POSITION INFORMATION IN TRANSFORMERS: AN OVERVIEW

Philipp Dufter*, Martin Schmitt*, Hinrich Schütze
Center for Information and Language Processing (CIS), LMU Munich, Germany
{philipp,martin}@cis.lmu.de

ABSTRACT

Transformers are arguably the main workhorse in recent Natural Language Processing research. By definition a Transformer is invariant with respect to reorderings of the input. However, language is inherently sequential and word order is essential to the semantics and syntax of an utterance. In this paper, we provide an overview of common methods to incorporate position information into Transformer models. The objectives of this survey are to i) showcase that position information in Transformer is a vibrant and extensive research area; ii) enable the reader to compare existing methods by providing a unified notation and meaningful clustering; iii) indicate what characteristics of an application should be taken into account when selecting a position encoding; iv) provide stimuli for future research.

1 INTRODUCTION

The Transformer model as introduced by Vaswani et al. (2017) has been found to perform well for many tasks, such as machine translation or language modeling. With the rise of pretrained language models (PLMs) (Peters et al., 2018; Howard & Ruder, 2018; Devlin et al., 2019; Brown et al., 2020) Transformer models have become even more popular. As a result they are at the core of many state of the art natural language processing (NLP) models. A Transformer model consists of several layers, or blocks. Each layer is a self-attention (Vaswani et al., 2017) module followed by a feed-forward layer. Layer normalization and residual connections are additional components of a layer.

A Transformer model itself is invariant with respect to re-orderings of the input. However, text data is inherently sequential. Without position information the meaning of a sentence is not well-defined (e.g., the sequence "the cat chases the dog" vs. the multi-set { the, the, dog, chases, cat }). Clearly it should be beneficial to incorporate this essential inductive bias into any model that processes text data.

Therefore, there is a range of different methods to incorporate position information into NLP models, especially PLMs that are based on Transformer models. Adding position information can be done by using position embeddings, manipulating attention matrices, or preprocessing the input with a recurrent neural network. Overall there is a huge variety of methods that add both absolute and relative position information to Transformer model. Similarly, many papers analyze and compare a subset of position embedding variants. But, to the best of our knowledge, there is no broad overview of relevant work on position information in Transformers that systematically aggregates and categorizes existing approaches and analyzes the differences between them.

The objective of this paper is to provide an overview of methods that incorporate and analyze position information in Transformer models. More specifically we aim at i) showcasing that position information in Transformer is a vibrant and extensive research area ii) enabling the reader to compare existing methods by providing a unified notation and meaningful clustering iii) indicating what characteristics of an application should be taken into account when selecting a position encoding iv) providing stimuli for future research. This paper is work in progress. We plan to actively continue adding papers that we missed or that are newly published. If you want to contribute, spot an error or miss a paper, please do reach out to us.

* Equal contribution - random order.

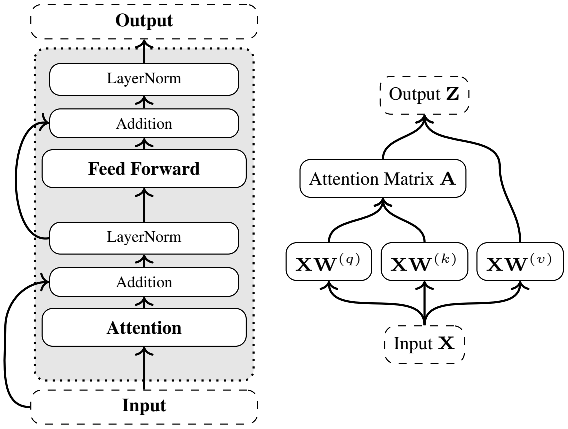


Figure 1: A rough overview of a plain Transformer Encoder Block (grey block) without any position information. They grey block is usually repeated for l layers. An overview of the actual attention computation is shown on the right.

2 Background

2.1 Notation

We denote scalars with lowercase letters $x \in \mathbb{R}$, vectors with boldface $\mathbf{x} \in \mathbb{R}^d$, and matrices with boldface uppercase letters $\mathbf{X} \in \mathbb{R}^{t \times d}$. We index vectors and matrices as follows $(x_i)_{i=1,2\ldots,d} = \mathbf{x}$, $(X_{ij})_{i=1,2\ldots,t,j=1,2,\ldots d} = \mathbf{X}$. Further, the i-th row of $\mathbf{X}$ is the vector $\mathbf{X}_i \in \mathbb{R}^d$. The transposed is denoted as $\mathbf{X}^\intercal$. When we are referring to positions we use $r, s, t, \ldots$, whereas we use $i, j, \ldots$ to denote components of a vector. The maximum sequence length is called $t_{\max}$.

2.2 Transformer Model

Attention mechanisms were first used in the context of machine translation by Bahdanau et al. (2015). While they still relied on a recurrent neural network in its core, Vaswani et al. (2017) proposed a model that relies on attention only. They found that it outperforms current recurrent neural network approaches by large margins on the machine translation task. In their paper they introduced a new neural network architecture, the *Transformer Model*, which is an encoder-decoder architecture. We now briefly describe the essential building block, *Transformer Encoder Blocks* as shown in Figure 1. One block, also called layer, is a function $f_\theta : \mathbb{R}^{t_{\max} \times d} \to \mathbb{R}^{t_{\max} \times d}$ with $f_\theta(\mathbf{X}) =: \mathbf{Z}$ that is defined by

$$\begin{aligned}
\mathbf{A} &= \sqrt{\frac{1}{d}}\mathbf{XW}^{(q)}(\mathbf{XW}^{(k)})^\intercal \\
\mathbf{M} &= \text{SoftMax}(\mathbf{A})\mathbf{XW}^{(v)} \\
\mathbf{O} &= \text{LayerNorm}_1(\mathbf{M} + \mathbf{X}) \\
\mathbf{F} &= \text{ReLU}(\mathbf{OW}^{(f_1)} + \mathbf{b}^{(f_1)})\mathbf{W}^{(f_2)} + \mathbf{b}^{(f_2)} \\
\mathbf{Z} &= \text{LayerNorm}_2(\mathbf{O} + \mathbf{F}).
\end{aligned} \tag{1}$$

Here, $\text{SoftMax}(\mathbf{A})_{ts} = e^{\mathbf{A}_{ts}} / \sum_{k=1}^{t_{\max}} e^{\mathbf{A}_{tk}}$ is the row-wise softmax function, $\text{LayerNorm}(\mathbf{X})_t = \mathbf{g} \odot (\mathbf{X}_t - \mu(\mathbf{X}_t)/\sigma(\mathbf{X}_t) + \mathbf{b}$ is layer normalization (Lei Ba et al., 2016) where $\mu(\mathbf{x}), \sigma(\mathbf{x})$ returns the mean, standard deviation of a vector, and last $\text{ReLU}(\mathbf{X}) = \max(0, \mathbf{X})$ is the maximum operator applied componentwise. Note that for addition of a vector to a matrix we assume broadcasting as implemented in NumPy.[1] Overall the parameters of a single layer are

$$\begin{aligned}
\theta = (&\mathbf{W}^{(q)}, \mathbf{W}^{(k)}, \mathbf{W}^{(v)} \in \mathbb{R}^{d \times d}, \mathbf{g}^{(1)}, \mathbf{g}^{(2)}, \mathbf{b}^{(1)}, \mathbf{b}^{(2)} \in \mathbb{R}^d, \\
&\mathbf{W}^{(f_1)} \in \mathbb{R}^{d \times d_f}, \mathbf{W}^{(f_2)} \in \mathbb{R}^{d_f \times d}, \mathbf{b}^{(f_1)} \in \mathbb{R}^{d_f}, \mathbf{b}^{(f_2)} \in \mathbb{R}^d),
\end{aligned} \tag{2}$$

[1] `https://numpy.org/`

with d the hidden dimension, d_f the intermediate dimension, and $t_{\max}$ the maximum sequence length. It is common to consider multiple, say h, attention heads. More specifically, $\mathbf{W}^{(q)}, \mathbf{W}^{(k)}, \mathbf{W}^{(v)} \in \mathbb{R}^{d \times d_h}$ where $d = hd_h$. Subsequently, the matrices $\mathbf{M}^{(h)} \in \mathbb{R}^{t_{\max} \times d_h}$ coming from each attention head are concatenated along the second dimension to obtain $\mathbf{M}$. A full *Transformer model* is then the function $T : \mathbb{R}^{t_{\max} \times d} \rightarrow \mathbb{R}^{t_{\max} \times d}$ that consists of the composition of multiple, say l layers, i.e., $T(\mathbf{X}) = f_{\theta^l} \circ f_{\theta^{l-1}} \circ \cdots \circ f_{\theta^1}(\mathbf{X})$.

When considering an input $U = (u_1, u_2, \ldots, u_t)$ that consists of t units, such as characters, subwords, words, first unit embeddings $\mathbf{U} \in \mathbb{R}^{t_{\max} \times d}$ are created by a lookup in the embedding matrix $\mathbf{E} \in \mathbb{R}^{n \times d}$ with n being the vocabulary size. More specifically, $\mathbf{U}_i = \mathbf{E}_{u_i}$ is the embedding vector that corresponds to the unit u_i. Finally, the $\mathbf{U}$ is then (among others) used as input to the Transformer model. In the case that U is shorter than $t_{\max}$, it is padded, i.e., filled with special PAD symbols.

