraints between the bucket embeddings through a small MLP network. Our experiments on several artificial tasks and three NLP tasks show the superior performance of our proposed model. Our extensive ablation studies and quantitative analysis empirically demonstrate the benefits of AT5 for varied tasks. The source code and the Appendix material are now available at https://github.com/wujsAct/Scalar-Embedding-of-Relative-Positions.

Related Work

In general, SA with the absolute positional encoding uses the sum of position embedding and token embedding as the input token embedding (Vaswani et al. 2017; Devlin et al. 2019). However, TUPE (Ke, He, and Liu 2020) points out that it is beneficial for the model to disentangle the computation of the attentive score for position embedding and the token embedding. Recently, T5 (Raffel et al. 2019) is proposed to generate scalar relative positional embedding independent of the sequence context and then directly add the scalar position embedding into the token's scaled dot-product context score to compute the final attention weights.

Associating the relative positions with a prior distribution appears to represent an important class of techniques in attention or transformer models. For example, in several recent works (e.g., (Guo, Zhang, and Liu 2019; Wang, Lee, and Chen 2019)), explicit prior distributions are designed for the relative positions, and the power of prior distributions is demonstrated. These approaches however do not take an SRPE approach. They use both absolute positional encoding (TPE) and some proposed priors.

Analysis

Attention with Scalar Relative Position Embedding (SRPE Attention)

In the T5 transformer, the attention mechanism uses a scalar embedding for relative positions. We now describe the structure of such attention, which we refer to as *SRPE attention*.

Let the input to a SRPE attention be a sequence of vectors $x_1, x_2, \ldots, x_n$ each having dimension d_{in}. The SRPE attention's objective is to generate, from the input sequence, a sequence of output vectors $y_1, y_2, \ldots, y_n$ each having dimension d_{out}. For simplicity, the input and output sequences will also be denoted as $\mathbf{x}$ and $\mathbf{y}$ respectively. We note that the position indices $1, 2, \ldots, n$ for the input and output are the absolute positions.

For the convenience of working with relative positions, we will switch to a relative position indexing. Specifically, we will use a superscript to denote the relative position of a vector with respect to the current output vector. For example, when consider generating y_i, we will denote y_i by $y^{(0)}$, x_i by $x^{(0)}$, x_{i+1} by $x^{(1)}$, x_{i-1} by $x^{(-1)}$ etc. We will use $\mathcal{L}$ to denote the set of all relative positions. Then $\mathcal{L}$ can be taken as $\{-n^* + 1, -n^* + 2, \ldots, 0, \ldots, n^* - 2, n^* - 1\}$, where n^* is the lengths of the longest document we deal with.

Associate with x_i a "key vector" k_i and a "qurey vector" q_i both having dimension d_{key}, as well as a "value vector" v_i having dimension d_{val}. In transformers, usually each of the three kinds of vectors is expressed as a respective linear transformation the corresponding input vector (Vaswani et al.

2017), where

$$k_i = \mathbf{w}_\mathrm{K} x_i \quad (1)$$
$$q_i = \mathbf{w}_\mathrm{Q} x_i \quad (2)$$
$$v_i = \mathbf{w}_\mathrm{V} x_i \quad (3)$$

where $\mathbf{w}_\mathrm{K}$, $\mathbf{w}_\mathrm{Q}$, $\mathbf{w}_\mathrm{V}$ are learnable parameters. We may again switch to indexing these vectors using relative position indices and, likewise, $k^{(l)}$, $q^{(l)}$, and $v^{(l)}$ are all well defined for all $l \in \mathcal{L}$.

To compute the output vector $y^{(0)}$, at each relative position $l \in \mathcal{L}$, let context "score" $s^{(l)}$ be defined as

$$s^{(l)} := \langle q^{(0)}, k^{(l)} \rangle \quad (4)$$

where $\langle \cdot, \cdot \rangle$ denotes inner product. For each relative position l, let there be a *scalar embedding* $e^{(l)}$. Let a distribution α over the set $\mathcal{L}$ of positions be defined as follows.

$$\alpha(l) := \frac{\exp\left(s^{(l)} + e^{(l)}\right)}{\sum\limits_{l' \in \mathcal{L}} \exp\left(s^{(l')} + e^{(l')}\right)} \quad (5)$$

For later use, we also define another distribution β on the relative positions $\mathcal{L}$: for each $l \in \mathcal{L}$

$$\beta(l) := \frac{\exp\left(e^{(l)}\right)}{\sum\limits_{l' \in \mathcal{L}} \exp\left(e^{(l')}\right)} \quad (6)$$

The output $y^{(0)}$ is then computed as

$$y^{(0)} := \sum_{l \in \mathcal{L}} \alpha(l) v^{(l)} \quad (7)$$

In T5 Transformer, an attention layer essentially contains multiple copies (with separate learnable parameters) of such an attention mechanism. Each copy is referred to as a "head".

