Survey

Position Information in Transformers: An Overview

Philipp Dufter*[†]
Center for Information and Language Processing, LMU Munich
philipp@cis.lmu.de

Martin Schmitt*[‡]
Center for Information and Language Processing, LMU Munich
martin@cis.lmu.de

Hinrich Schütze
Center for Information and Language Processing, LMU Munich
inquiries@cislmu.org

Transformers are arguably the main workhorse in recent natural language processing research. By definition, a Transformer is invariant with respect to reordering of the input. However, language is inherently sequential and word order is essential to the semantics and syntax of an utterance. In this article, we provide an overview and theoretical comparison of existing methods to incorporate position information into Transformer models. The objectives of this survey are to (1) showcase that position information in Transformer is a vibrant and extensive research area; (2) enable the reader to compare existing methods by providing a unified notation and systematization of different approaches along important model dimensions; (3) indicate what characteristics of an application should be taken into account when selecting a position encoding; and (4) 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

*First two authors contributed equally. [†]Now at Apple. [‡]Now at celebrate company.

Action Editor: Rico Sennrich. Submission received: 27 August 2021; revised version received: 9 March 2022; accepted for publication: 30 March 2022.

https://doi.org/10.1162/COLI_a_00445

rise of pretrained language models (Peters et al. 2018; Howard and 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 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 plain Transformer model is invariant with respect to reordering of the input. However, text data is inherently sequential. Without position information the meaning of a sentence is not well-defined—for example, compare the sequence "the cat chases the dog" to 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 Transformer models. Adding position information can be done by using position embeddings, manipulating attention matrices, or alternative methods such as preprocessing the input with a recurrent neural network. Overall, there is a large variety of methods that add absolute and relative position information to Transformer models. 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.

This survey gives an overview of existing work on incorporating and analyzing position information in Transformer models. Concretely, we provide a theoretical comparison of over 30 Transformer position models and a systematization of different approaches along important model dimensions, such as the number of learnable parameters, and elucidate their differences by means of a unified notation. The goal of this work is not to identify the best way to model position information in Transformer but rather to analyze existing works, and identify common components and blind spots of current research efforts. In summary, we aim to

(1) showcase that position information in Transformer is a vibrant and extensive research area;

(2) enable the reader to compare existing methods by providing a unified notation and systematization of different approaches along important model dimensions;

(3) provide stimuli for future research.

2. Background

2.1 Notation

Throughout this article 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: $(x_i)_{i=1,2\dots,d} = \mathbf{x}$, $(X_{ij})_{i=1,2\dots,t,j=1,2,\dots d} = \mathbf{X}$. Further, the i-th row of $\mathbf{X}$ is the vector $\mathbf{X}_i \in \mathbb{R}^d$. The transpose is denoted as $\mathbf{X}^\intercal$. When we are referring to positions we use $r, s, t, \dots$ whereas we use $i, j, \dots$ 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, Cho, and Bengio (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 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, the **Transformer encoder block**, as shown in Figure 1. Our description and notation follows (Dufter 2021). 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

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

Here, $\mathrm{SoftMax}(\mathbf{A})_{ts} = e^{\mathbf{A}_{ts}} / \sum_{k=1}^{t_{\max}} e^{\mathbf{A}_{tk}}$ is the row-wise softmax function; $\mathrm{LayerNorm}(\mathbf{X})_t = \mathbf{g} \odot (\mathbf{X}_t - \mu(\mathbf{X}_t)/\sigma(\mathbf{X}_t) + \mathbf{b}$ is layer normalization (Ba, Kiros, and Hinton 2016) where $\mu(\mathbf{x}), \sigma(\mathbf{x})$ returns the mean and standard deviation of a vector; and $\mathrm{ReLU}(\mathbf{X}) = \max(0, \mathbf{X})$ is the maximum operator applied component-wise. Note that for addition of a vector to a matrix, we assume broadcasting as implemented in NumPy (Harris et al. 2020). Overall the parameters of a single layer are

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

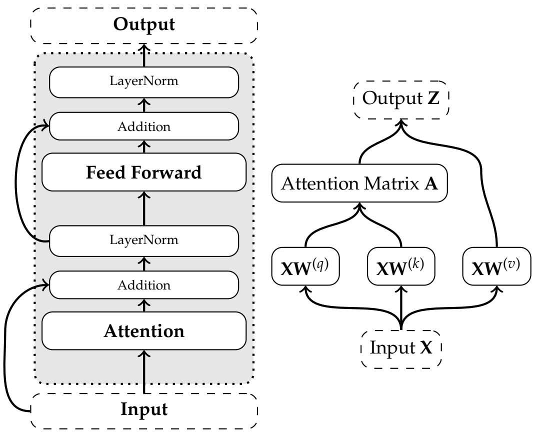


Figure 1
A rough overview of a plain Transformer Encoder Block (gray block) without any position information. The Transformer Encoder Block is usually repeated for l layers. An overview of the attention computation is shown on the right.

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 $\mathbf{M}^{(h)} \in \mathbb{R}^{t_{\max} \times d_h}$ from each attention head are concatenated along their 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, that is, $T(\mathbf{X}) = f_{\theta^l} \circ f_{\theta^{l-1}} \circ \cdots \circ f_{\theta^1}(\mathbf{X})$.

When considering an input $U = (u_1, u_2, \ldots, u_t)$ that consists of t units, such as characters, subwords, or words, the 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 matrix $\mathbf{U}$ is then (among others) used as input to the Transformer model. In the case that t is smaller or larger than $t_{\max}$, the input sequence is padded, that is, filled with special PAD symbols, or truncated.

2.3 Order Invariance

If we take a close look at the Transformer model, we see that it is invariant to reordering 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 obtains $\mathbf{P}_\pi \text{SoftMax}(\mathbf{A}) \mathbf{P}_\pi{}^\intercal \mathbf{P}_\pi \mathbf{X} \mathbf{W}^{(v)} = \mathbf{P}_\pi \mathbf{M}$, 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 set-ups for using 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 block 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 majority of the article. See §4.4 for position encodings that are tailored for encoder–decoder architectures.

3. Recurring Concepts in Position Information Models

Although there are a variety of approaches to integrate position information into Transformers, there are some recurring ideas, which we outline in this section. Based on these concepts we also provide a clustering of the considered position information models (see Table 1).

3.1 Reference Point: 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 sense intuitively, as in sentences like "The cat chased the dog." and "Suddenly, the cat chased

Table 1
Comparison according to several criteria: 📍 = (**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 (notation follows the text paper, d hidden dimension, h # attention heads, n vocabulary size, $t_{\max}$ longest sequence length, l # layers). 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).