2.3 Order Invariance

If we take a close look at the Transformer model, we see that it is invariant to re-orderings of the input. More specifically, consider any permutation matrix $\mathbf{P}_\pi \in \mathbb{R}^{t_{\max} \times t_{\max}}$. When passing $\mathbf{P}_\pi \mathbf{X}$ to a Transformer layer, one gets $\mathbf{P}_\pi \text{SoftMax}(\mathbf{A}) \mathbf{P}_\pi{}^\intercal \mathbf{P}_\pi \mathbf{X} \mathbf{W}^{(v)} = \mathbf{P}_\pi \mathbf{M}$, as $\mathbf{P}_\pi{}^\intercal \mathbf{P}_\pi$ is the identity matrix. All remaining operations are position-wise and thus $\mathbf{P}_\pi T(\mathbf{X}) = T(\mathbf{P}_\pi \mathbf{X})$ for any input $\mathbf{X}$. As language is inherently sequential it is desirable to have $\mathbf{P}_\pi T(\mathbf{X}) \neq T(\mathbf{P}_\pi \mathbf{X})$, which can be achieved by incorporating position information.

2.4 Encoder-Decoder

There are different setups how to use a Transformer model. One common possibility is to have an encoder only. For example, BERT (Devlin et al., 2019) uses a Transformer model $T(\mathbf{X})$ as encoder to perform masked language modeling. In contrast, a traditional sequence-to-sequence approach can be materialized by adding a decoder. The decoder works almost identically to the encoder with two exceptions: 1) The upper triangle of the attention matrix $\mathbf{A}$ is usually masked in order to avoid information flow from future positions during the decoding process. 2) The output of the encoder is integrated through a cross-attention layer inserted before the feed forward layer. See (Vaswani et al., 2017) for more details. The differences between an encoder and encoder-decoder architecture are mostly irrelevant for the injection of position information and many architectures rely just on encoder layers. Thus for the sake of simplicity we will talk about Transformer Encoder Blocks in general for the rest of the paper. See §4.4 for position encodings that are tailored for encoder-decoder architectures.

3 Recurring Concepts in Position Information Models

While there is a variety of approaches to integrate position information into Transformers, there are some recurring ideas, which we outline here.

3.1 Absolute vs. Relative Position Encoding

Absolute positions encode the absolute position of a unit within a sentence. Another approach is to encode the position of a unit *relative* to other units. This makes intuitively sense, as in sentences like “The cat chased the dog.” and “Suddenly, the cat chased the dog.” the change in absolute positions comes only with a slight semantic change, whereas the relative positions of “cat” and “dog” are decisive for the meaning of the sentences.

3.2 Representation of Position Information

Adding Position Embeddings (APE). One common approach is to add position embeddings to the input before it is fed to the actual Transformer model: If $\mathbf{U} \in \mathbb{R}^{t_{\max} \times d}$ is the matrix of unit embeddings, a matrix $\mathbf{P} \in \mathbb{R}^{t_{\max} \times d}$ representing the position information is added, i.e., their sum is fed to the Transformer model: $T(\mathbf{U} + \mathbf{P})$. For the first Transformer layer, this has the following

$$\begin{array}{c|ccc|}
 & \text{You} & \text{are} & \text{great} \\
\text{You} & a_{11} & a_{12} & a_{13} \\
\text{are} & a_{21} & a_{22} & a_{23} \\
\text{great} & a_{31} & a_{32} & a_{33}
\end{array}
\quad
\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}
\quad
\begin{bmatrix} r_0 & r_1 & r_2 \\ r_{-1} & r_0 & r_1 \\ r_{-2} & r_{-1} & r_0 \end{bmatrix}$$

Attention Matrix *Absolute Position Bias* *Relative Position Bias*

Figure 2: Example of absolute and relative position biases that can be added to the attention matrix. *Left:* self-attention matrix for an example sentence. *Middle:* learnable absolute position biases. *Right:* position biases with a relative reference point. They are different to absolute encodings as they exhibit an intuitive weight sharing pattern.

effect:

$$\begin{aligned}
\tilde{\mathbf{A}} &= \sqrt{\frac{1}{d}}(\mathbf{U}+\mathbf{P})\mathbf{W}^{(q)}\mathbf{W}^{(k)\intercal}(\mathbf{U}+\mathbf{P})^\intercal \\
\tilde{\mathbf{M}} &= \text{SoftMax}(\tilde{\mathbf{A}})(\mathbf{U}+\mathbf{P})\mathbf{W}^{(v)} \\
\tilde{\mathbf{O}} &= \text{LayerNorm}_2(\tilde{\mathbf{M}}+\mathbf{U}+\mathbf{P}) \\
\tilde{\mathbf{F}} &= \text{ReLU}(\tilde{\mathbf{O}}\mathbf{W}^{(f_1)}+\mathbf{b}^{(f_1)})\mathbf{W}^{(f_2)}+\mathbf{b}^{(f_2)} \\
\tilde{\mathbf{Z}} &= \text{LayerNorm}_1(\tilde{\mathbf{O}}+\tilde{\mathbf{F}}).
\end{aligned} \tag{3}$$

Modifying Attention Matrix (MAM). Instead of adding position embeddings, other approaches directly modify the attention matrix. For example, by adding absolute or relative position biases to the matrix, see Figure 2. In fact, one big effect of adding position embeddings is that it modifies the attention matrix as follows

$$\hat{\mathbf{A}} \sim \underbrace{\mathbf{U}\mathbf{W}^{(q)}\mathbf{W}^{(k)\intercal}\mathbf{U}^\intercal}_{\text{unit-unit }\sim\mathbf{A}} + \underbrace{\mathbf{P}\mathbf{W}^{(q)}\mathbf{W}^{(k)\intercal}\mathbf{U}^\intercal + \mathbf{U}\mathbf{W}^{(q)}\mathbf{W}^{(k)\intercal}\mathbf{P}^\intercal}_{\text{unit-position}} + \underbrace{\mathbf{P}\mathbf{W}^{(q)}\mathbf{W}^{(k)\intercal}\mathbf{P}^\intercal}_{\text{position-position}}. \tag{4}$$

As indicated, the matrix $\mathbf{A}$ can then be decomposed into unit-unit interactions as well as unit-position and position-position interactions. We write $\sim$ as we omit the scaling factor for the attention matrix for simplicity.

As adding position embeddings results in a modification of the attention matrix, these two approaches are highly interlinked. Still, we make a distinction between these two approaches for two reasons: i) While adding position embeddings results, among other effects, in a modified attention matrix, MAM *only* modifies the attention matrix. ii) APE involves learning embeddings for position information whereas MAM is often interpreted as adding or multiplying scalar biases to the attention matrix $\mathbf{A}$, see Figure 2

3.3 INTEGRATION

In theory there is many possibilities where to inject position information, but in practice the information is either integrated in the input, at each attention matrix or directly before the output. When adding position information at the beginning, it only affects the first layer and has to be propagated to upper layers indirectly. Often, APE is only added at the beginning, and MAM approaches are used for each layer and attention head.

4 CURRENT POSITION INFORMATION MODELS

In this section we provide an overview of current position information models. Note that we use the term *position information model* to refer to a method that integrates position information, the term *position encoding* refers to a position ID associated to units (e.g., numbered from 0 to t, or

			Reference Point	
		Absolute	**Absolute & Relative**	**Relative**
Topic	**Sequential**	(Devlin et al., 2019) (Kitaev et al., 2020) (Liu et al., 2020b) (Press et al., 2020) (Wang et al., 2020) (Dehghani et al., 2019)	(Shaw et al., 2018) (Ke et al., 2020) (Dufter et al., 2020) (He et al., 2020)	(Dai et al., 2019) (Raffel et al., 2020) (Wu et al., 2020) (Huang et al., 2020) (Shen et al., 2018) (Neishi & Yoshinaga, 2019)
	Sinusoidal	(Vaswani et al., 2017) (Li et al., 2019)		(Yan et al., 2019)
	Graphs	(Shiv & Quirk, 2019)	(Wang et al., 2019)	(Zhu et al., 2019) (Cai & Lam, 2020) (Schmitt et al., 2021)
	Decoder	(Takase & Okazaki, 2019) (Oka et al., 2020) (Bao et al., 2019)		
	Crosslingual	(Artetxe et al., 2020) (Ding et al., 2020) (Liu et al., 2020a) (Liu et al., 2020c)		
	Analysis	(Yang et al., 2019) (Wang & Chen, 2020)	(Rosendahl et al., 2019) (Wang et al., 2021)	

Table 1: Overview and categorization of papers dealing with position information. We categorize along two dimensions: topic, i.e., a tag that describes the main topic of a paper, and which reference point is used for the position encodings.

assigning relative distances). A *position embedding* then refers to a numerical vector associated with a position encoding. We cluster position information models along two dimensions: *reference point* and *topic*, see Table 1. The following sections deal with each topic and within each topic we discuss approaches with different reference point. Table 2 provides more details for each method and aims at making comparisons easier.

4.1 Sequential

4.1.1 Absolute Position Encodings

The original Transformer paper considered absolute position encodings. One of the two approaches proposed by Vaswani et al. (2017) follows Gehring et al. (2017) and learns a position embedding matrix $\mathbf{P} \in \mathbb{R}^{t_{\max} \times d}$ corresponding to the absolute positions $1, 2, \dots, t_{\max} - 1, t_{\max}$ in a sequence. This matrix is simply added to the unit embeddings $\mathbf{U}$ before they are fed to the Transformer model (APE).

In the simplest case, the position embeddings are randomly initialized and then adapted during training of the network (Gehring et al., 2017; Vaswani et al., 2017; Devlin et al., 2019). Gehring et al. (2017) find that adding position embeddings only help marginally in a convolutional neural network. A Transformer model without any position information, however, performs much worse for some tasks (e.g., Wang et al., 2019, Wang et al., 2021).