SRPE Attention Is Probabilistically Principled

Next, we will show that such an attention structure has an elegant probabilistic interpretation.

Let $\mathbf{X}$ be a compact representation of the input sequence; here, we use a capitalized notation to emphasize that the sequence and each vector within are treated as random variables. Consider a stochastic system that takes $\mathbf{X}$ as input and generates a vector V of dimension d_out. The system will run for n times (recall that n is the length of the input sequence), and at each time i, it generated such a vector V and listed them in order as a $d_\mathrm{in} \times n$ matrix. Thus time i corresponds to column i, or position i, of the output.

We now specify the system, namely, defining how it generates the current output V at each output position i, and we will use relative indexing of the positions as needed.

Let L be a random variable taking values in $\mathcal{L}$. Here L indicates as the relative position in the input sequence the system must attend to in order to generate the current output V. Note that L is dependent on the entire sequence $\mathbf{X}$. To model this dependency, we define the joint distribution $p_{L\mathbf{X}}$ as

$$p_{L\mathbf{X}}(l, x_1, x_2, \ldots, x_n) = \frac{1}{c} \beta(l) \exp(s^{(l)}) \quad (8)$$

where c is a normalization constant so that $p_{L\mathbf{X}}$ is a valid probability measure. Additionally, if the relative position l is attended to, then the system outputs

$$V = v^{(l)} \quad (9)$$

At this point we have completely specified the system. The following result can then be proved.

Theorem 1 *Let p_L and $p_{L|\mathbf{X}}$ be respectively the distribution of L and the conditional distribution of L given X in the above defined system. Then for all $l \in \mathcal{L}$ and all input sequences $(x_1, x_2, \ldots, x_n)$,*

$$p_L(l) = \beta(l) \quad (10)$$
$$p_{L|\mathbf{X}}(l | x_1, x_2, \ldots, x_n) = \alpha(l) \quad (11)$$
$$\mathbb{E}\{V | \mathbf{X} = (x_1, x_2, \ldots, x_n)\} = y^{(0)} \quad (12)$$

This result implies the following. Under the above defined probability model, β, which is only specified via the scalar relative position embedding, can be regarded as the *prior distribution* of the relative position that should be attended to, without observing the sequence $\mathbf{X}$. On the other hand, α, which specifies the attention weights, can be regarded as the *posterior distribution* of relative position upon observing the input sequence $\mathbf{X}$. Finally, the output vector $y^{(0)}$ of SRPE attention at the current position is just the expected output of the defined system when input sequence $(x_1, x_2, \ldots, x_n)$ is fed to the system.

At this end, we have endowed SRPE attention with a probabilistic principle. Under such a principle, designing embedding for SRPE attention, learnable or not, can be seen as designing the prior distribution for the relative positions.

SRPE in T5 Transformer

In general, designing SRPE boils down to designing an encoding function $f : \mathcal{L} \to \mathbb{R}$ from the set $\mathcal{L}$ of relative positions to the set $\mathbb{R}$ of real numbers so that for every $l \in \mathcal{L}$,

$$e^{(l)} = f(l) \quad (13)$$

We now explain the encoding function f_T5 for SRPE used in the attention layers of the T5 transformer.

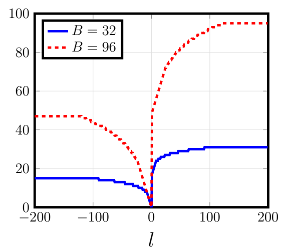
Overall the function f_T5 is expressed as the composition $g \circ b$ of a "bucketing function" b, which maps a relative position to a "bucket" index, and a bucket-embedding function g, which represents a bucket index as a real value. We now describe the two functions concisely.

Bucketing Function b Assume that there are B_max buckets, with indices taking values in $\mathcal{B} := \{0, 1, \ldots, B_\mathrm{max} - 1\}$.

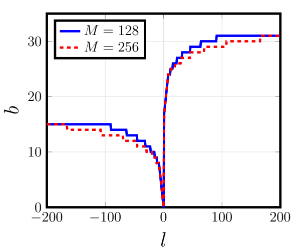
Let $M > B_\mathrm{max}$ be another integer. For each relative position $l \in \mathcal{L}$ and any positive even integer K, define

$$A(l, K) := \begin{cases} l, \text{if } 0 \le l < K/2 \\ \min\left(\frac{K}{2} + \left\lfloor \frac{\log 2l - \log K}{\log 2M - \log K} \cdot \left(\frac{K}{2}\right) \right\rfloor, K - 1\right) \end{cases} \quad (14)$$

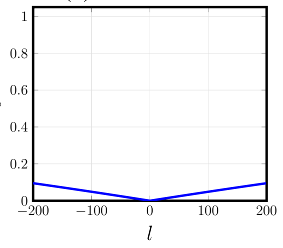
where $\lfloor \cdot \rfloor$ denote the flooring operation.