Data Structure		**Model**	Ref. Point 📍	Inject. Met. 💉	Learnable 🎓	Recurring ≋	Unbound ∞	**#Param**
	Sequence	Transformer w/ emb. (Vaswani et al. 2017)						$t_{\max}d$
		BERT (Devlin et al. 2019)	A	APE	✓	✗	✗	
		Reformer (Kitaev, Kaiser, and Levskaya 2020)						$(d-d_1)\frac{t_{\max}}{t_1}+d_1t_1$
		FLOATER (Liu et al. 2020)	A	APE	✓	✓	✓	0 or more
		Shortformer (Press, Smith, and Lewis 2021)	A	APE	✗	✓	✓	0
		Wang et al. (2020)	A	–	✓	✗	✓	$2nd$
		Shaw, Uszkoreit, and Vaswani (2018) (abs)	A	MAM	✓	✓	✗	$2t_{\max}^2dl$
		Shaw, Uszkoreit, and Vaswani (2018) (rel)						$2(2t_{\max}-1)dl$
		T5 (Raffel et al. 2020)	R	MAM	✓	✓	✗	$(2t_{\max}-1)h$
		Huang et al. (2020)						$dlh(2t_{\max}-1)$
		DeBERTa (He et al. 2021)	B	Both	✓	✓	✗	$3t_{\max}d$
		Transformer XL (Dai et al. 2019)						$2d+d^2lh$
		TENER (Yan et al. 2019)	R	MAM	✓	✓	✓	$2dlh$
		DA-Transformer (Wu, Wu, and Huang 2021)						$2h$
		TUPE (Ke, He, and Liu 2021)	B	MAM	✓	✗	✗	$2d^2+t_{\max}(d+2)$
		RNN-Transf. (Neishi and Yoshinaga 2019)	R	–	✓	✗	✓	$6d^2+3d$
		SPE (Liutkus et al. 2021)	R	MAM	✓	✓	✗	$3Kdh+ld$
		Transformer w/ sin. (Vaswani et al. 2017)						
		Li et al. (2019)	A	APE	✗	✗	✓	0
		Takase and Okazaki (2019)						
		Oka et al. (2020)						
		Universal Transf. (Dehghani et al. 2019)	A	APE	✗	✓	✓	0
		DiSAN (Shen et al. 2018)	R	MAM	✗	✓	✓	0
		Rotary (Su et al. 2021)						
	Tree	SPR-abs (Wang et al. 2019)	A	APE	✗	✗	✓	0
		SPR-rel (Wang et al. 2019)	R	MAM	✓	✗	✗	$2(2t_{\max}+1)d$
		TPE (Shiv and Quirk 2019)	A	APE	✓	✗	✗	$\frac{d}{D_{\max}}$
	Graph	Struct. Transformer (Zhu et al. 2019)	R	MAM	✓	✓	✓	$5d^2+(d+1)d_r$
		Graph Transformer (Cai and Lam 2020)						$7d^2+3d$
		Graformer (Schmitt et al. 2021)	R	MAM	✓	✓	✗	$2(D_{\max}+1)h$
		Graph Transformer (Dwivedi and Bresson 2020)	A	APE	✗	✗	✓	0
		GRAPH-BERT (Zhang et al. 2020)	B					

the dog." The change in absolute positions due to the added word "Suddenly" causes only a small semantic change, whereas the relative position of "cat" and "dog" is decisive for the meaning of the sentences. The advantage of relative position encoding is that it is invariant with respect to such shifts.

Despite this advantage, it has never been shown conclusively that relative position encoding outperforms an absolute one and thus both systems continue to co-exist—even in the most recent works (see Table 1). Learnable relative position embeddings do have the undeniable disadvantage that they have to consider twice as many different positions (for relative positions to the right and to the left of a word). Hence, in general, they need to train and store more parameters. Table 1 refers to this distinction as **reference point** 📍.

3.2 Injection Method

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 and their sum is fed to the Transformer model: $T(\mathbf{U} + \mathbf{P})$. For the first Transformer layer, this has the following effect:

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

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 effect of adding position embeddings is that it modifies the attention matrix as follows

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

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

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:* attention matrix for an example sentence. *Middle:* learnable absolute position biases. *Right:* position biases with a relative reference point. They are different from absolute encodings as they exhibit an intuitive weight sharing pattern.

As indicated, the matrix **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 APE results in a modification of the attention matrix, APE and MAM are highly interlinked. Still, we make a distinction between these two approaches for multiple reasons: (1) While adding position embedding results, among other effects, in a modified attention matrix, MAM *only* modifies the attention matrix. (2) APE involves learning embeddings for position information whereas MAM is often interpreted as adding or multiplying scalar biases to the attention matrix **A** (see Figure 2). (3) APE is often tied to individual positions and interactions between two positions are computed based on parameters by the model. In contrast, MAM often directly models the interaction of two positions.

Note that methods using relative position encodings exclusively rely on MAM rather than APE. Intuitively, this makes sense because relative position encodings consider pairs of positions and their relation to each other and an attention matrix already models pairs of positions and their interaction. Thus, incorporating positional information into the attention matrix is a straightforward approach. Furthermore, APE relies on the fact that every input unit can be assigned a unique position embedding. As each unit has a different relative position to each other's unit, APE is inherently incompatible with relative position information. Although absolute position information is, in principle, compatible with MAM (see Shaw, Uszkoreit, and Vaswani 2018), there is also a strong correlation between absolute position encodings and APE. Probably, this is because absolute position encodings consider only a single time step at a time, which makes it more intuitive to model position information at the unit level and directly assign position embeddings one unit at a time.

Table 1 distinguishes APE and MAM position information models in its column **injection method** 💉.

3.3 Recurring Integration

In theory, there are many possibilities for integrating position information into a Transformer model, 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 thus has to be propagated to upper layers indirectly. Therefore, the more direct approach of reminding the model of position information in each layer seems more desirable. Then again, this approach also denies the model the flexibility of choosing how strong position information should influence word representations in higher layers.

Often, APE is only added at the beginning, and MAM approaches are used for each layer and attention head. There is, however, no theoretical reason to pair these approaches in that manner. The *recurring* ≋ column in Table 1 marks those approaches where position information is added in each layer anew.

3.4 Fundamental Model Properties

Besides the aforementioned distinctions, we include three other properties per model in Table 1: (1) **Learnable** 🎓 distinguishes whether or not the position information model is learned from data or not. (2) **#Param** provides the number of (trainable) parameters the position information model uses. And (3) **Unbound** ∞ concerns the (theoretical) ability of a position information model to generalize beyond the longest input it has seen during training.

On the one hand, learnable position embeddings give the model more flexibility to adapt the position representations to the task. On the other hand, it also adds parameters, which can lead to overfitting. For trainable position information models, there is a notable trend toward cutting the number of parameters while maintaining good performance; see for example Raffel et al. (2020). Whereas the number of parameters is a fundamental property of any machine learning model, notably the *unbound* ∞ property is specific to models handling position information. Although it is highly desirable for a model to be able to handle input of any length, often bounded values can suffice in practice. Schemes where high length values are clipped, which means that high position values are not distinguished anymore, are also considered bounded even though inputs of any length can be processed by such a model.