For very long sequences, i.e., large $t_{\max}$, the number of parameters added with $\mathbf{P}$ is significant. Thus, Kitaev et al. (2020) proposed a more parameter-efficient factorization called *axial position embeddings*. Although their method is not described in the paper, a description can be found in their code.[2] Intuitively, they have one embedding that marks a larger segment and a second embedding that indicates the position within each segment, see Figure 3 for an overview. More specifically, the matrix $\mathbf{P}$ gets split into two embedding matrices $\mathbf{P}^{(1)} \in \mathbb{R}^{t_1 \times d_1}$, $\mathbf{P}^{(2)} \in \mathbb{R}^{t_2 \times d_2}$ with $d = d_1 + d_2$

[2] e.g., `https://huggingface.co/transformers/model_doc/reformer.html`

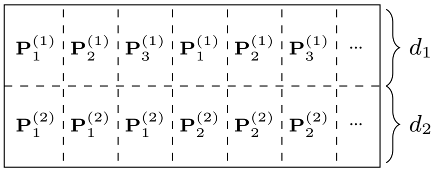


Figure 3: Overview of the structure of $\mathbf{P}$ with axial Position Embeddings by Kitaev et al. (2020). They use two position embeddings, one of which can be interpreted as encoding a segment (bottom, $\mathbf{P}^{(2)}$) and the position within that segment (top, $\mathbf{P}^{(1)}$). This factorization is more parameter efficient, especially for long sequences.

and $t_{\max} = t_1 t_2$. Then

$$\mathbf{P}_{tj} = \begin{cases} \mathbf{P}^{(1)}_{r,j} \text{ if } j \leq d_1, \ r = t \bmod t_1 \\ \mathbf{P}^{(2)}_{s,j-d_1} \text{ if } j > d_1, \ s = \lfloor \frac{t}{t_1} \rfloor. \end{cases} \tag{5}$$

Liu et al. (2020b) argue that position embeddings should be parameter-efficient, data-driven, and should be able to handle sequences that are longer than any sequence in the training data. They propose a new model called *flow-based Transformer* or *FLOATER*, where they model position information with a continuous dynamic model. More specifically, consider $\mathbf{P}$ as a sequence of timesteps $p_1, p_2, \dots, p_{t_{\max}}$. They suggest to model position information as a continuous function $p : \mathbb{R}_+ \to \mathbb{R}^d$ with

$$p(t) = p(s) + \int_s^t h\left(\tau, p(\tau), \theta_h\right) \mathrm{d}\tau \tag{6}$$

for $0 \leq s < t$ with some initial value for $p(0)$, where h is some function, e.g., a neural network with parameters θ_h. In the simplest case they then define $p_i := p(i\Delta t)$ for some fixed offset Δt. They experiment both with adding the information only in the first layer and at each layer (layerwise APE). Even though they share parameters across layers, they use different initial values $p(0)$ and thus have different position embeddings at each layer. Sinusoidal position embeddings (cf.§4.2) are a special case of their dynamic model. Further, they provide a method to use the original position embeddings of a pretrained Transformer model while adding the dynamic model during finetuning only. In their experiments they observe that FLOATER outperforms learned and sinusoidal position embeddings, especially for long sequences. Further, adding position information at each layer increases performance.

Another approach to increase the Transformer efficiency both during training and inference is to keep $t_{\max}$ small. The *Shortformer* by Press et al. (2020) caches previously computed unit representations and therefore does not need to handle a large number of units at the same time. This is made possible by what they call *position-infused attention*, where the position embeddings are added to the keys and queries, but not the values. Thus, the values are position independent and representations from previous subsequences can seamlessly be processed. More specifically, they propose

$$\begin{aligned} \tilde{\mathbf{A}} &\sim (\mathbf{U} + \mathbf{P})\mathbf{W}^{(q)} \mathbf{W}^{(k)\intercal} (\mathbf{U} + \mathbf{P})^\intercal \\ \tilde{\mathbf{M}} &= \text{SoftMax}(\tilde{\mathbf{A}}) \mathbf{U} \mathbf{W}^{(v)}. \end{aligned} \tag{7}$$

The computation of the attention matrix $\tilde{\mathbf{A}}$ still depends on absolute position encodings in Shortformer, but $\tilde{\mathbf{M}}$ does not contain it, as it is only a weighted sum of unit embeddings in the first layer. Consequently, Shortformer can attend to outputs of previous subsequences and the position information has to be added in each layer again. Press et al. (2020) report large improvements in training speed, as well as language modeling perplexity.

While the former approaches all follow the APE methodology, Wang et al. (2020) propose an alternative to simply summing position and unit embeddings. Instead of having one embedding per unit, they model the representation as a function over positions. That is, instead of feeding $\mathbf{U}_t + \mathbf{P}_t$ to the model for position t, they suggest to model the embedding of unit u as a function $g^{(u)} : \mathbb{N} \to \mathbb{R}^d$ such that the unit has a different embedding depending on the position at which it occurs. After having proposed desired properties for such functions (position-free offset and boundedness), they introduce *complex-valued unit embeddings* where their k-th component is defined as follows

$$g^{(u)}(t)_k = r_k^{(u)} \exp\left(i(\omega_k^{(u)} t + \theta_k^{(u)})\right). \tag{8}$$

Then, $\mathbf{r}^{(u)}, \boldsymbol{\omega}^{(u)}, \boldsymbol{\theta}^{(u)} \in \mathbb{R}^d$ are learnable parameters that define the unit embedding for the unit u. Their approach can also be interpreted as having a word embedding, parameterized by $\mathbf{r}^{(u)}$, that is component-wise multiplied with a position embedding, parameterized by $\boldsymbol{\omega}^{(u)}, \boldsymbol{\theta}^{(u)}$. They test these position-sensitive unit embeddings not only on Transformer models, but also on static embeddings, LSTMs, and CNNs, and observe large improvements.

4.1.2 Relative Position Encodings

Among the first, Shaw et al. (2018) introduced an alternative method for incorporating both absolute and *relative position encodings*. In their absolute variant they propose to change the computation to

$$\mathbf{A}_{ts} \sim \mathbf{U}_t{}^\intercal \mathbf{W}^{(q)} \left(\mathbf{W}^{(k)\intercal} \mathbf{U}_s + \mathbf{a}^{(k)}_{(t,s)} \right) \tag{9}$$

where $\mathbf{a}^{(k)}_{(t,s)} \in \mathbb{R}^d$ models the interaction between position t and s. Further they modify the computation of the values to

$$\mathbf{M}_t = \sum_{s=1}^{t_{\max}} \text{SoftMax}(\mathbf{A})_{ts} \left(\mathbf{W}^{(v)\intercal} \mathbf{U}_s + \mathbf{a}^{(v)}_{(t,s)} \right) \tag{10}$$

where $\mathbf{a}^{(v)}_{(t,s)} \in \mathbb{R}^d$ models again the interaction. While it cannot directly be compared with the effect of simple addition of position embeddings, they roughly omit the position-position interaction and have only one unit-position term. In addition, they do not share the projection matrices but directly model the pairwise position interaction with the vectors $\mathbf{a}$. In an ablation analysis they found that solely adding $\mathbf{a}^{(k)}_{(t,s)}$ might be sufficient.

To achieve relative positions they simply set

$$\mathbf{a}^{(k)}_{(t,s)} := \mathbf{w}^{(k)}_{(clip(s-t,r))}, \tag{11}$$

where $clip(x, r) = \max(-r, \min(r, x))$ and $\mathbf{w}^{(k)}_{(t)} \in \mathbb{R}^d$ for $-r \leq t \leq r$ for a maximum relative distance r. Analogously for $\mathbf{a}^{(v)}_{(t,s)}$. To reduce space complexity, they share the parameters across attention heads. While it is not explicitly mentioned in their paper we assume that they add the position information in each layer but do not share the parameters. The authors find that relative position embeddings perform better in machine translation and the combination of absolute and relative embeddings does not improve the performance.

Dai et al. (2019) propose the *Transformer XL* model. The main objective is to cover long sequences and to overcome the constraint of having a fixed-length context. To this end they fuse Transformer models with recurrence. This requires special handling of position information and thus a new position information model. At each attention head they adjust the computation of the attention matrix to

$$\mathbf{A}_{ts} \sim \underbrace{\mathbf{U}_t{}^\intercal \mathbf{W}^{(q)} \mathbf{W}^{(k)\intercal} \mathbf{U}_s}_{\text{content-based addressing}} + \underbrace{\mathbf{U}_t{}^\intercal \mathbf{W}^{(q)} \mathbf{V}^{(k)\intercal} \mathbf{R}_{t-s}}_{\text{content-dependent position bias}} + \underbrace{\mathbf{b}^\intercal \mathbf{W}^{(k)\intercal} \mathbf{U}_s}_{\text{global content bias}} + \underbrace{\mathbf{c}^\intercal \mathbf{V}^{(k)\intercal} \mathbf{R}_{t-s}}_{\text{global position bias}}, \tag{12}$$

where $\mathbf{R} \in \mathbb{R}^{\tau \times d}$ is a sinusoidal position embedding matrix as in (Vaswani et al., 2017) and $\mathbf{b}, \mathbf{c} \in \mathbb{R}^d$ are learnable parameters. They use different projection matrices for the relative positions, namely $\mathbf{V}^{(k)} \in \mathbb{R}^{d \times d}$. Note that Transformer-XL is unidirectional and thus $\tau = t_m + t_{\max} - 1$, where t_m is the memory length in the model. Furthermore they add this mechanism to all attention heads and layers, while sharing the position parameters across layers and heads.