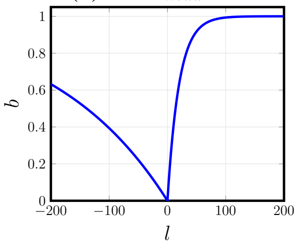
(a) T5: $M = 128$



(b) T5: $B_{max} = 32$



(c) AT5: $\gamma_- = \gamma_+ = 0.1$



(d) AT5:$\gamma_- = 1, \gamma_+ = 10$

Figure 3: Examples of T5 and AT5 bucketing function.

If the T5 transformer is bidirectional (namely attending to both sides of the current position), the bucketing function b is defined as

$$b(l) := \begin{cases} A(-l, \frac{B_{\max}}{2}), & \text{if } l \leq 0; \\ A(l, \frac{B_{\max}}{2}) + \frac{B_{\max}}{2} & \text{if } l > 0. \end{cases} \tag{15}$$

If the T5 transformer only considers the information from the previous tokens, using algorithm A, the bucketing function b is defined as:

$$b(l) := \begin{cases} A(-l, B_{\max}), & \text{if } l \leq 0; \\ 0 & \text{if } l > 0. \end{cases} \tag{16}$$

Note that in both cases, the bucketing function b is not learnable and the function is completely controlled by hyper-parameters $(M, B_{\max})$. We plot the bucket function in T5 with different M and $B_{\max}$ in Figure 3.

Bucket-embedding Function g In T5, the bucketing embedding function g is a mapping from $\{0, 1, \ldots, B_{\max}\}$ to real numbers. Let denote the bucket-embedding matrix as $E \in \mathbb{R}^{B_{max} \times 1}$. Then the $g(b) = \mathbf{e}_b^T E$, where $\mathbf{e}_b \in \mathbb{R}^{B_{max} \times 1}$ is the one-hot vector with a 1 in the b-th coordinate and 0's elsewhere. The unconstrained matrix E is randomly initialized and continuously updated during the training process.

Adaptive T5 Model

Note that our earlier theoretical analysis has provided a principled interpretation of SRPE in the general setting of attention models, where SRPE defines a prior distribution over relative positions. This interpretation allows us to dissect the SRPE structure in T5 in the section above. Two weaknesses of T5 SRPE can be identified under this interpretation. A) the bucketing algorithm is independent of the dataset. B) the embedding of bucket indices is independent of each other. Note that weakness A insists the same bucketing algorithm works for all datasets. But the granularity of semantics depends on the data set and the learning task. The rigid bucketing algorithm then potentially risks an under-fitting. On the other hand, weakness B provides unnecessarily high model freedom in defining the priors over positions (which are grouped rigidly in buckets). This is because one expects that adjacent buckets have similar or correlated prior probability values. Thus excessive modeling freedom induced by weakness B may potentially risk an over-fitting. Further, note the under-fitting induced by weakness A is on a different "dimension" from the over-fitting induced by weakness B. Thus the two issues won't cancel each other. Rather they are expected to demonstrate a compounding negative effect.

We now propose an "Adaptive T5" (AT5) model to alleviate these weaknesses of T5 by replacing its bucketing function with learnable exponential functions, and its bucket-embedding function with small MLP networks.

Let $\mathcal{L} = \mathcal{L}^+ \cap \mathcal{L}^-$ denote the relative position, where $\mathcal{L}^- = \{-n, -n+1, \cdots, -1\}$ and $\mathcal{L}^+ = \{0, 1, \cdots, n\}$ denote the negative relative position set and positive relative position set respectively. We next define the bucketing function and bucket-embedding function for positive and negative relative position separately.

For a positive relative position $l \in \mathcal{L}^+$, we define the bucketing function as:

$$b_+(l) = 1 - \exp^{-l \times A \times \max(0, \gamma_+)} \tag{17}$$

where γ_+ is the trainable parameter to control ramping speed and is initialized from range $[\Gamma_{min}, \Gamma_{max}]$. A is a constant scalar and set as $\frac{1}{n}$. Recall n is the length of the input sequence.

Then we obtain the bucket-embedding through a function g_+, defined as:

$$g_+(b_+) := \text{MLP}(b_+; \theta_+) \tag{18}$$

Here the function MLP denotes the multi-layer perceptrons with two hidden layers and θ_+ denotes the parameters of the multi-layer perceptrons.