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 with units, for example numbered from 0 to t, or assigning relative distances. A **position embedding** then refers to a numerical vector associated with a position encoding.

For the sake of clarity and for easier reading we structure this overview into subsections. To this end, we systematize position information models along two dimensions: **reference point** and **topic**. We chose reference point as it reflects a foundational design choice for each position information model. This dimension can have the values **absolute**, **relative**, or both. Further, we choose the prevalent topic of each paper as second dimension, that is, generic, sinusoidal, graphs, decoder, crosslingual, and analysis. The objective of these categories is not to create a mutually exclusive or exhaustive classification. Given that each paper usually deals with multiple aspects of position information models and sometimes proposes multiple models, this would be challenging if not impossible. Rather, we want to guide the reader to find relevant papers quickly and thus decided on categorizing along topic, similar to keywords. For instance, a reader interested in encoder–decoder models or decoder-only models will find relevant papers in the topic "Decoder." Conversely, papers listed in other topics might be applicable or relevant for decoders as well.

Table 2 shows which papers were assigned to which categories. The following sections deal with each topic and within each topic we discuss approaches with different reference points.

4.1 Generic

In this section we present the first topical cluster, called **Generic**. The papers discussed here exhibit a great variety ranging from learned absolute position embeddings in the original Transformer paper (Vaswani et al. 2017) over complex-valued embeddings (Wang et al. 2020) to adding a recurrent neural network layer before the Transformer (Neishi and Yoshinaga 2019). All these works do not stand out by a particular mathematical characteristic, such as using sinusoidal functions, or a specific theme, such as focusing on encoding graph structures, and therefore do not fit into one of the other categories. Thus, this section contains the most fundamental and original position information models as well as later ones that are equally general for processing sequential

Table 2
Overview and categorization of papers dealing with position information. We categorize along two dimensions: a keyword and topic, which describes the main topic of a paper, and whose reference point is used for the position encodings.

			Reference Point	
		Absolute	**Absolute & Relative**	**Relative**
Topic	**Generic**	Devlin et al. (2019)	Shaw, Uszkoreit, and Vaswani (2018)	Dai et al. (2019)
		Kitaev, Kaiser, and Levskaya (2020)	Ke, He, and Liu (2021)	Raffel et al. (2020)
		Liu et al. (2020)	Dufter, Schmitt, and Schútze (2020)	Chang et al. (2021)
		Press, Smith, and Lewis (2021)	He et al. (2021)	Wu, Wu, and Huang (2021)
		Wang et al. (2020)		Huang et al. (2020)
				Shen et al. (2018)
				Neishi and Yoshinaga (2019)
				Liutkus et al. (2021)
	Sinusoidal	Vaswani et al. (2017)		Yan et al. (2019)
		Dehghani et al. (2019)		Su et al. (2021)
		Li et al. (2019)		
		Likhomanenko et al. (2021)		
	Graphs	Shiv and Quirk (2019)	Wang et al. (2019)	Zhu et al. (2019)
		Dwivedi and Bresson (2020)	Zhang et al. (2020)	Cai and Lam (2020)
				Schmitt et al. (2021)
	Decoder	Takase and Okazaki (2019)		
		Oka et al. (2020)		
		Bao et al. (2019)		
	Crossling.	Artetxe, Ruder, and Yogatama (2020)		
		Ding, Wang, and Tao (2020)		
		Liu et al. (2021a)		
		Liu et al. (2021b)		
	Analysis	Yang et al. (2019)	Rosendahl et al. (2019)	
		Wang and Chen (2020)	Wang et al. (2021)	
			Chen et al. (2021)	

data structures. We first describe papers dealing with absolute position encodings followed by methods that deal with relative ones.

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 $1, 2, \ldots, t_{\max} - 1, t_{\max}$ 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 helps marginally in a convolutional neural network. A Transformer model without any position information, however, performs much worse for some tasks—see for example Wang et al. 2019, Wang et al. 2021.

For very long sequences, that is, large $t_{\max}$, the number of parameters added with $\mathbf{P}$ is significant. Thus, Kitaev, Kaiser, and Levskaya (2020) proposed a more parameter-

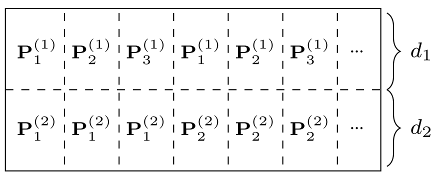

$\mathbf{P}_1^{(1)}$ $\mathbf{P}_2^{(1)}$ $\mathbf{P}_3^{(1)}$ $\mathbf{P}_1^{(1)}$ $\mathbf{P}_2^{(1)}$ $\mathbf{P}_3^{(1)}$ ...
d_1
$\mathbf{P}_1^{(2)}$ $\mathbf{P}_1^{(2)}$ $\mathbf{P}_1^{(2)}$ $\mathbf{P}_2^{(2)}$ $\mathbf{P}_2^{(2)}$ $\mathbf{P}_2^{(2)}$ ...
d_2

Figure 3
Overview of the structure of **P** with axial position embeddings by Kitaev, Kaiser, and Levskaya (2020). They use two position embeddings, 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.

efficient factorization called **axial position embeddings**. Although their method is not described in the paper, a description can be found in their code. 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 **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 $d = d_1 + d_2$ 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. (2020) 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 **P** as a sequence of timesteps $p_1, p_2, \ldots, p_{t_{\max}}$. They suggest modeling 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, for example, 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 (see §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, Smith, and Lewis (2021) 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

$$\tilde{\mathbf{A}} \sim (\mathbf{U}+\mathbf{P})\mathbf{W}^{(q)}\mathbf{W}^{(k)\intercal}(\mathbf{U}+\mathbf{P})^\intercal \qquad (7)$$
$$\tilde{\mathbf{M}} = \mathrm{SoftMax}(\tilde{\mathbf{A}})\mathbf{U}\mathbf{W}^{(v)}$$

The computation of the attention matrix $\bar{\mathbf{A}}$ still depends on absolute position encodings in Shortformer, but $\bar{\mathbf{M}}$ 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, Smith, and Lewis (2021) 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 modeling 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:

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

Then, $\mathbf{r}^{(u)}, \omega^{(u)}, \theta^{(u)} \in \mathbb{R}^d$ 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 $\omega^{(u)}, \theta^{(u)}$. The number of parameters dedicated to the position model therefore does not depend on the number $t_{\max}$ of considered positions but rather on the vocabulary size n. In total, $3nd$ trainable parameters are used compared to nd parameters of a traditional lookup table. We thus mark the number of parameters of the position model alone as $3nd - nd = 2nd$ in Table 1 because this difference is responsible for covering position information. Wang et al. (2020) test their position-sensitive unit embeddings not only on Transformer models, but also on static embeddings, LSTMs, and CNNs, and observe large performance improvements.