There are more approaches that explore variants of Eq. 4. Ke et al. (2020) propose *untied position embeddings*. More specifically, they simply put $\mathbf{U}$ into the Transformer and then modify the attention matrix $\mathbf{A}$ in the first layer by adding a *position bias*

$$\mathbf{A} \sim \mathbf{U}\mathbf{W}^{(q)} \mathbf{W}^{(k)\intercal} \mathbf{U}^\intercal + \mathbf{P}\mathbf{V}^{(q)} \mathbf{V}^{(k)\intercal} \mathbf{P}^\intercal. \tag{13}$$

Compared to Eq. 4 they omit the unit-position interaction terms and use different projection matrices, $\mathbf{V}^{(q)}, \mathbf{V}^{(k)} \in \mathbb{R}^{d \times d}$ for units and positions. Similarly, they add relative position embeddings by

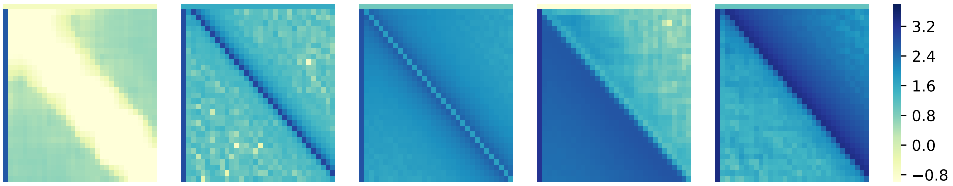


Figure 4: Figure by Ke et al. (2020). Their position bias is independent of the input and can thus be easily visualized. The absolute position biases learn intuitive patterns as shown above. Patterns (from left to right) include ignoring position information, attending locally, globally, to the left, and to the right. One can clearly see the untied position bias for the first token, i.e., the [CLS] token, on the left and top of each matrix.

adding a scalar value. They add a matrix $\mathbf{A}^r \in \mathbb{R}^{t_{\max} \times t_{\max}}$, where $\mathbf{A}^r_{t,s} = \mathbf{b}_{t-s+t_{\max}}$ and $\mathbf{b} \in \mathbb{R}^{2t_{\max}}$ are learnable parameters, which is why we categorize this approach as MAM. A very similar idea with relative position encodings has also been used by Raffel et al. (2020). They further argue that the [CLS] token has a special role and thus they replace the terms $\mathbf{P}_1{}^\intercal \mathbf{V}^{(q)} \mathbf{V}^{(k)}{}^\intercal \mathbf{P}_s$ with a single parameter θ_1 and analogously $\mathbf{P}_t{}^\intercal \mathbf{V}^{(q)} \mathbf{V}^{(k)}{}^\intercal \mathbf{P}_1$ with θ_2, i.e., they disentangle the position of the [CLS] token from the other position interactions. They provide theoretical arguments that their absolute and relative position embeddings are complementary. Indeed, in their experiments the combination of relative and absolute embeddings boosts performance on the GLUE benchmark. They provide an analysis of the position biases learned by their network, see Figure 4.

A similar idea has been explored in (Dufter et al., 2020), where in a more limited setting, i.e., in the context of PoS-tagging, learnable *absolute or relative position biases* are learned instead of full position embeddings.

Complementary to that line of research is a method by He et al. (2020): In their model *DeBERTa*, they omit the position-position interaction and focus on unit-position interactions. However, their embeddings are still untied or disentangled as they use different projection matrices for unit and position embeddings. They introduce relative position embeddings $\mathbf{A}^r \in \mathbb{R}^{2t_{\max} \times d}$ and define

$$\delta(t,s) = \begin{cases} 0 & \text{if} \quad t-s \leq -t_{\max} \\ 2t_{\max}-1 & \text{if} \quad t-s \geq t_{\max} \\ t-s+t_{\max} & \text{else.} \end{cases} \tag{14}$$

They then compute

$$\mathbf{A}_{ts} \sim \mathbf{U}_t{}^\intercal \mathbf{W}^{(q)} \mathbf{W}^{(k)}{}^\intercal \mathbf{U}_s + \mathbf{U}_t{}^\intercal \mathbf{W}^{(q)} \mathbf{V}^{(k)}{}^\intercal \mathbf{A}^r_{\delta(t,s)} + \mathbf{A}^r_{\delta(s,t)}{}^\intercal \mathbf{V}^{(q)} \mathbf{W}^{(k)}{}^\intercal \mathbf{U}_s \tag{15}$$

as the attention in each layer. While they share the weights of $\mathbf{A}^r \in \mathbb{R}^{2t_{\max} \times d}$ across layers, the weight matrices are separate for each attention head and layer. In addition they change the scaling factor from $\sqrt{1/d_h}$ to $\sqrt{1/(3d_h)}$. In the last layer they inject a traditional absolute position embedding matrix $\mathbf{P} \in \mathbb{R}^{t_{\max} \times d}$. Thus they use both MAM and APE. They want relative encodings to be available in every layer but argue that the model should be reminded of absolute encodings right before the masked language model prediction. In their example sentence *a new store opened beside the new mall* they argue that *store* and *mall* have similar relative positions to *new* and thus absolute positions are required for predicting masked units.

The following two approaches do not work with embeddings, but instead propose a direct multiplicative smoothing on the attention matrix and can thus be categorized as MAM. Wu et al. (2020) propose a smoothing based on relative positions in their model *DA-Transformer*. They consider the matrix of absolute values of relative distances $\mathbf{R} \in \mathbb{N}^{t_{\max} \times t_{\max}}$ where $\mathbf{R}_{ts} = |t-s|$. For each attention head m they obtain $\mathbf{R}^{(m)} = w^{(m)} \mathbf{R}$ with $w^{(m)} \in \mathbb{R}$ being a learnable scalar parameter. They then compute

$$\mathbf{A} \sim \text{ReLU}\left((\mathbf{X} \mathbf{W}^{(q)} \mathbf{W}^{(k)}{}^\intercal \mathbf{X}^\intercal) \circ \hat{\mathbf{R}}^{(m)} \right), \tag{16}$$

where $\hat{\mathbf{R}}^{(m)}$ is a rescaled version of $\mathbf{R}^{(m)}$ and $\circ$ is component-wise multiplication. For rescaling they use a learnabel sigmoid function, i.e.,

$$\hat{\mathbf{R}}^{(m)} = \frac{1+\exp(v^{(m)})}{1+\exp(v^{(m)} - \mathbf{R}^{(m)})}. \tag{17}$$

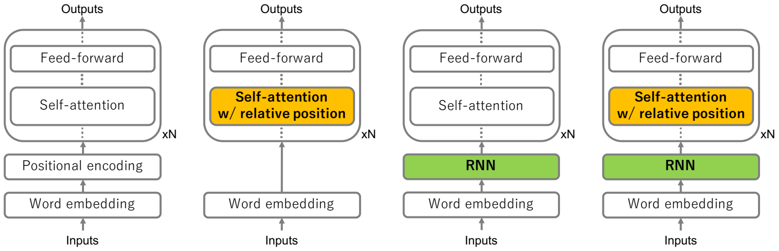


Figure 5: Figure by Neishi & Yoshinaga (2019). Overview of the architecture when using an RNN for learning position information. They combine their RNN-Transformer with relative position embeddings by Shaw et al. (2018) in a model called *RR-Transformer* (far right).

Overall, they only add $2h$ parameters as each head has two learnable parameters. Intuitively, they want to allow each attention head to choose whether to attend to long range or short range dependencies. Note that their model is direction-agnostic. The authors observe improvements for text classification both over vanilla Transformer, relative position encodings by (Shaw et al., 2018), Transformer-XL (Dai et al., 2019) and TENER (Yan et al., 2019).

Related to the DA-Transformer, Huang et al. (2020) review absolute and relative position embedding methods and propose four position information models with relative position encodings: (i) Similar to (Wu et al., 2020) they scale the attention matrix by

$$\mathbf{A} \sim (\mathbf{X}\mathbf{W}^{(q)}\mathbf{W}^{(k)\intercal}\mathbf{X}^\intercal) \circ \mathbf{R}, \tag{18}$$

where $\mathbf{R}_{ts} = \mathbf{r}_{|s-t|}$ and $\mathbf{r} \in \mathbb{R}^{t_{\max}}$ is a learnable vector. (ii) They consider $\mathbf{R}_{ts} = \mathbf{r}_{s-t}$ as well to distinguish different directions. (iii) As a new variant they propose

$$\mathbf{A}_{ts} \sim \mathit{sum\_product}(\mathbf{W}^{(q)\intercal}\mathbf{X}_t, \mathbf{W}^{(k)\intercal}\mathbf{X}_s, \mathbf{r}_{s-t}), \tag{19}$$

where $\mathbf{r}_{s-t} \in \mathbb{R}^d$ are learnable parameters and *sum_product* is the scalar product extended to three vectors. (iv) Last, they extend the method by Shaw et al. (2018) to not only add relative positions to the key, but also to the query in Eq. 9, and in addition remove the position-position interaction. More specifically,

$$\mathbf{A}_{ts} \sim \left(\mathbf{W}^{(q)\intercal}\mathbf{U}_t + \mathbf{r}_{s-t}\right)^\intercal \left(\mathbf{W}^{(k)\intercal}\mathbf{U}_s + \mathbf{r}_{s-t}\right) - \mathbf{r}_{s-t}{}^\intercal\mathbf{r}_{s-t} \tag{20}$$

On several GLUE tasks (Wang et al., 2018) they find that the last two methods perform best.