Similarly, we define the bucketing function for a negative relative position $l \in \mathcal{L}^-$ as:

$$b_-(l) = 1 - \exp^{l \times A \times \max(0, \gamma_-)} \tag{19}$$

where $\gamma_- \in [\Gamma_{min}, \Gamma_{max}]$ is the trainable parameter to control ramping speed.

And similarly we define the bucket-embedding function g_- as:

$$g_-(b_-) := \text{MLP}(b_-; \theta_-) \tag{20}$$

where θ_- is the parameters of the multi-layer perceptrons.

For illustrative purposes, the bucketing function of AT5 with different (γ_+, γ_-) settings is plotted in Figure 3.

Experiment

We evaluate the proposed AT5 model on some artificial tasks, text classification, question answering, and machine translation. We adopt AT5 in the Transformer encoder model in classification or regression tasks, including artificial tasks, text classification, and question answering. In machine translation, we apply AT5 in both the encoder and decoder part of the Transformer-based encoder-decoder model. We implement all models using Tensorflow and run the experiments on NVIDIA V100 8GB GPU or NVIDIA V100 32GB GPU.

Model	Reber	Process-50	Adding-100
Position	absolute	relative	neither
Non-PE	0.496	0.549	1.0
T5-NoB	1.0	0.772	0.943
T5	1.0	0.808	1.0
AT5-NoB	1.0	0.843	0.961
AT5	1.0	**0.843**	1.0

Table 1: The evaluation results on artificial tasks. Bold-face values indicate the best performance.

Baseline Models

The proposed adaptive T5 model is compared with two baseline models: (1) without position information (Non-PE); (2) with T5 relative position encoding (T5) (Raffel et al. 2019). We introduce a suffix "NoB" to indicate a positional encoding approach without a bucketing function for the ablation study. Specifically, T5-NoB denotes the Transformer model with T5 positional encoding but does not utilize the bucketing function. AT5-NoB denotes the Transformer model with AT5 positional encoding but maps each relative position to a scalar value between $[0, 1]$ as the input for the bucket-embedding layer. We also report other baseline models in specific tasks for the completeness of the experiments. Such as the Word2vec (Conneau et al. 2017), Sent2vec (Pagliardini, Gupta, and Jaggi 2018), QuickThoughts (Logeswaran and Lee 2018) models in the text classification task and the QANet (Yu et al. 2018) model in question answering task.

Artificial Tasks

The artificial tasks that we use to evaluate the baseline models and the AT5 model include:

(1) *Embedded Reber Grammar:* The goal of this task is to predict the last character of a given string (a sequence of characters). The string is generated by a logic engine named *Reber grammar*. Note that, the last character of the generated string only depends on the string's second character, which means this task relies on the absolute positional information. The reader can refer to the Appendix for details.

(2) *Process Classification:* This task is a sequence-classification task, in which each sample is a binary sequence generated by a transition probability matrix. The sequences generated by the same transition probability matrix are assigned with the same label. In this paper, we generate two classes of sequences by using two different transition matrix. In this task, models need to rely on the neighbor elements to infer the state transition pattern. That is, this task relies on relative positional information.

(3) *Adding Problem:* The task aims to predict a target value given a sequence of value-marker pairs. Let T denote the length of the pair sequence and (v_i, m_i) denote the i^{th} value-marker pair. Here $v_i \in [-1, 1]$ is a value element and $m_i \in \{-1, 0, 1\}$ is a marker element which is used to represent the start and end position. The target value of a pair sequence in the data is calculated as $0.5 + \frac{v_j + v_k}{4}$, where $j < 10$ and $k < \frac{T}{2} - 1$ are randomly selected pair indexes, and the corresponding maker elements m_j and m_k are set to 1. This task relies on neither absolute positional information nor relative positional information, since the computation of the target value is independent with positions. As a standard evaluation procedure, we treat a predicted value whose absolute error with the target value less than 0.04 as a correct prediction (Hochreiter and Schmidhuber 1997).

The dimension of the word embedding and the hidden layers are 256 and 512. The number of heads in the self-attention module is 8. The learning rate is chosen from the set $\{1e-4, 5e-4\}$. The number of layers in all Transformer models is set to 1. The numbers of training samples are 5000 for Process-50 and 1000 for Reber and Adding-100. The number of testing samples is 5000 for all tasks. For Process-50, the average of the encoder's output vectors is used as features for classification. And for Reber and Adding-100, the last output vector is used as features for classification. In the T5 model, the ramping parameter M is set as the longest sequence length in each task's training data. The bucket number is 32. In AT5 model, the hyper-parameter pair $(\Gamma_{min}, \Gamma_{max})$ are selected from $\{(0.1, 10.0), (1.0, 10.0)\}$, and the hidden sizes for MLP are $(100, 5)$ for reber and $(15, 2)$ for both Process-50 and Adding-100. All the hyper-parameters are fine-tuned on the validation set. We run the experiments with the same setting five times for each model and report its average performance.