4.1.2 Relative Position Encodings. Among the first, Shaw, Uszkoreit, and Vaswani (2018) introduced an alternative method for incorporating both absolute and **relative position encodings**. In their absolute variant they propose changing the computation to

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

where $\mathbf{a}^{(k)}_{(t,s)} \in \mathbb{R}^d$ models the interaction between positions 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. Although 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 **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 understand 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 Equation (4). Ke, He, and Liu (2021) propose **untied position embeddings**. More specifically, they simply put **U** into the Transformer and then modify the attention matrix **A** in the first layer by adding a **position bias**

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

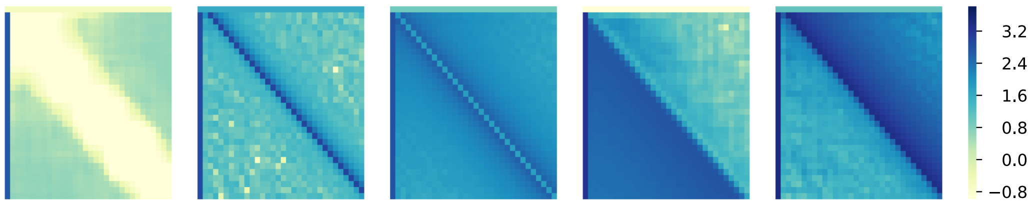


Figure 4
Figure by Ke, He, and Liu (2021). Their position bias is independent of the input and can thus be easily visualized. The absolute position biases learn intuitive patterns as shown above. Patterns (from left to right) include ignoring position information, attending locally, globally, to the left, and to the right. One can clearly see the untied position bias for the first token, which corresponds to the [CLS] token, on the left and top of each matrix.

Compared to Equation (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 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). Ke, He, and Liu (2021) 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, that is, 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, Schmitt, and Schütze (2020), where, in a more limited setting, more specifically in the context of PoS-tagging, learnable **absolute or relative position biases** are learned instead of full position embeddings.

Chang et al. (2021) provide a theoretical link between the position information models proposed by Shaw, Uszkoreit, and Vaswani (2018) and Raffel et al. (2020) and convolutions. They find that combining these two relative position information models increases performance on natural language understanding tasks.

Complementary to that line of research is a method by He et al. (2021): 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, Wu, and Huang (2021) 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 sigmoid function with learnable scalar weights $v^{(m)} \in \mathbb{R}$. More specifically,

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

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 and more elaborate position information models, in particular, relative position encodings by Shaw, Uszkoreit, and Vaswani (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: (1) Similar to (Wu, Wu, and Huang 2021) 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. (2) They consider $\mathbf{R}_{ts} = \mathbf{r}_{s-t}$ as well to distinguish different directions. (3) 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. (4) Last, they extend the method by Shaw, Uszkoreit, and Vaswani (2018) to not only add relative positions to the key, but also to the query in Equation (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 approach is not directly related to relative position encodings, but it can be interpreted as using relative position information. Shen et al. (2018) propose **Directional Self-Attention Networks (DiSAN)**. Besides other differences to plain self-attention, such as multidimensional attention, they notably mask out the upper/lower triangular matrix or the diagonal in **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 by enabling the model to distinguish directions.

Neishi and Yoshinaga (2019) argue that recurrent neural networks (RNN) in the 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, Uszkoreit, and Vaswani (2018) improves performance further, a method they call **RR-Transformer**.

Relative position information models usually require the computation of the full attention matrix **A** because each cell depends on a different kind of relative position interaction. Liutkus et al. (2021) proposed an alternative called **Stochastic Positional Encoding (SPE)**. By approximating relative position interactions as cross-covariance structures of correlated Gaussian processes, they make relative position encodings available to linear-complexity Transformers, such as the Performer (Choromanski et al. 2021), that do not compute the full attention matrix, which would lead to a quadratic complexity. Liutkus et al. (2021) describe two variants of SPE, **sineSPE** that combines K learned sinusoidal components and **convSPE** that learns convolutional filters. Notably, they also propose a gating mechanism that controls with a learnable parameter how much the attention in each vector dimension depends on content vs. position information. The description of SPE in Table 1 is based on gated sineSPE. The experiments

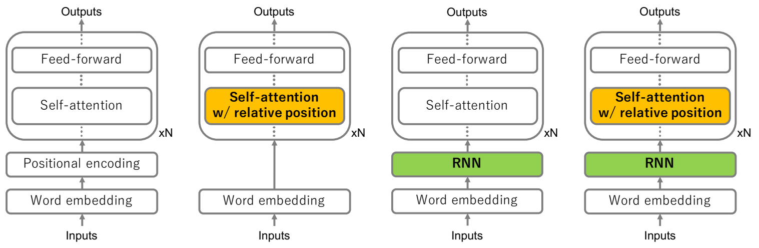


Figure 5
Figure by Neishi and Yoshinaga (2019). Overview of the architecture when using an RNN for learning position information. They combine their RNN-Transformer with relative position embeddings by Shaw, Uszkoreit, and Vaswani (2018) in a model called **RR-Transformer** (far right).

in Liutkus et al. (2021) show that SPE leads to performance improvements compared to absolute position encodings for tasks involving long-range dependencies (Tay et al. 2021b).

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 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. (2020). 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 are 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: (1) 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. (2) $\mathbf{P}_t{}^\intercal \mathbf{P}_{t-r} = \mathbf{P}_t{}^\intercal \mathbf{P}_{t+r}$, which means that sinusoidal position embeddings are unaware of direction. 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

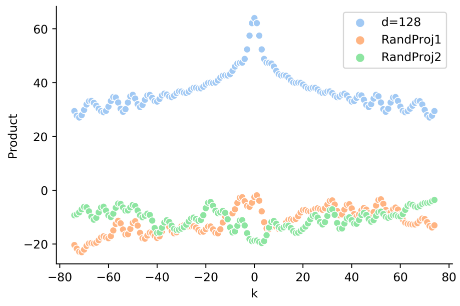


Figure 6
Figure by Yan et al. (2019). Shown is the value of dot product on the y-axis between sinusoidal position embeddings with different relative distance (k) shown on the x-axis. The blue line shows the dot product without projection matrices and the other two lines with random projections. Relative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

maintains these properties and can, in addition, distinguish between directions. 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 $1/\sqrt{d}$ 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 l-th application of the Transformer layer to the input, they add the 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 $k > 1000$.