The next two approaches are not directly related to relative position encodings, but they can be interpreted as using relative position information. Shen et al. (2018) do not work directly with a Transformer model. Still they propose *Directional Self-Attention Networks (Di-SAN)*. Besides other differences to plain self-attention (e.g., multidimensional attention), they notably mask out the upper/lower triangular matrix or the diagonal in $\mathbf{A}$ to achieve non-symmetric attention matrices. Allowing attention only in a specific direction does not add position information directly, but still makes the attention mechanism position-aware to some extent, i.e., enables the model to distinguish directions.

Neishi & Yoshinaga (2019) argue that recurrent neural networks (RNN) in form of Gated recurrent units (GRU) (Cho et al., 2014) are able to encode relative positions. Thus they propose to replace position encodings by adding a single GRU layer on the input before feeding it to the Transformer, see Figure 5. With their models called *RRN-Transformer* they observe comparable performance compared to position embeddings, however for longer sequences the GRU yields better performance. Combining their approach with the method by Shaw et al. (2018) improves performance further, a method they call *RR-Transformer*.

4.2 Sinusoidal

Another line of work experiments with sinusoidal values that are kept fixed during training to encode position information in a sequence. The approach proposed by Vaswani et al. (2017) is an instance

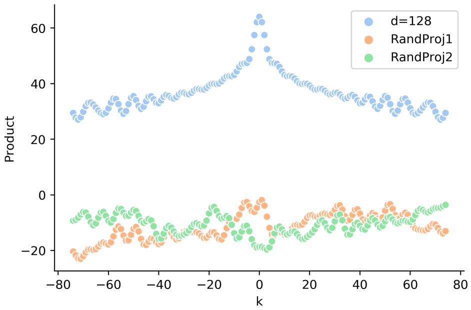


Figure 6: Figure by Yan et al. (2019). Shown is the value of dot product on the y-axis between sinusoidal position embeddings with different relative distance (k) shown on the x-axis. The blue line shows the dot product without projection matrices and the other two lines with random projections. Relative position without directionality can be encoded without projection matrices, but with the projections this information is destroyed.

of the absolute position APE pattern, called *sinusoidal position embeddings*, defined as

$$\mathbf{P}_{tj} = \begin{cases} \sin(10000^{-\frac{j}{d}} t) \text{ if } j \text{ even} \\ \cos(10000^{-\frac{(j-1)}{d}} t) \text{ if } j \text{ odd.} \end{cases} \tag{21}$$

They observe comparable performance between learned absolute position embeddings and their sinusoidal variant. However, they hypothesize that the sinusoidal structure helps for long range dependencies. This is for example verified by Liu et al. (2020b). An obvious advantage is also that they can handle sequences of arbitrary length, which most position models cannot. They are usually kept fixed and are not changed during training and thus very parameter-efficient.

Indeed, sinusoidal position embeddings exhibit useful properties in theory. Yan et al. (2019) investigate the dot product of sinusoidal position embeddings and prove important properties: (i) The dot product of two sinusoidal position embeddings depends only on their relative distance. That is, ${\mathbf{P}_t}^\intercal \mathbf{P}_{t+r}$ is independent of t. (ii) ${\mathbf{P}_t}^\intercal \mathbf{P}_{t-r} = {\mathbf{P}_t}^\intercal \mathbf{P}_{t+r}$, which means that sinusoidal position embeddings are unaware of directions. However, in practice the sinusoidal embeddings are projected with two different projection matrices, which destroys these properties, see Figure 6. Thus, Yan et al. (2019) propose a *Direction- and Distance-aware Attention* in their model *TENER* that maintains these properties and can, in addition, distinguish between directionality. They compute

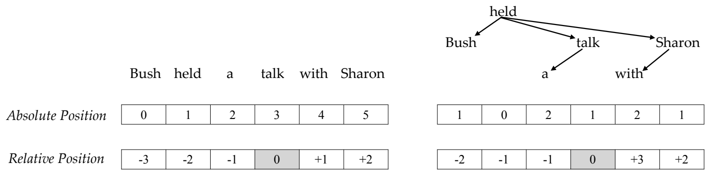
$$\mathbf{A}_{ts} \sim \underbrace{{\mathbf{U}_t}^\intercal \mathbf{W}^{(q)} {\mathbf{W}^{(k)}}^\intercal \mathbf{U}_s}_{\text{unit-unit}} + \underbrace{{\mathbf{U}_t}^\intercal \mathbf{W}^{(q)} \mathbf{R}_{t-s}}_{\text{unit-relative position}} + \underbrace{\mathbf{u}^\intercal {\mathbf{W}^{(k)}}^\intercal \mathbf{U}_s}_{\text{unit-bias}} + \underbrace{\mathbf{v}^\intercal \mathbf{R}_{t-s}}_{\text{relative position bias}} , \tag{22}$$

where $\mathbf{R}_{t-s} \in \mathbb{R}^d$ is a sinusoidal relative position vector defined as

$$\mathbf{R}_{t-s,j} = \begin{cases} \sin((t-s)10000^{-\frac{j}{d}}) \text{ if } j \text{ even} \\ \cos((t-s)10000^{-\frac{(j-1)}{d}}) \text{ if } j \text{ odd,} \end{cases} \tag{23}$$

and $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$ are learnable parameters for each head and layer. In addition, they set $\mathbf{W}^{(k)}$ to the identity matrix and omit the scaling factor $\frac{1}{\sqrt{d}}$ as they find that it performs better. Overall, the authors find massive performance increases for named entity recognition compared to standard Transformer models.

Dehghani et al. (2019) use a variant of sinusoidal position embeddings in their *Universal Transformer*. In their model they combine Transformers with the recurrent inductive bias of recurrent neural networks. The basic idea is to replace the layers of a Transformer model with a single layer that is recurrently applied to the input, that is they share the weights across layers. In addition they propose conditional computation where they can halt or continue computation for each position individually. When l denotes their lth application of the Transformer layer to the input, they add the



	Bush	held	a	talk	with	Sharon
Absolute Position	0	1	2	3	4	5
Relative Position	-3	-2	-1	0	+1	+2

(a) **Sequential** Position Encoding

Absolute Position	1	0	2	1	2	1
Relative Position	-2	-1	-1	0	+3	+2

(b) **Structural** Position Encoding

Figure 7: Figure by Wang et al. (2019). They compute absolute and relative encodings not based on the sequential order of a sentence (left), but based on a dependency tree (right). Both absolute and relative encodings can be created.

position embeddings as follows

$$\mathbf{P}^{l}_{t,j} = \begin{cases} \sin(10000^{-\frac{j}{d}}t) + \sin(10000^{-\frac{j}{d}}l) \text{ if } j \text{ even} \\ \cos(10000^{-\frac{j-1}{d}}t) + \cos(10000^{-\frac{j-1}{d}}l) \text{ if } j \text{ odd.} \end{cases} \tag{24}$$

Their approach can be interpreted as adding sinusoidal position embeddings at each layer.

Li et al. (2019) argue that the variance of sinusoidal position embeddings per position across dimensions varies greatly: for small positions it is rather small and for large positions it is rather high. The authors consider this a harmful property and propose *maximum variances position embeddings (mvPE)* as a remedy. They change the computation to

$$\mathbf{P}_{tj} = \begin{cases} \sin(10000^{-\frac{j}{d}}kt) \text{ if } j \text{ even} \\ \cos(10000^{-\frac{j-1}{d}}kt) \text{ if } j \text{ odd.} \end{cases} \tag{25}$$

They claim that suitable values for the hyperparameter k are $k > 1000$.

4.3 GRAPHS

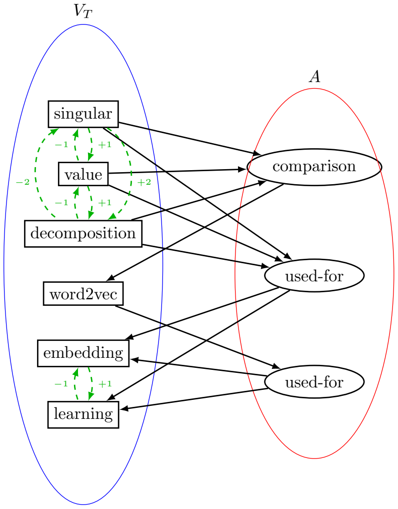
In the following section, we will take a look at position information models for graphs, i.e., cases where Transformers have been used for genuine graph input as well as cases where the graph is used as a sentence representation, e.g., a dependency graph. We distinguish two types of graph position models according to the assumptions they make about the graph structure: positions in hierarchies (trees) and arbitrary graphs.

4.3.1 HIERARCHIES (TREES)

Wang et al. (2019) propose *structural position representations* (SPR), see Figure 7. This means that instead of treating a sentence as a sequence of information, they perform dependency parsing and compute distances on the parse tree (dependency graph).[3] We can distinguish two settings: (i) Analogously to absolute position encodings in sequences, where unit u_t is assigned position t, absolute SPR assigns u_t the position $abs(u_t) := d_{\text{tree}}(u_t, \text{ROOT})$ where ROOT is the root of the dependency tree (the main verb of the sentence) and $d_{\text{tree}}(x, y)$ is the path length between x and y in the tree. (ii) For the relative SPR between the units u_t, u_s, they define $rel(u_t, u_s) = abs(u_t) - abs(u_s)$ if u_t is on the path from u_s to the root or vice versa. Otherwise, they use $rel(u_t, u_s) = sgn(t - s)(abs(u_t) + abs(u_s))$. So we see that SPR does not only assume the presence of a graph hierarchy but also needs a strict order to be defined on the graph nodes, because rel equally encodes sequential relative position. This makes SPR a suitable choice for working with dependency graphs but renders SPR incompatible with other tree structures.