The experimental results in artificial tasks are shown in Table 1. From the table, we observe that the Non-PE Transformer model performs much worse than other Transformer models, except for the adding problem. This fact indicates that Non-PE may struggle with tasks highly relying on the positional information.

In the Embedded Reber Grammar task, as the classification target relies on the character in the second position (absolute position), models with positional encoding achieve much better performance than the Transformer with Non-PE. In contrast, Non-PE performs nearly randomly guessing. T5 and AT5 achieve the same accuracy because they focus on modeling relative positions instead of absolute positions.

In the Process-50 task relying on relative positional information, the Non-PE model still achieves bad performance, only a little better than random guessing. T5-NoB achieves lower accuracy than T5, revealing the importance of the bucketing function for relative positional encoding. The higher accuracy of AT5 comparing to T5 demonstrates the effectiveness of the proposed learnable bucketing function and dependency modeling of scalar embeddings in AT5.

In the Adding-100 task, we observe that either the models with positional encoding or without positional encoding achieve similar and very high accuracy. The reason is that the Adding problem relies on neither absolute position nor relative position, thus just a simple Transformer without positional encoding can perfectly solve this problem. AT5-NoB (T5-NoB) performs worse than AT5 (T5), revealing that SRPE without bucketing function may bring noises for the Transformer when handling this problem.

Text Classification

We evaluate the proposed AT5 model on six real-world text classification datasets, including MR, SUBJ, CR, MPQA, SST, and TREC. The statistics for these text classifica-

Dataset	MR	SUBJ	CR	MPQA	SST	TREC
Class	2	2	2	2	2	6
Length	59	120	105	36	52	37
Train	9595	9000	3397	9549	67349	5452
Dev	1067	1000	378	1061	872	329
Test	-	-	-	-	1821	500

Table 2: Dataset statistic of text classification.

Model	MR	SUBJ	CR	MPQA	SST	TREC
W2V	0.777	0.909	0.798	0.883	0.797	0.836
S2V	0.763	0.912	0.791	0.872	0.802	0.858
QT	<u>0.824</u>	<u>0.948</u>	<u>0.86</u>	0.902	-	0.924
Non-PE	0.805	0.941	0.844	0.902	0.847	0.916
T5-NoB	0.805	0.940	0.844	0.903	0.851	0.927
T5	0.806	0.939	0.845	0.903	0.849	0.925
AT5-NoB	0.810	0.940	0.846	0.903	0.851	0.920
AT5	**0.812**	**0.943**	**0.851**	0.901	<u>**0.863**</u>	<u>**0.94**</u>

Table 3: Text classification evaluation results. W2V: Word2Vec; S2V: Sentence2Vec; QT: QuickThoughts.

tion datasets are presented in Table 2. Following the prior work (Wang et al. 2020), on the SST and TREC datasets, we report the average accuracy for five runs. For other datasets, the results of the 10-fold cross-validation are reported.

The word-embedding is initialized using Glove (Pennington, Socher, and Manning 2014). The number of layers for all Transformer models is set to 5. The number of heads is 6. The learning rate is 2e-4 and the input dropout probability is 0.4. The residual dropout probability is set to 0.3. The features for all datasets are the last output vector from the encoder. In the T5 model, the ramping parameter M is set as 128, and the bucket number is 32. In AT5 model, $(\Gamma_{min}, \Gamma_{max})$ is set as $(1.0, 10.0)$, and the hidden sizes for MLP are 64 and 8. The position scores of T5 and AT5 are added at the first layer of the Transformer model.

The evaluation results for text classification task are shown in Table 3. From the table, we observe that the models with the Transformer encoder outperform Word2Vec and Sent2Vec models, demonstrating the advantage of the Transformer. The Transformer models with positional embedding achieve similar performance with the Non-PE model on the SUBJ, CR, and MPQA datasets. This phenomenon illustrates that the positional information provides little contribution to classifying the text in these datasets. AT5 outperforms T5 with large margins on the MR, SST, and TREC datasets, which indicates the superiority of AT5 compared to T5.

Question Answering

Given a passage and a query, the SQuAD question answering task's goal is to find the answer in the passage. In general, The answer is a text span from the passage. The total number of examples in the dataset is $107.7k$. Similar to the previous work QANet (Yu et al. 2018), we extract $10.1k$ examples as the validation set, another $10.1k$ examples as the test set, and the left $87.5k$ as the training dataset. Moreover, the maximum length of the passages and queries are 400 and 50.