Likhomanenko et al. (2021) introduce **continuous augmented positional embeddings** and focus on making sinusoidal position embeddings work for other modalities such as vision or speech. More specifically, they propose converting discrete positions to a continuous range and suggest noise augmentations to avoid the model taking up spurious correlations. Instead of using the position t in sinusoidal position embeddings they create t' using mean normalization followed by a series of three random augmentations: (1) global shift $t' = t + \Delta$, (2) local shift $t' = t + \epsilon$, and (3) global scaling $t' = \lambda t$.

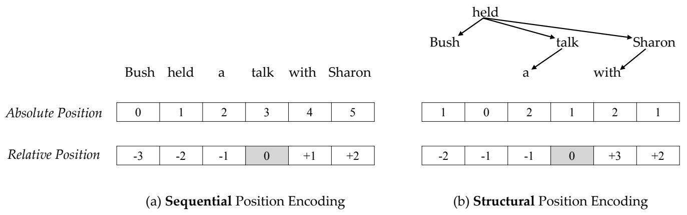


Figure 7
Figure by Wang et al. (2019). They compute absolute and relative encodings not based on the sequential order of a sentence (left), but based on a dependency tree (right). Both absolute and relative encodings can be created.

$\Delta \sim \mathcal{U}(-\Delta_{\max}, \Delta_{\max})$, $\epsilon \sim \mathcal{U}(-\epsilon_{\max}, \epsilon_{\max})$, and $\lambda \sim \mathcal{U}(-\log(\lambda_{\max}), \log(\lambda_{\max}))$ are sampled from a uniform distribution. Note that during inference only mean normalization is performed. As expected, they find their model to work well on vision and speech data. On natural language it performed on par with minor improvements compared with sinusoidal position embeddings as measured on machine translation.

Su et al. (2021) propose multiplying sinusoidal position embeddings rather than adding them in their model **rotary position embeddings**. Intuitively, they rotate unit representations according to their position in a sequence. More specifically, they modify the attention computation to

$$\mathbf{A}_{ts} \sim \mathbf{U}_t{}^{\intercal}\mathbf{W}^{(q)}\mathbf{R}_{\Theta,t-s}\mathbf{W}^{(k)\intercal}\mathbf{U}_s \tag{26}$$

where $\mathbf{R}_{\Theta,t-s} = \mathbf{R}_{\Theta,s}{}^{\intercal}\mathbf{R}_{\Theta,t}$ with $\mathbf{R}_{\Theta,s} \in \mathbb{R}^{d\times d}$ is a block-diagonal matrix with rotation matrices on its diagonal. More specifically, given the parameters $\Theta = (\theta_i)_{i=1,2,\ldots,d/2}$ where $\theta_i = 10000^{-2(i-1)/d}$, the matrix $\mathbf{R}_{\Theta,s}$ has the following rotation matrices on its diagonal:

$$\begin{pmatrix} \cos s\theta_i & -\sin s\theta_i \\ \sin s\theta_i & \cos s\theta_i \end{pmatrix} \tag{27}$$

They find that their method matches the performance of learned absolute position embeddings. Further they find that their approach is beneficial for long sequences.

4.3 Graphs

In the following section, we will take a look at position information models for graphs—more specifically, 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).[1] We can distinguish two settings: (1) 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, that is, the main verb of the sentence, and $d_{\text{tree}}(x, y)$ is the path length between x and y in the tree. (2) 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 in combination with 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, that is, without additional sequential information, leads to a large drop in performance.

Shiv and 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 k-ary tree to some node is defined as the individual decisions that lead to the destination, that is, which of the k children is the next to be visited at each intermediate step. These decisions are encoded as one-hot vectors of size k. 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):

$$\mathbf{A}_{ts} \sim \mathbf{U}_t^{\intercal}\mathbf{W}^{(q)}\left(\mathbf{W}^{(k)\intercal}\mathbf{U}_s + \mathbf{W}^{(r)\intercal}\mathbf{r}_{(t,s)}\right) \tag{28}$$

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

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 ($\uparrow$ and $\downarrow$) 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

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

approaches: (1) A CNN with d kernels of size 4 that convolutes the embedded label sequence $\mathbf{U}^{(r)}$ into $\mathbf{r}$ (see Kalchbrenner, Grefenstette, and Blunsom 2014) and (2) a one-layer self-attention module with sinusoidal position embeddings $\mathbf{P}$ (see § 4.2):

$$\begin{aligned}
\mathbf{A}^{(r)} &\sim (\mathbf{U}^{(r)} + \mathbf{P})\mathbf{W}^{(qr)}\mathbf{W}^{(kr)\intercal}(\mathbf{U}^{(r)} + \mathbf{P})^\intercal \\
\mathbf{M}^{(r)} &= \text{SoftMax}(\mathbf{A}^{(r)})(\mathbf{U}^{(r)} + \mathbf{P})\mathbf{W}^{(vr)} \\
\mathbf{a}^{(r)} &= \text{SoftMax}(\mathbf{W}^{(r_2)} tanh(\mathbf{W}^{(r_1)}\mathbf{M}^{(r)\intercal})) \\
\mathbf{r} &= \sum_{k=1}^{t^{(r)}_{\max}} a^{(r)}_k \mathbf{M}^{(r)}_k
\end{aligned} \tag{29}$$

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. Although it is easily possible to extend this work to disconnected graphs by introducing another special symbol, the effect on performance is unclear.

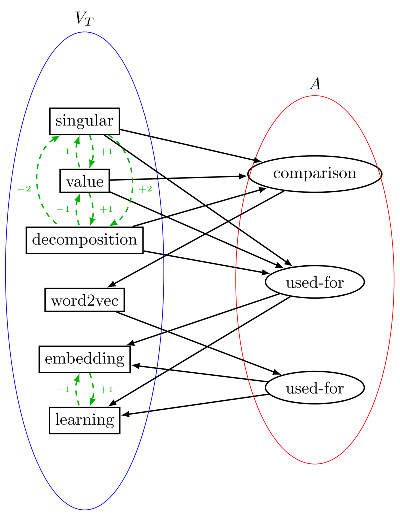
Cai and Lam (2020) also define relative position in a graph based on shortest paths. They differ from the former approach in omitting the edge direction symbols and using a bidirectional GRU (Cho et al. 2014), to aggregate the label information on the paths, similar to the RNN-Transformer described by Neishi and 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

$$\begin{aligned}
\mathbf{A}_{st} &\sim (\mathbf{U}_s + \mathbf{r}_{s\to t})^\intercal \mathbf{W}^{(q)}\mathbf{W}^{(k)\intercal}(\mathbf{U}_t + \mathbf{r}_{t\to s}) \\
&= \underbrace{\mathbf{U}_s{}^\intercal \mathbf{W}^{(q)}\mathbf{W}^{(k)\intercal}\mathbf{U}_t}_{\text{content-based addressing}} + \underbrace{\mathbf{U}_s{}^\intercal \mathbf{W}^{(q)}\mathbf{W}^{(k)\intercal}\mathbf{r}_{t\to s}}_{\text{source relation bias}} \\
&\quad + \underbrace{\mathbf{r}_{s\to t}{}^\intercal \mathbf{W}^{(q)}\mathbf{W}^{(k)\intercal}\mathbf{U}_t}_{\text{target relation bias}} + \underbrace{\mathbf{r}_{s\to t}{}^\intercal \mathbf{W}^{(q)}\mathbf{W}^{(k)\intercal}\mathbf{r}_{t\to s}}_{\text{universal relation bias}}
\end{aligned} \tag{30}$$