Having defined the position of a node in a tree, Wang et al. (2019) inject their SPR via sinusoidal APE for absolute and via learned embeddings + MAM for relative positions. It is noteworthy, that Wang et al. (2019) achieve their best performance by combining both variants of SPR with sequential position information and that SPR as sole sentence representation, i.e., without additional sequential information, leads to a large drop in performance.

[3] Dependency parsers usually do not operate on subwords. So subwords are assigned the position of their main word.



	V_T						A		
	s	v	d	w	e	l	c	u1	u2
s	0	4	5	2	2	2	1	1	3
v	-4	0	4	2	2	2	1	1	3
d	-5	-4	0	2	2	2	1	1	3
w	-2	-2	-2	0	2	2	-1	∞	1
e	-2	-2	-2	-2	0	4	-3	-1	-1
l	-2	-2	-2	-2	-4	0	-3	-1	-1
c	-1	-1	-1	1	3	3	0	∞	2
u1	-1	-1	-1	∞	1	1	∞	0	∞
u2	-3	-3	-3	-1	1	1	-2	∞	0

Figure 8: Figure from (Schmitt et al., 2021), showing their definition of relative position encodings in a graph based on the lengths of shortest paths. ∞ means that there is no path between two nodes. Numbers higher than 3 and lower than -3 represent sequential relative position in multi-token node labels (dashed green arrows).

Shiv & Quirk (2019) propose alternative absolute *tree position encodings* (TPE). They draw inspiration from the mathematical properties of sinusoidals but do not use them directly like Wang et al. (2019). Also unlike SPR, their position encodings consider the full path from a node to the root of the tree and not only its length, thus assigning every node a unique position. This is more in line with the spirit of absolute sequential position models (§4.1.1). The first version of TPE is parameter-free: The path from the root of an n-ary tree to some node is defined as the individual decisions that lead to the destination, i.e., which of the n children is the next to be visited at each intermediate step. These decisions are encoded as one-hot vectors of size n. The whole path is simply the concatenation of these vectors (padded with 0s for shorter paths). In a second version, multiple instances of parameter-free TPE are concatenated and each one is weighted with a different learned parameter. After scaling and normalizing these vectors, they are added to the unit embeddings before the first Transformer layer (APE).

4.3.2 Arbitrary Graphs

Zhu et al. (2019) were the first to propose a Transformer model capable of processing arbitrary graphs. Their position information model solely defines the relative position between nodes and incorporates this information by manipulating the attention matrix (MAM):

$$\begin{aligned}\mathbf{A}_{ts} &\sim \mathbf{U}_t{}^{\intercal}\mathbf{W}^{(q)}\left(\mathbf{W}^{(k)\intercal}\mathbf{U}_s + \mathbf{W}^{(r)\intercal}\mathbf{r}_{(t,s)}\right) \\ \mathbf{M}_t &= \sum_{s=1}^{t_{\max}} \mathrm{SoftMax}(\mathbf{A})_{ts}\left(\mathbf{W}^{(v)\intercal}\mathbf{U}_s + \mathbf{W}^{(f)\intercal}\mathbf{r}_{(t,s)}\right)\end{aligned} \tag{26}$$

where $\mathbf{W}^{(r)}, \mathbf{W}^{(f)} \in \mathbb{R}^{d\times d}$ are additional learnable parameters, and $\mathbf{r}_{(t,s)} \in \mathbb{R}^d$ is a representation of the sequence of edge labels and special edge direction symbols (↑ and ↓) on the shortest path between the nodes u_t and u_s. Zhu et al. (2019) experiment with 5 different ways of computing $\mathbf{r}$, where the best performance is achieved by two approaches: (i) A CNN with d kernels of size 4 that convolutes the embedded label sequence $\mathbf{U}^{(r)}$ into $\mathbf{r}$ (cf. Kalchbrenner et al., 2014) and (ii) a one-layer self-attention module with sinusoidal position embeddings $\mathbf{P}$ (cf. §4.2):

$$\begin{aligned}\mathbf{A}^{(r)} &\sim (\mathbf{U}^{(r)}+\mathbf{P})\mathbf{W}^{(qr)}\mathbf{W}^{(kr)\intercal}(\mathbf{U}^{(r)}+\mathbf{P})^\intercal\\ \mathbf{M}^{(r)} &= \text{SoftMax}(\mathbf{A}^{(r)})(\mathbf{U}^{(r)}+\mathbf{P})\mathbf{W}^{(vr)}\\ \mathbf{a}^{(r)} &= \text{SoftMax}(\mathbf{W}^{(r_2)}tanh(\mathbf{W}^{(r_1)}\mathbf{M}^{(r)\intercal}))\\ \mathbf{r} &= \sum_{k=1}^{t_{\max}^{(r)}} a_k^{(r)}\mathbf{M}_k^{(r)}\end{aligned} \tag{27}$$

with $\mathbf{W}^{(r_1)} \in \mathbb{R}^{d_r \times d}, \mathbf{W}^{(r_2)} \in \mathbb{R}^{1\times d_r}$ additional model parameters. While there is a special symbol for the empty path from one node to itself, this method implicitly assumes that there is always at least one path between any two nodes. While it is easily possible to extend this work to disconnected graphs by introducing another special symbol, the effect on performance is unclear.

Cai & Lam (2020) also define relative position in a graph based on shortest paths. They differ from the former approach in omitting the edge direction symbols and using a bidirectional GRU (Cho et al., 2014), to aggregate the label information on the paths (cf. the RNN-Transformer described by Neishi & Yoshinaga (2019)). After linearly transforming the GRU output, it is split into a forward and a backward part: $[\mathbf{r}_{t\to s};\mathbf{r}_{s\to t}] = \mathbf{W}^{(r)}GRU(\dots)$. These vectors are injected into the model in a variant of APE:

$$\begin{aligned}\mathbf{A}_{st} &\sim (\mathbf{U}_s+\mathbf{r}_{s\to t})^\intercal\mathbf{W}^{(q)}\mathbf{W}^{(k)\intercal}(\mathbf{U}_t+\mathbf{r}_{t\to s})\\ &= \underbrace{\mathbf{U}_s{}^\intercal\mathbf{W}^{(q)}\mathbf{W}^{(k)\intercal}\mathbf{U}_t}_{\text{content-based addressing}} + \underbrace{\mathbf{U}_s{}^\intercal\mathbf{W}^{(q)}\mathbf{W}^{(k)\intercal}\mathbf{r}_{t\to s}}_{\text{source relation bias}}\\ &\quad+ \underbrace{\mathbf{r}_{s\to t}{}^\intercal\mathbf{W}^{(q)}\mathbf{W}^{(k)\intercal}\mathbf{U}_t}_{\text{target relation bias}} + \underbrace{\mathbf{r}_{s\to t}{}^\intercal\mathbf{W}^{(q)}\mathbf{W}^{(k)\intercal}\mathbf{r}_{t\to s}}_{\text{universal relation bias}}\end{aligned} \tag{28}$$

It is noteworthy that Cai & Lam (2020) additionally include absolute SPR (see §4.3.1) in their model to exploit the hierarchical structure of the AMR (abstract meaning representation) graphs they evaluate on. It is unclear, which position model has more impact on performance.

Schmitt et al. (2021) avoid computational overhead in their *Graformer* model by defining relative position encodings in a graph as the length of shortest paths instead of the sequence of edge labels (see Figure 8 for an example):

$$r_{(t,s)} = \begin{cases}\infty, & \text{if there is no path between } t, s\\ \begin{array}{l}\text{sequential relative position of } u_t, u_s\\ \text{shifted by a constant to avoid clashes,}\end{array} & \text{if subwords } u_t, u_s \text{ belong to the same main word}\\ d_{\text{graph}}(t,s), & \text{if } d_{\text{graph}}(t,s) \le d_{\text{graph}}(s,t)\\ -d_{\text{graph}}(s,t), & \text{if } d_{\text{graph}}(t,s) > d_{\text{graph}}(s,t)\end{cases}$$

where $d_{\text{graph}}(x, y)$ is the length of the shortest path between x and y. This definition also avoids the otherwise problematic case where there is more than one shortest path between two nodes because the length is always the same even if the label sequences are not. The so-defined position information is injected via learnable scalar embeddings + MAM (cf. Raffel et al., 2020).

In contrast to the other approaches, Graformer explicitly models disconnected graphs (∞) and does not add any sequential position information. Unfortunately, Schmitt et al. (2021) do not evaluate Graformer on the same tasks as the other discussed approaches, which makes a performance comparison difficult.