Model	EM	F1
QANet	73.6	82.7
T5-NoB	70.95(0.45)	80.08(0.4)
T5 (32-128)	71.29(0.19)	80.36(0.12)
T5(64-512)	71.28(0.24)	80.36(0.21)
T5(128-512)	71.07(0.3)	80.14(0.22)
AT5-NoB	70.49(0.51)	79.88(0.52)
AT5	**72.12(0.41)**	**80.93(0.21)**

Table 4: The performance of QANet with varied positional encoding for SQuAD 1.0.

Sent length	2014t	2015t	2016t	2017t	2017corpus
(0, 20]	975	796	1313	1216	2,003,208
(20, 40]	1379	942	1274	1404	2,474,947
(40, 60]	523	345	350	336	989,654
(60, 80]	110	64	54	43	296,460
(80, 100]	14	16	7	3	77,006
(100, 120]	2	6	1	2	8,992

Table 5: The statistics of the machine translation datasets. Here 2014t, 2015t, 2016t and 2017t represent the newstest2014, newstest2015, newstest2016 and newstest2017.

For the QANet model, the embedding encoder layer and the model encoder layer is a stack of three basic blocks: several convolutional layers, a self-attention-layer, and a feed-forward layer. In this study, we focus on evaluating different positional embedding approaches. Thus, we remove the convolutional layers from the QANet. To study the affection of positional encoding in the QA task, we replace TPE in the QANet with T5 or AT5. We add T5 or AT5 positional encoding at each layer. In each encoder block of AT5, we set the same hyper-parameters for each layer's bucketing algorithm. In T5, the best ramping parameter M is set as the 128, and the bucket number is 32. In AT5, $(\Gamma_{min}, \Gamma_{max})$ is set as $(5.0, 15.0)$ and the hidden sizes of MLP are 50 and 10. We tune parameter on validation dataset.

The evaluation results for the question answering task are presented in Table 4. The results of the original QANet are from (Yu et al. 2018). We report the average EM and F1 scores in five runs. In Table 4, AT5 performs better than T5. Different configurations of bucket numbers and ramping parameters do not provide a significant performance improvement. We can also find that without a bucketing algorithm, the performance of both T5 and AT5 will decline.

Machine Translation

We utilize the WMT 2017 English-German (en2de) corpus as the training set and the newstest2014 dataset as the validation dataset in the machine translation (MT) task. The newstest2015, newstest2016, and newstest2017 datasets are employed as the test sets. The statistics for the machine translation datasets are shown in Table 5.

We implement the machine translation models on THUMT platform (Zhang et al. 2017). Label smooth (Müller, Kornblith, and Hinton 2019) is set to 0.1. Following the previous researches (Klein et al. 2017; Vaswani et al. 2018), we use the

Dataset	2014t	2015t	2016t	2017t
T5-NoB	28.24(0.13)	30.07(0.1)	33.73(0.23)	28.06(0.15)
T5	28.15(0.07)	30.05(0.25)	33.64(0.23)	28.0(0.13)
AT5-NoB	27.91(0.31)	29.86(0.19)	33.58(0.27)	28.0(0.20)
AT5	**28.34(0.07)**	**30.18(0.13)**	**34.26(0.25)**	**28.21(0.14)**

Table 6: Comparison of Transformer-based NMT model with different positional encoding methods on test datasets. The value in bracket is the standard deviation.

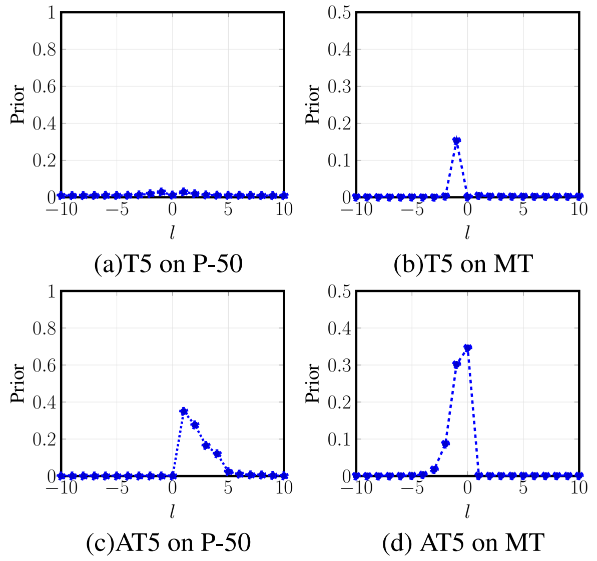


Figure 4: The prior probabilities of one head at a self-attention layer learned by T5 and AT5 on the Process-50 (P-50) and machine translation (MT) tasks.

pre-norm residual unit, and the residual dropout is 0.1. The maximum sequence length for encoder and decoder are both 256. The length penalty term in the beam search (Wu et al. 2016) is 0.6. The maximum length ratio of a translation is 50. The batch size is 25000. The evaluation metrics is Multi-BLEU. We only add T5 and AT5 positional encoding at the first layer of the encoder and decoder. The hyper-parameters B_{max} and M for T5 are 64 and 128 respectively. In AT5 model, $(\Gamma_{min}, \Gamma_{max})$ is set as $(1.0, 15.0)$ and the hidden sizes of MLP are 32 and 4.