It is noteworthy that Cai and Lam (2020) additionally include absolute SPR (see §4.3.1) in their model to exploit the hierarchical structure of the 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):

$$r_{(t,s)} = \begin{cases} \infty, & \text{if there is no path between } t, s \\ \text{sequential relative position of } u_t, u_s & \\ \text{shifted by a constant to avoid clashes,} & \text{if subwords } u_t, u_s \text{ from same word} \\ d_{\text{graph}}(t,s), & \text{if } d_{\text{graph}}(t,s) \le d_{\text{graph}}(s,t) \\ -d_{\text{graph}}(s,t), & \text{if } d_{\text{graph}}(t,s) > d_{\text{graph}}(s,t) \end{cases} \tag{31}$$



	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).

where $0 \leq d_{\text{graph}}(x, y) \leq D_{\max}$ 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 as MAM similar to 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.

All the approaches discussed so far have in common that they allow any node to compute attention over the complete set of nodes in the graph—similar to the global self-attention over tokens in the original Transformer—and that they inject the graph structure solely over a relative position encoding. Dwivedi and Bresson (2020) restrict attention in their graph Transformer to the local node neighborhood and therefore do not need to capture the graph structure by defining the relative position between nodes. Instead they use an absolute APE model by adding Laplacian eigenvectors to the node embeddings before feeding them to the Transformer encoder. Like sinusoidal position embeddings only depend on the (unchanging) order of words, Laplacian eigenvectors only depend on the (unchanging) graph structure. Thus, these position embeddings are parameter-free and can be precomputed for efficient processing. Again, however, an empirical comparison is impossible because Dwivedi and Bresson (2020) evaluate their model on node classification and graph regression whereas the approaches discussed above are tested on graph-to-text generation.

A parameter-free approach is described by Zhang et al. (2020). In their pretraining based on linkless subgraph batching, they combine different features of each node, both predefined (such as node labels) and structural information (such as shortest path lengths), translate them to integers (the position encoding) and, finally, map them to real numbers via sinusoidal position embeddings (see §4.2). The final GRAPH-BERT model takes the overall sum as its input (APE).

4.4 Decoding

Takase and 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 position embeddings** are

$$\mathbf{P}_{tj} = \begin{cases} \sin(10000^{-\frac{j}{d}}(l-t)) \text{ if } j \text{ even} \\ \cos(10000^{-\frac{(j-1)}{d}}(l-t)) \text{ if } j \text{ odd} \end{cases} \tag{32}$$

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

$$\mathbf{P}_{tj} = \begin{cases} \sin(l^{-\frac{j}{d}}t) \text{ if } j \text{ even} \\ \cos(l^{-\frac{(j-1)}{d}}t) \text{ if } j \text{ odd} \end{cases} \tag{33}$$

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, the 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. English uses a subject–verb–object ordering (SVO), but all possible orderings of S, V, and O have been argued to occur in the world's languages. Also, whereas unit ordering is rather fixed in English, it varies considerably in other languages, for example, in German. 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, Ruder, and Yogatama (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

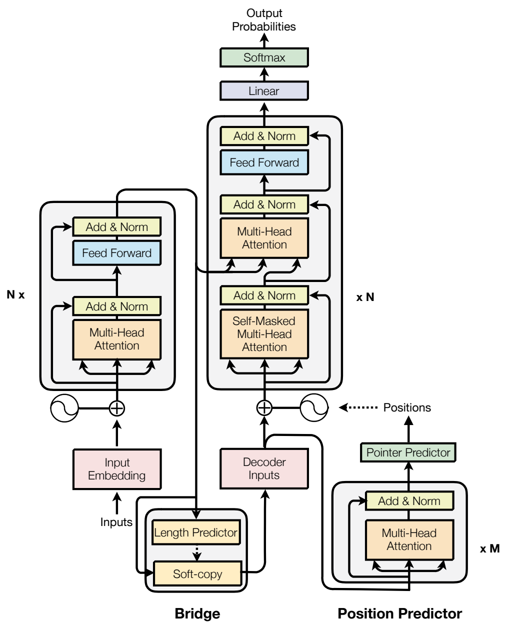


Figure 9
Figure by Bao et al. (2019). Overview of their PNAT architecture with the position prediction module. They use the encoder output to predict the output length and use a modified version as input to the decoder. The position predictor then predicts a permutation of position encodings for the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

harmful. They experimented with learned absolute position embeddings as proposed in Devlin et al. (2019).

Ding, Wang, and Tao (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. (2021a) 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 thereby achieve large improvements in zero-shot translation.

Similarly, Liu et al. (2021b) find that unit order information harms crosslingual transfer, for example in a zero-shot transfer setting. They reduce position information by (1) removing the position embeddings, and replacing them with one-dimensional convolutions, that is, leveraging only local position information, (2) randomly shuffling the unit order in the source language, and (3) 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

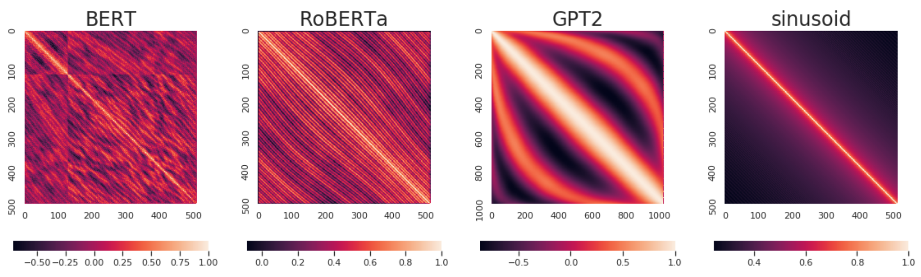


Figure 10
Figure by Wang and Chen (2020). Shown is the position-wise cosine similarity of position embeddings (APE) after pretraining. They compare three pretrained language models that use learned absolute position embeddings as in Devlin et al. (2019), and sinusoidal positions as in Vaswani et al. (2017). BERT shows a cut-off 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.

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 and 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 their 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.