4.4 DECODER

Takase & Okazaki (2019) propose a simple extension to sinusoidal embeddings by incorporating sentence lengths in the position encodings of the decoder. Their motivation is to be able to control the output length during decoding and to enable the decoder to generate any sequence length independent of what lengths have been observed during training. The proposed *length-difference*

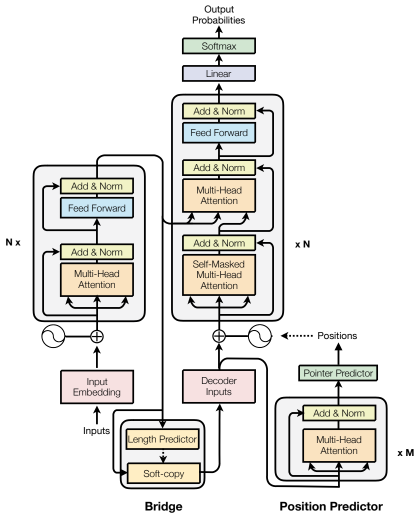


Figure 9: Figure by Bao et al. (2019). Overview of their PNAT architecture with the position prediction module. They use the encoder output to predict the output length and use a modified version as input to the decoder. The position predictor then predicts a permutation of position encodings for the output sequence.

position embeddings are

$$\mathbf{P}_{tj} = \begin{cases} \sin(10000^{-\frac{j}{d}}(l-t)) \text{ if } j \text{ even} \\ \cos(10000^{-\frac{(j-1)}{d}}(l-t)) \text{ if } j \text{ odd.} \end{cases} \tag{29}$$

where l is a given length constraint. Similarly, they propose a *length-ratio position embedding* given by

$$\mathbf{P}_{tj} = \begin{cases} \sin(l^{-\frac{j}{d}}t) \text{ if } j \text{ even} \\ \cos(l^{-\frac{(j-1)}{d}}t) \text{ if } j \text{ odd.} \end{cases} \tag{30}$$

The length constraint l is the output length of the gold standard. They observe that they can control the output length effectively during decoding. Oka et al. (2020) extended this approach by adding noise to the length constraint (adding a randomly sampled integer to the length) and by predicting the target sentence length using the Transformer model. Although, in theory, these approaches could also be used in the encoder, above work focuses on the decoder.

Bao et al. (2019) propose to predict positions word units in the decoder in order to allow for effective non-autoregressive decoding, see Figure 9. More specifically, they predict the target sentence length and a permutation from decoder inputs and subsequently reorder the position embeddings in the decoder according to the predicted permutation. Their model called *PNAT* achieves performance improvements in machine translation.

4.5 CROSSLINGUAL

Unit order across different languages is quite different. While English uses a subject-verb-object ordering (SVO) other languages use variants of this principle ordering. For example, unit order is quite flexible in German, whereas it is rather strict in English. This raises the question whether it is useful to share position information across languages.

Per default, position embeddings are shared in multilingual models (Devlin et al., 2019; Conneau et al., 2020). Artetxe et al. (2020) observe mixed results with *language specific position embeddings* in the context of transferring monolingual models to multiple languages: for most languages it helps, but for some it seems harmful. They experimented with learned absolute position embeddings as proposed in (Devlin et al., 2019).

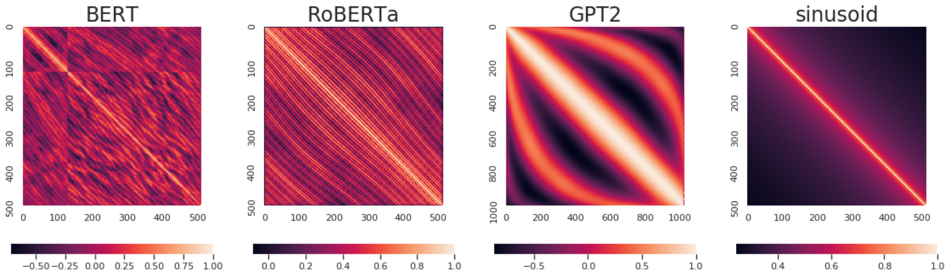


Figure 10: Figure by Wang & Chen (2020). Shown is the position-wise cosine similarity of position embeddings (APE) after pretraining. They compare three pretrained language models that use learned absolute position embeddings as in (Devlin et al., 2019), and sinusoidal positions as in (Vaswani et al., 2017). BERT shows a cutoff at 128 as it is first trained on sequences with 128 tokens and subsequently extended to longer sequences. GPT-2 exhibits the most homogenous similarity patterns.

Ding et al. (2020) use crosslingual position embeddings (*XL PE*): in the context of machine translation, they obtain reorderings of the source sentence and subsequently integrate both the original and reordered position encodings into the model and observe improvements on the machine translation task.

Liu et al. (2020a) find that position information hinders zero-shot crosslingual transfer in the context of machine translation. They remove a residual connection in a middle layer to break the propagation of position information, and thus achieve large improvements in zero-shot translation.

Similarly, Liu et al. (2020c) find that unit order information harms crosslingual transfer, e.g., in a zero-shot transfer setting. They reduce position information by a) removing the position embeddings, and replacing them with one dimensional convolutions, i.e., leveraging only local position information, b) randomly shuffling the unit order in the source language, and c) using position embeddings from a multilingual model and freezing them. Indeed they find that reducing order information with these three methods increases performance for crosslingual transfer.

4.6 Analysis

There is a range of work comparing and analyzing position information models. Rosendahl et al. (2019) analyze them in the context of machine translation. They find similar performance for absolute and relative encodings, but relative encodings are superior for long sentences. In addition, they find that the number of learnable parameters can often be reduced without performance loss.

Yang et al. (2019) evaluate the ability of recovering the original word positions after shuffling some input words. In a comparison of recurrent neural networks, Transformer models, and DiSAN (both with learned position embeddings), they find that RNN and DiSAN achieve similar performance on the word reordering task, whereas Transformer is worse. However, when trained on machine translation Transformer performs best in the word reordering task.

Wang & Chen (2020) provide an in-depth analysis of what position embeddings in large pretrained language models learn. They compare the embeddings from BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), GPT-2 (Radford et al., 2019), and sinusoidal embeddings. See Figure 10 for an analysis.

More recently, Wang et al. (2021) present an extensive analysis of position embeddings. They empirically compare 13 variants of position embeddings. Among other findings, they conclude that absolute position embeddings are favorable for classification tasks and relative embeddings perform better for span prediction tasks.

We provide a high level comparison of the discussed methods in Table 2. In this table we clustered similar approaches from a methodological point of view. The objective is to make comparisons easier and spot commonalities faster.

	Model	Ref. Point 📍	Inject. Met. 💉	Learnable 🎓	Recurring ☰	Unbound ∞	#Param
Sequences	Transformer w/ emb. (Vaswani et al., 2017)	A	APE	✓	✗	✗	$t_{\max}d$
	BERT (Devlin et al., 2019)						
	Reformer (Kitaev et al., 2020)						$(d-d_1)\frac{t_{\max}}{t_1}+d_1t_1$
	FLOATER (Liu et al., 2020b)	A	APE	✓	✓	✓	0 or more
	Shortformer (Press et al., 2020)	A	APE	✗	✓	✓	0
	Wang et al. (2020)	A	-	✓	✗	✓	$3t_{\max}d$
	Shaw et al. (2018) (abs)	A	MAM	✓	✓	✗	$2t_{\max}^2dl$
	Shaw et al. (2018) (rel)	R	MAM	✓	✓	✗	$2(2t_{\max}-1)dl$
	T5 (Raffel et al., 2020)						$(2t_{\max}-1)h$
	Huang et al. (2020)						$2dlh(t_{\max}-1)$
	DeBERTa (He et al., 2020)	B	Both	✓	✓	✗	$t_{\max}d+2d^2$
	Transformer XL (Dai et al., 2019)	R	MAM	✓	✓	✓	$2d+d^2lh$
	DA-Transformer (Wu et al., 2020)						$2h$
	TUPE (Ke et al., 2020)	B	MAM	✓	✗	✗	$2d^2+t_{\max}(d+2)$
	Dufter et al. (2020)						$t_{\max}^2h+2t_{\max}h$
	RNN-Transf. (Neishi & Yoshinaga, 2019)	R	-	✓	✗	✓	$6d^2+3d$
	Transformer w/ sin. (Vaswani et al., 2017)	A	APE	✗	✗	✓	0
	Li et al. (2019)						
	Takase & Okazaki (2019)						
	Oka et al. (2020)						
	Universal Transf. (Dehghani et al., 2019)	A	APE	✗	✓	✓	0
	Di-SAN (Shen et al., 2018)	R	MAM	✗	✓	✓	0
	TENER (Yan et al., 2019)						$2dlh$
Trees	SPR-abs (Wang et al., 2019)	A	APE	✗	✗	✓	0
	SPR-rel (Wang et al., 2019)	R	MAM	✓	✗	✗	$2(2t_{\max}+1)d$
	TPE (Shiv & Quirk, 2019)	A	APE	✓	✗	✗	$\frac{d}{D_{\max}}$
Graphs	Struct. Transformer (Zhu et al., 2019)	R	MAM	✓	✓	✓	$5d^2+(d+1)d_r$
	Graph Transformer (Cai & Lam, 2020)	R	APE	✓	✓	✓	$7d^2+3d$
	Graformer (Schmitt et al., 2021)	R	MAM	✓	✓	✗	$2(D_{\max}+1)h$

Table 2: Comparison according to several criteria: 📍 = Reference Point (**A**bsolute, **R**elative, or **B**oth); 💉 = Injection method (APE or MAM); 🎓 = Are the position representations learned during training?; ☰ = Is position information recurring at each layer vs. only before first layer?; ∞ = Can the position model generalize to longer inputs than a fixed value?; **#Param** = Number of parameters introduced by the position model (with d hidden dimension, h number of attention heads, $t_{\max}$ longest considered sequence length, l number of layers, $D_{\max}$ biggest length of all shortest paths in a graph). Approaches are clustered to avoid repetition and otherwise listed in the same order as discussed in the text. The **-** symbol means that an entry does not fit into our categories. Note that a model as a whole can combine different position models while this comparison focuses on the respective novel part(s).