The evaluation results of the machine translation task are shown in Table 6. AT5 consistently performs better than T5, especially on the testnews2016 dataset. Comparing the performance of T5 and T5-NoB, we find that heuristic bucketing function in T5 do not provide performance improveme nt on all test dataset in MT task. In contrast, the bucketing function proposed in AT5 brings benefits to model performance.

Quantitative Analysis

To explain how the AT5 works better than T5, we make quantitative analysis about T5 and AT5. Specifically, we first analyze the prior probabilities of T5 and AT5 generated on the Process-50 and MT, then make an ablation study on the heads in the Transformer. Figure 4 presents the distributions of prior probabilities learned by T5 and AT5 on different tasks. On the Process-50 task, the prior probabilities learned by T5 uniformly distribute at relative positions. Different from T5, the prior probabilities learned by AT5 are higher and uneven among different relative positions. This phenomenon indicates that T5 fails to learn the difference among relative positions, leading to worse performance than AT5. The high prior probabilities of AT5 near the peak demonstrate that AT5 pays more attention to short-distance relative positions than long-distance relative positions. On the MT task, both T5 and AT5 successfully learned the difference in relative positions. However, AT5 learned better distribution for the prior probabilities and achieved higher BLEU than T5.

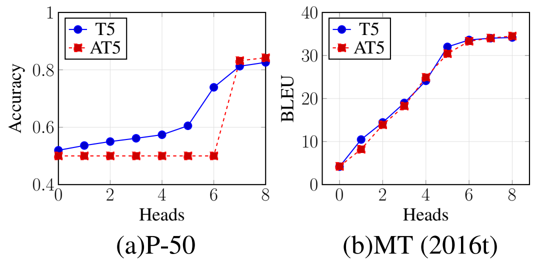


Figure 5: Prior probability ablation study for T5 and AT5.

The ablation study in Figure 5 illustrates the changes of the test performance for T5 and AT5 when the heads are added into the Transformer one by one according to their maximum probabilities among all relative positions (from heads with lower maximum probabilities to heads with higher maximum probabilities). From the figure, we observe that when the lower-probability heads are added into the transformer, the T5 performs better than AT5, and when the higher-probability heads are added into the transformer, the performance of AT5 improves faster than T5 and eventually outperforms T5. This fact demonstrates that the higher-probability heads in AT5 make primary contribution for its performance gain.

Conclusion

In this paper, we provide a probabilistic interpretation of the T5 scalar relative positional encoding and reveal its limitations in its fixed bucketing heuristics and in its lack of constraint across bucket embeddings. To overcome these limitations, we propose a novel Adaptive T5 (AT5) model that combines a learnable bucketing function with dependency modeling of bucket embeddings. The proposed model is demonstrated to improve upon T5 in artificial tasks, text classification, question answering, and machine translation tasks, and may apply to other NLP tasks.

Acknowledgments

This work is supported partly by the National Natural Science Foundation of China (No. 61772059, 61421003), by the Beijing Advanced Innovation Center for Big Data and Brain Computing (BDBC), by the Fundamental Research Funds for the Central Universities, by the Beijing S&T Committee (No. Z191100008619007) and by the State Key Laboratory of Software Development Environment (No. SKLSDE-2020ZX-14).

References

Conneau, A.; Kiela, D.; Schwenk, H.; Barrault, L.; and Bordes, A. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In Palmer, M.; Hwa, R.; and Riedel, S., eds., *Proceedings of EMNLP*, 670–680. Association for Computational Linguistics. URL https://doi.org/10.18653/v1/d17-1070.

Dai, B.; Li, J.; and Xu, R. 2020. Multiple Positional Self-Attention Network for Text Classification. In *Proceedings of AAAI*, 7610–7617. AAAI Press. URL https://aaai.org/ojs/index.php/AAAI/article/view/6261.

Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J. G.; Le, Q. V.; and Salakhutdinov, R. 2019. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In Korhonen, A.; Traum, D. R.; and Màrquez, L., eds., *Proceedings of ACL*, 2978–2988. Association for Computational Linguistics. URL https://doi.org/10.18653/v1/p19-1285.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Burstein, J.; Doran, C.; and Solorio, T., eds., *Proceedings of NAACL-HLT*, 4171–4186. Association for Computational Linguistics. URL https://doi.org/10.18653/v1/n19-1423.