Chen et al. (2021) compare absolute and relative position embeddings as introduced by Ke, He, and Liu (2021). They slightly modify the formulation, add segment embeddings as used in the original BERT formulation (Devlin et al. 2019), and investigate sharing parameters across heads and layers. They find that an argued superiority of relative position embeddings might have been due to the fact that they are added to each attention head. When applying the same procedure with absolute position embeddings they find the best performance across a range of natural language understanding tasks.

We provide a high-level comparison of the discussed methods in Table 1. In this table we group similar approaches from a methodological point of view. The objective is to make comparisons easier and spot commonalities faster.

5. Conclusion

This article presented an overview of methods to inject position information into Transformer models. We hope our unified notation and systematic comparison will foster understanding and spark new ideas in this important research area. In this section, we outline limitations of our survey and possible directions for future research.

5.1 Limitations

While this article aims at providing an exhaustive and systematic overview of position information models that assists researchers in finding relevant work, we would like to point out the following limitations.

(1) There is a range of work proposing modifications to the core Transformer architecture that have an indirect effect on handling position information. In the following, we briefly describe three of these approaches. We decided not to include them in the main discussion because their focus is to modify core components of the Transformer architecture such as the dot-product attention mechanism rather than focusing specifically on how position information is modeled. Note that this survey focuses on position information in Transformers. Including all model architectures that are derived from Transformers that change how position information is handled would quickly go beyond the scope of this survey. You, Sun, and Iyyer (2020) propose hard-coded Gaussian attention. More specifically, they replace the dot-product attention with fixed values based on a Gaussian distribution that is centered around the position t of the query. This can be interpreted as a locality bias where tokens around position t should have most influence. Related to this, Beltagy, Peters, and Cohan (2020) introduce a sliding-window attention that only attends to local context. Within the sliding window, they use the standard dot-product attention. This allows the model to process longer sequences efficiently. As a last example, we describe the Synthesizer (Tay et al. 2021a), which replaces the dot-product attention with random matrices in the extreme case. Here, positional information does not impact the attention matrix at all anymore.

(2) This article does not provide a quantitative comparison of different position information models for two reasons. First, the described models are used in a large number of different tasks and datasets. Picking a single experimental setting as comparison benchmark would bias the reader into thinking that some position information models are universally better than others while this might not be the case. Second, evaluating all models mentioned in this article on a fair and exhaustive set of tasks and datasets is computationally too expensive and constitutes a research effort on its own.

(3) This article presents position information models in a certain structure as outlined in Table 2 and compares them along selected dimensions in Table 1. We do not claim that these categories are exhaustive or mutually exclusive. Similarly, there exist countless alternative categorizations. We outlined the reasons why we decided on this particular presentation at the beginning of §3 and §4.

5.2 Future Work

There are many open questions and starting points for future work. We believe that the following areas are important topics for future work related to position information models.

(1) 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 a wide range of structural information in text that is not considered currently. Some examples include tables, document layout such as headlines, list enumerations, sentence order, and link structures. Can this structure be integrated with current position information models or are new methods required for representing document structure? Is including the document structure useful for downstream tasks such as document-level machine translation?

(2) The majority of the presented position information models are designed with word or subword tokenization in mind. From the beginnings of neural language models (Sutskever, Martens, and Hinton 2011) up to recent research (e.g., Lee, Cho, and Hofmann 2017; Xue et al. 2021), character- and byte-level processing has been a vibrant research area. Designing position information models specifically for character- or byte-level processing thus seems a logical next step. Future directions could make byte-level positions aware of the encoding structure or character-level positions aware of word structures.

(3) Some analysis papers such as Wang et al. (2021) are extensive and provide many insights. Still, many aspects and differences of the position information models are not fully understood. A promising future direction is to continue an empirical comparison of different position information models on more tasks, languages, and datasets.

(4) For many tasks, treating sentences as bag-of-words could be sufficient. Indeed, Wang et al. (2021) show that without position embeddings the performance drops for some tasks are marginal. Thus we consider it interesting to investigate which tasks require explicit position information.

Acknowledgments

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

Artetxe, Mikel, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, pages 4623–4637. https://doi.org/10.18653/v1/2020.acl-main.421

Ba, Lei Jimmy, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*.

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

Beltagy, Iz, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, Curran Associates, Inc.

Cai, Deng and Wai Lam. 2020. Graph transformer for graph-to-sequence learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(5):7464–7471. `https://doi.org/10.1609/aaai.v34i05.6243`

Chang, Tyler, Yifan Xu, Weijian Xu, and Zhuowen Tu. 2021. Convolutions and self-attention: Re-interpreting relative positions in pre-trained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4322–4333. `https://doi.org/10.18653/v1/2021.acl-long.333`

Chen, Pu-Chin, Henry Tsai, Srinadh Bhojanapalli, Hyung Won Chung, Yin-Wen Chang, and Chun-Sung Ferng. 2021. Demystifying the better performance of position encoding variants for Transformer. *CoRR*, abs/2104.08698.

Cho, Kyunghyun, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. `https://doi.org/10.3115/v1/D14-1179`

Choromanski, Krzysztof Marcin, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. 2021. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021*, OpenReview.net.

Conneau, Alexis, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, pages 8440–8451. `https://doi.org/10.18653/v1/2020.acl-main.747`

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

Dehghani, Mostafa, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *7th International Conference on Learning Representations, ICLR 2019*, OpenReview.net.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186.

Ding, Liang, Longyue Wang, and Dacheng Tao. 2020. Self-attention with cross-lingual position representation. In *Proceedings of the 58th Annual Meeting of the Association for*

Computational Linguistics, ACL 2020, pages 1679–1685. https://doi.org/10.18653/v1/2020.acl-main.153

Dufter, Philipp. 2021. *Distributed Representations for Multilingual Language Processing*. Ph.D. thesis, Ludwig-Maximilians-Universität München.

Dufter, Philipp, Martin Schmitt, and Hinrich Schütze. 2020. 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*, pages 3630–3636. https://doi.org/10.18653/v1/2020.coling-main.324

Dwivedi, Vijay Prakash and Xavier Bresson. 2020. A generalization of transformer networks to graphs. *CoRR*, abs/2012.09699.

Gehring, Jonas, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, pages 1243–1252.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature*, 585(7825):357–362. https://doi.org/10.1038/s41586-020-2649-2

He, Pengcheng, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net.

Howard, Jeremy and Sebastian Ruder. 2018. 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)*, pages 328–339. https://doi.org/10.18653/v1/P18-1031

Huang, Zhiheng, Davis Liang, Peng Xu, and Bing Xiang. 2020. Improve transformer models with better relative position embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020*, pages 3327–3335. https://doi.org/10.18653/v1/2020.findings-emnlp.298

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

Ke, Guolin, Di He, and Tie-Yan Liu. 2021. Rethinking positional encoding in language pre-training. In *9th International Conference on Learning Representations, ICLR 2021*, OpenReview.net.