5 Conclusion

We presented an overview of methods to inject position information in Transformer models. We hope our unified notation and systematic comparison (Table 2) will foster understanding and spark new ideas in this important research area.

Some open questions that consider somewhat unanswered up to now and can be part of future work include:

i) How do the current position information models compare empirically on different tasks? Some analyses papers such as Wang et al. (2021) are extensive and provide many insights. Still, not all aspects and differences of the position information models are fully understood.

ii) How important is word order for specific tasks? For many tasks, treating sentences as bag-of-words could be sufficient. Indeed, Wang et al. (2021) show that without position embeddings the performance drop for some tasks is marginal. Thus we consider it interesting to investigate for which tasks position information is essential.

iii) Can we use position information models to include more information about the structure of text? While there are many models for processing sequential and graph-based structures, there is wide range of structural information in text that is not considered currently. Some examples include tables, document layouts, list enumerations and sentence orders. Could these structures be integrated with current position information models or are new methods required for representing document structure?

Acknowledgements

This work was supported by the European Research Council (# 740516) and by the BMBF as part of the project MLWin (01IS18050). The first author was supported by the Bavarian research institute for digital transformation (bidt) through their fellowship program. We also gratefully acknowledge a Ph.D. scholarship awarded to the second author by the German Academic Scholarship Foundation (Studienstiftung des deutschen Volkes). We thank Nikita Datcenko for helpful discussions and valuable insights.

References

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. On the cross-lingual transferability of monolingual representations. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 4623–4637. Association for Computational Linguistics, 2020. URL https://www.aclweb.org/anthology/2020.acl-main.421/.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

Yu Bao, Hao Zhou, Jiangtao Feng, Mingxuan Wang, Shujian Huang, Jiajun Chen, and Lei Li. Non-autoregressive transformer by position learning. *CoRR*, abs/1911.10677, 2019. URL http://arxiv.org/abs/1911.10677.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Computer Research Repository*, 2005.14165, 2020. URL https://arxiv.org/abs/2005.14165.

Deng Cai and Wai Lam. Graph transformer for graph-to-sequence learning. *AAAI Conference on Artificial Intelligence*, 2020. URL https://aaai.org/Papers/AAAI/2020GB/AAAI-CaiD.6741.pdf.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1724–1734. ACL, 2014. doi: 10.3115/v1/d14-1179. URL `https://doi.org/10.3115/v1/d14-1179`.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 8440–8451. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.747. URL `https://doi.org/10.18653/v1/2020.acl-main.747`.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1285. URL `https://www.aclweb.org/anthology/P19-1285`.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=HyzdRiR9Y7`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://www.aclweb.org/anthology/N19-1423`.

Liang Ding, Longyue Wang, and Dacheng Tao. Self-attention with cross-lingual position representation. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 1679–1685. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.153. URL `https://doi.org/10.18653/v1/2020.acl-main.153`.

Philipp Dufter, Martin Schmitt, and Hinrich Schütze. Increasing learning efficiency of self-attention networks through direct position interactions, learnable temperature, and convoluted attention. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 3630–3636, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.324. URL `https://www.aclweb.org/anthology/2020.coling-main.324`.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1243–1252. PMLR, 2017. URL `http://proceedings.mlr.press/v70/gehring17a.html`.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020. URL `https://arxiv.org/pdf/2006.03654.pdf`.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 328–339, Melbourne, Australia, July 2018. Association for Computational

Linguistics. doi: 10.18653/v1/P18-1031. URL https://www.aclweb.org/anthology/P18-1031.

Zhiheng Huang, Davis Liang, Peng Xu, and Bing Xiang. Improve transformer models with better relative position embeddings. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pp. 3327–3335. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.298. URL https://doi.org/10.18653/v1/2020.findings-emnlp.298.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 655–665, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-1062. URL https://www.aclweb.org/anthology/P14-1062.

Guolin Ke, Di He, and Tie-Yan Liu. Rethinking the positional encoding in language pre-training. *arXiv preprint arXiv:2006.15595*, 2020. URL https://arxiv.org/pdf/2006.15595.pdf.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. URL https://arxiv.org/pdf/1607.06450.pdf.

Hailiang Li, Adele Y. C. Wang, Yang Liu, Du Tang, Zhibin Lei, and Wenye Li. An augmented transformer architecture for natural language generation tasks. In Panagiotis Papapetrou, Xueqi Cheng, and Qing He (eds.), *2019 International Conference on Data Mining Workshops, ICDM Workshops 2019, Beijing, China, November 8-11, 2019*, pp. 1–7. IEEE, 2019. doi: 10.1109/ICDMW48858.2019.9024754. URL https://doi.org/10.1109/ICDMW48858.2019.9024754.

Danni Liu, Jan Niehues, James Cross, Francisco Guzmán, and Xian Li. Improving zero-shot translation by disentangling positional information. *arXiv preprint arXiv:2012.15127*, 2020a. URL https://arxiv.org/pdf/2012.15127.pdf.

Xuanqing Liu, Hsiang-Fu Yu, Inderjit S. Dhillon, and Cho-Jui Hsieh. Learning to encode position for transformer with continuous dynamical model. *CoRR*, abs/2003.09229, 2020b. URL https://arxiv.org/abs/2003.09229.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

Zihan Liu, Genta Indra Winata, Samuel Cahyawijaya, Andrea Madotto, Zhaojiang Lin, and Pascale Fung. On the importance of word order information in cross-lingual sequence labeling. *CoRR*, abs/2001.11164, 2020c. URL https://arxiv.org/abs/2001.11164.

Masato Neishi and Naoki Yoshinaga. On the relation between position information and sentence length in neural machine translation. In Mohit Bansal and Aline Villavicencio (eds.), *Proceedings of the 23rd Conference on Computational Natural Language Learning, CoNLL 2019, Hong Kong, China, November 3-4, 2019*, pp. 328–338. Association for Computational Linguistics, 2019. doi: 10.18653/v1/K19-1031. URL https://doi.org/10.18653/v1/K19-1031.

Yui Oka, Katsuki Chousa, Katsuhito Sudoh, and Satoshi Nakamura. Incorporating noisy length constraints into transformer with length-aware positional encodings. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 3580–3585, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. URL https://www.aclweb.org/anthology/2020.coling-main.319.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://www.aclweb.org/anthology/N18-1202.

Ofir Press, Noah A. Smith, and Mike Lewis. Shortformer: Better language modeling using shorter inputs, 2020. URL https://arxiv.org/abs/2012.15832.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Jan Rosendahl, Viet Anh Khoa Tran, Weiyue Wang, and Hermann Ney. Analysis of positional encodings for neural machine translation. *IWSLT, Hong Kong, China*, 2019. URL https://www-i6.informatik.rwth-aachen.de/publications/download/1132/RosendahlJanTranVietAnhKhoaWangWeiyueNeyHermann--AnalysisofPositionalEncodingsforNeuralMachineTranslation--2019.pdf.

Martin Schmitt, Leonardo F. R. Ribeiro, Philipp Dufter, Iryna Gurevych, and Hinrich Schütze. Modeling graph structure via relative position for better text generation from knowledge graphs. *Computing Research Repository*, 2006.09242, 2021. URL https://arxiv.org/abs/2006.09242.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In Marilyn A. Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pp. 464–468. Association for Computational Linguistics, 2018. doi: 10.18653/v1/n18-2074. URL https://doi.org/10.18653/v1/n18-2074.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. Disan: Directional self-attention network for rnn/cnn-free language understanding. In Sheila A. McIlraith and Kilian Q. Weinberger (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 5446–5455. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16126.

Vighnesh Leonardo Shiv and Chris Quirk. Novel positional encodings to enable tree-based transformers. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pp. 12058–12068, 2019. URL http://papers.nips.cc/paper/9376-novel-positional-encodings-to-enable-tree-based-transformers.

Sho Takase and Naoaki Okazaki. Positional encoding to control output sequence length. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3999–4004, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1401. URL https://www.aclweb.org/anthology/N19-1401.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg,

S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf`.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL `https://www.aclweb.org/anthology/W18-5446`.

Benyou Wang, Donghao Zhao, Christina Lioma, Qiuchi Li, Peng Zhang, and Jakob Grue Simonsen. Encoding word order in complex embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=Hke-WTVtwr`.

Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. On position embeddings in BERT. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=onxoVA9FxMw`.

Xing Wang, Zhaopeng Tu, Longyue Wang, and Shuming Shi. Self-attention with structural position representations. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 1403–1409. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1145. URL `https://doi.org/10.18653/v1/D19-1145`.

Yu-An Wang and Yun-Nung Chen. What do position embeddings learn? an empirical study of pre-trained language model positional encoding. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 6840–6849. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.555. URL `https://doi.org/10.18653/v1/2020.emnlp-main.555`.

Chuhan Wu, Fangzhao Wu, and Yongfeng Huang. Da-transformer: Distance-aware transformer. *CoRR*, abs/2010.06925, 2020. URL `https://arxiv.org/abs/2010.06925`.

Hang Yan, Bocao Deng, Xiaonan Li, and Xipeng Qiu. TENER: adapting transformer encoder for named entity recognition. *CoRR*, abs/1911.04474, 2019. URL `http://arxiv.org/abs/1911.04474`.

Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Zhaopeng Tu. Assessing the ability of self-attention networks to learn word order. In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 3635–3644. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1354. URL `https://doi.org/10.18653/v1/p19-1354`.

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. Modeling graph structure in transformer for better AMR-to-text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5459–5468, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1548. URL `https://www.aclweb.org/anthology/D19-1548`.