Guo, M.; Zhang, Y.; and Liu, T. 2019. Gaussian Transformer: A Lightweight Approach for Natural Language Inference. In *Proceedings of AAAI*, 6489–6496. AAAI Press. URL https://doi.org/10.1609/aaai.v33i01.33016489.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation* 9(8): 1735–1780. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Ke, G.; He, D.; and Liu, T. 2020. Rethinking Positional Encoding in Language Pre-training. *CoRR* abs/2006.15595. URL https://arxiv.org/abs/2006.15595.

Klein, G.; Kim, Y.; Deng, Y.; Senellart, J.; and Rush, A. M. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In Bansal, M.; and Ji, H., eds., *Proceedings of ACL*, 67–72. Association for Computational Linguistics. URL https://doi.org/10.18653/v1/P17-4012.

Logeswaran, L.; and Lee, H. 2018. An efficient framework for learning sentence representations. In *Proceedings of ICLR*. OpenReview.net. URL https://openreview.net/forum?id=rJvJXZb0W.

Müller, R.; Kornblith, S.; and Hinton, G. E. 2019. When does label smoothing help? In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., *Processing of NeurIPS*, 4696–4705. URL http://papers.nips.cc/paper/8717-when-does-label-smoothing-help.

Pagliardini, M.; Gupta, P.; and Jaggi, M. 2018. Unsupervised Learning of Sentence Embeddings Using Compositional n-Gram Features. In Walker, M. A.; Ji, H.; and Stent, A., eds., *Proceedings of NAACL-HLT*, 528–540. Association for Computational Linguistics. URL https://doi.org/10.18653/v1/n18-1049.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global Vectors for Word Representation. In Moschitti, A.; Pang, B.; and Daelemans, W., eds., *Proceedings of EMNLP*, 1532–1543. ACL. URL https://doi.org/10.3115/v1/d14-1162.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *CoRR* abs/1910.10683. URL http://arxiv.org/abs/1910.10683.

Shaw, P.; Uszkoreit, J.; and Vaswani, A. 2018. Self-Attention with Relative Position Representations. In Walker, M. A.; Ji, H.; and Stent, A., eds., *Proceedings of NAACL-HLT*, 464–468. Association for Computational Linguistics. URL https://doi.org/10.18653/v1/n18-2074.

Vaswani, A.; Bengio, S.; Brevdo, E.; Chollet, F.; Gomez, A. N.; Gouws, S.; Jones, L.; Kaiser, L.; Kalchbrenner, N.; Parmar, N.; Sepassi, R.; Shazeer, N.; and Uszkoreit, J. 2018. Tensor2Tensor for Neural Machine Translation. In Cherry, C.; and Neubig, G., eds., *Proceedings of AMTA*, 193–199. Association for Machine Translation in the Americas. URL https://www.aclweb.org/anthology/W18-1819/.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Proceedings of NIPs*, 5998–6008. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.

Wang, B.; Zhao, D.; Lioma, C.; Li, Q.; Zhang, P.; and Simonsen, J. G. 2020. Encoding word order in complex embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. URL https://openreview.net/forum?id=Hke-WTVtwr.

Wang, Y.; Lee, H.; and Chen, Y. 2019. Tree Transformer: Integrating Tree Structures into Self-Attention. In Inui, K.; Jiang, J.; Ng, V.; and Wan, X., eds., *Proceedings of EMNLP-IJCNLP*, 1061–1070. Association for Computational Linguistics. URL https://doi.org/10.18653/v1/D19-1098.

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; Klingner, J.; Shah, A.; Johnson, M.; Liu, X.; Kaiser, L.; Gouws, S.; Kato, Y.; Kudo, T.; Kazawa, H.; Stevens, K.; Kurian, G.; Patil, N.; Wang, W.; Young, C.; Smith, J.; Riesa, J.; Rudnick, A.; Vinyals, O.; Corrado, G.; Hughes, M.; and Dean, J. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR* abs/1609.08144. URL http://arxiv.org/abs/1609.08144.

Yu, A. W.; Dohan, D.; Luong, M.; Zhao, R.; Chen, K.; Norouzi, M.; and Le, Q. V. 2018. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. In *Proceedings of ICLR*. OpenReview.net. URL https://openreview.net/forum?id=B14TlG-RW.

Zhang, J.; Ding, Y.; Shen, S.; Cheng, Y.; Sun, M.; Luan, H.; and Liu, Y. 2017. THUMT: An Open Source Toolkit for Neural Machine Translation. *CoRR* abs/1706.06415. URL http://arxiv.org/abs/1706.06415.