Kitaev, Nikita, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020*, OpenReview.net.

Lee, Jason, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics.*, 5:365–378. https://doi.org/10.1162/tacl_a_00067

Li, Hailiang, Adele Y. C. Wang, Yang Liu, Du Tang, Zhibin Lei, and Wenye Li. 2019. An augmented transformer architecture for natural language generation tasks. In *2019 International Conference on Data Mining Workshops, ICDM Workshops 2019*, pages 1–7. https://doi.org/10.1109/ICDMW48858.2019.9024754

Likhomanenko, Tatiana, Qiantong Xu, Ronan Collobert, Gabriel Synnaeve, and Alex Rogozhnikov. 2021. CAPE: Encoding relative positions with continuous augmented positional embeddings. *CoRR*, abs/2106.03143.

Liu, Danni, Jan Niehues, James Cross, Francisco Guzmán, and Xian Li. 2021a. Improving zero-shot translation by disentangling positional information. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers)*, pages 1259–1273. https://doi.org/10.18653/v1/2021.acl-long.101

Liu, Xuanqing, Hsiang-Fu Yu, Inderjit S. Dhillon, and Cho-Jui Hsieh. 2020. Learning to encode position for Transformer with continuous dynamical model. In *Proceedings of the 37th International*

Conference on Machine Learning, ICML 2020, pages 6327–6335.

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

Liu, Zihan, Genta Indra Winata, Samuel Cahyawijaya, Andrea Madotto, Zhaojiang Lin, and Pascale Fung. 2021b. On the importance of word order information in cross-lingual sequence labeling. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021*, pages 13461–13469.

Liutkus, Antoine, Ondřej Cífka, Shih-Lun Wu, Umut Şimşekli, Yi-Hsuan Yang, and Gaël Richard. 2021. Relative positional encoding for Transformers with linear complexity. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7067–7079.

Neishi, Masato and Naoki Yoshinaga. 2019. On the relation between position information and sentence length in neural machine translation. In *Proceedings of the 23rd Conference on Computational Natural Language Learning, CoNLL 2019*, pages 328–338. `https://doi.org/10.18653/v1/K19-1031`

Oka, Yui, Katsuki Chousa, Katsuhito Sudoh, and Satoshi Nakamura. 2020. Incorporating noisy length constraints into Transformer with length-aware positional encodings. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3580–3585. `https://doi.org/10.18653/v1/2020.coling-main.319`

Peters, Matthew, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. 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)*, pages 2227–2237. `https://doi.org/10.18653/v1/N18-1202`

Press, Ofir, Noah A. Smith, and Mike Lewis. 2021. Shortformer: Better language modeling using shorter inputs. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers)*, 5493–5505. `https://doi.org/10.18653/v1/2021.acl-long.427`

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

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:140:1–140:67.

Rosendahl, Jan, Viet Anh Khoa Tran, Weiyue Wang, and Hermann Ney. 2019. Analysis of positional encodings for neural machine translation. In *Proceedings of the 16th International Conference on Spoken Language Translation, IWSLT 2019*, 6 pages.

Schmitt, Martin, Leonardo F. R. Ribeiro, Philipp Dufter, Iryna Gurevych, and Hinrich Schütze. 2021. Modeling graph structure via relative position for text generation from knowledge graphs. In *Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15)*, pages 10–21. `https://doi.org/10.18653/v1/2021.textgraphs-1.2`

Shaw, Peter, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, Volume 2 (Short Papers)*, pages 464–468. `https://doi.org/10.18653/v1/N18-2074`

Shen, Tao, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. DiSAN: Directional self-attention network for RNN/CNN-free language understanding. In *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)*, pages 5446–5455.

Shiv, Vighnesh Leonardo and Chris Quirk. 2019. Novel positional encodings to enable tree-based transformers. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information*

Processing Systems 2019, NeurIPS 2019, pages 12058–12068.

Su, Jianlin, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2021. RoFormer: Enhanced transformer with rotary position embedding. *CoRR*, abs/2104.09864.

Sutskever, Ilya, James Martens, and Geoffrey E. Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 1017–1024.

Takase, Sho and Naoaki Okazaki. 2019. 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)*, pages 3999–4004. https://doi.org/10.18653/v1/N19-1401

Tay, Yi, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. 2021a. Synthesizer: Rethinking self-attention for transformer models. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, pages 10183–10192.

Tay, Yi, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2021b. Long range arena: A benchmark for efficient transformers. In *9th International Conference on Learning Representations, ICLR 2021*, OpenReview.net.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., pages 5998–6008.

Wang, Alex, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. 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*, pages 353–355. https://doi.org/10.18653/v1/W18-5446

Wang, Benyou, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. 2021. On position embeddings in BERT. In *9th International Conference on Learning Representations, ICLR 2021*, OpenReview.net.

Wang, Benyou, Donghao Zhao, Christina Lioma, Qiuchi Li, Peng Zhang, and Jakob Grue Simonsen. 2020. Encoding word order in complex embeddings. In *8th International Conference on Learning Representations, ICLR 2020*, OpenReview.net.

Wang, Xing, Zhaopeng Tu, Longyue Wang, and Shuming Shi. 2019. Self-attention with structural position representations. 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 2019*, pages 1403–1409. https://doi.org/10.18653/v1/D19-1145

Wang, Yu-An and Yun-Nung Chen. 2020. What do position embeddings learn? An empirical study of pre-trained language model positional encoding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pages 6840–6849. https://doi.org/10.18653/v1/2020.emnlp-main.555

Wu, Chuhan, Fangzhao Wu, and Yongfeng Huang. 2021. Da-Transformer: Distance-aware Transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*, pages 2059–2068. https://doi.org/10.18653/v1/2021.naacl-main.166

Xue, Linting, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2021. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *CoRR*, abs/2105.13626. https://doi.org/10.1162/tacl_a_00461

Yan, Hang, Bocao Deng, Xiaonan Li, and Xipeng Qiu. 2019. TENER: Adapting Transformer Encoder for named entity recognition. *CoRR*, abs/1911.04474.

Yang, Baosong, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Zhaopeng Tu. 2019. Assessing the ability of self-attention networks to learn word order. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Volume 1: Long Papers*, pages 3635–3644. https://doi.org/10.18653/v1/P19-1354

You, Weiqiu, Simeng Sun, and Mohit Iyyer. 2020. Hard-coded Gaussian attention for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association* *for Computational Linguistics*, pages 7689–7700. https://doi.org/10.18653/v1/2020.acl-main.687

Zhang, Jiawei, Haopeng Zhang, Congying Xia, and Li Sun. 2020. Graph-BERT: Only attention is needed for learning graph representations. *CoRR*, abs/2001.05140.

Zhu, Jie, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. 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)*, pages 5459–5468. https://doi.org/10.18653/v1/D19-1548