TACO: Pre-training of Deep Transformers with Attention Convolution using Disentangled Positional Representation

Anonymous ACL submission

Abstract

Word order, as a crucial part to understand natural language, has been carefully considered in pre-trained models by incorporating different kinds of positional encodings. However, existing pre-trained models mostly lack the ability to maintain robustness against minor permutation of words in learned representations. We therefore propose a novel architecture named **T**ransformer with **A**ttention **CO**nvolution (**TACO**), to explicitly disentangle positional representations and incorporate convolution over multi-source attention maps before softmax in self-attention. Additionally, we design a novel self-supervised task, masked position modeling (MPM), to assist our TACO model in capturing complex patterns with regard to word order. Combining MLM (masked language modeling) and MPM objectives, the proposed TACO model can efficiently learn two disentangled vectors for each token, representing its content and position respectively. Experimental results show that TACO significantly outperforms BERT in various downstream tasks with fewer model parameters. Remarkably, TACO achieves +2.6% improvement over BERT on SQuAD 1.1 task, +5.4% on SQuAD 2.0 and +3.4% on RACE, with only 46K pre-training steps.

1 Introduction

In recent years, pre-training/fine-tuning has become a popular paradigm in a wide range of natural language processing applications including text classification (Dai and Le, 2015; Howard and Ruder, 2018), sentiment analysis (Ke et al., 2020b; Peters et al., 2018), summarization (Liu and Lapata, 2019; Zhang et al., 2020) and text generation (Radford et al., 2019; Bao et al., 2020b; Keskar et al., 2019). Among various applications under such a paradigm, BERT (Devlin et al., 2019) is the most popular pre-trained model with significant better performance than previous benchmarks in 11 NLP tasks. Since then, many kinds of pre-trained models, which use Transformer block (Vaswani et al., 2017) or its variants (Lin et al., 2021) as backbone, have been proposed and researches in this field are still under active exploration for better performance, efficiency and interpretation (Yang et al., 2019; Joshi et al., 2020; Zhang et al., 2019; Bao et al., 2020a; Qiu et al., 2020).

Normally, the major challenge of these pre-trained models is how to fully comprehend a sentence, including the concept of each word and the grammatical structure of the sentence. Specifically, the concept underlying a word can be represented by its semantic meaning, and the grammatical structure is actually expressed as a "natural" word order. In some recent reports, the importance of "word order" has been emphasized (Pham et al., 2020), since it is crucial to generate and understand a natural sentence. Comparing these three sentences as below, we can explicitly elaborate the importance and flexibility of word order during composing a reasonable sentence.

1. *Bob once asked Alice to borrow a book.*
2. *Once, Bob asked Alice to borrow a book.*
3. *Bob once borrow Alice to asked a book.*

Obviously, the first two sentences express the same meaning, which state the event about borrowing. The third sentence is in fact illegal since two critical verbs ("asked" and "borrow") are exchanged. Although we can sometimes infer actual meaning from a wrong sentence, it may still impede our understanding of natural language if the incorrectness is critical.

Generally, there are two kinds of approaches to incorporate word order into current pre-trained models. One is absolute position embeddings as employed by Transformer (Vaswani et al., 2017) and BERT (Devlin et al., 2019). Every word in a given sentence corresponds to a fixed position encoded by an embedding vector. The posi-

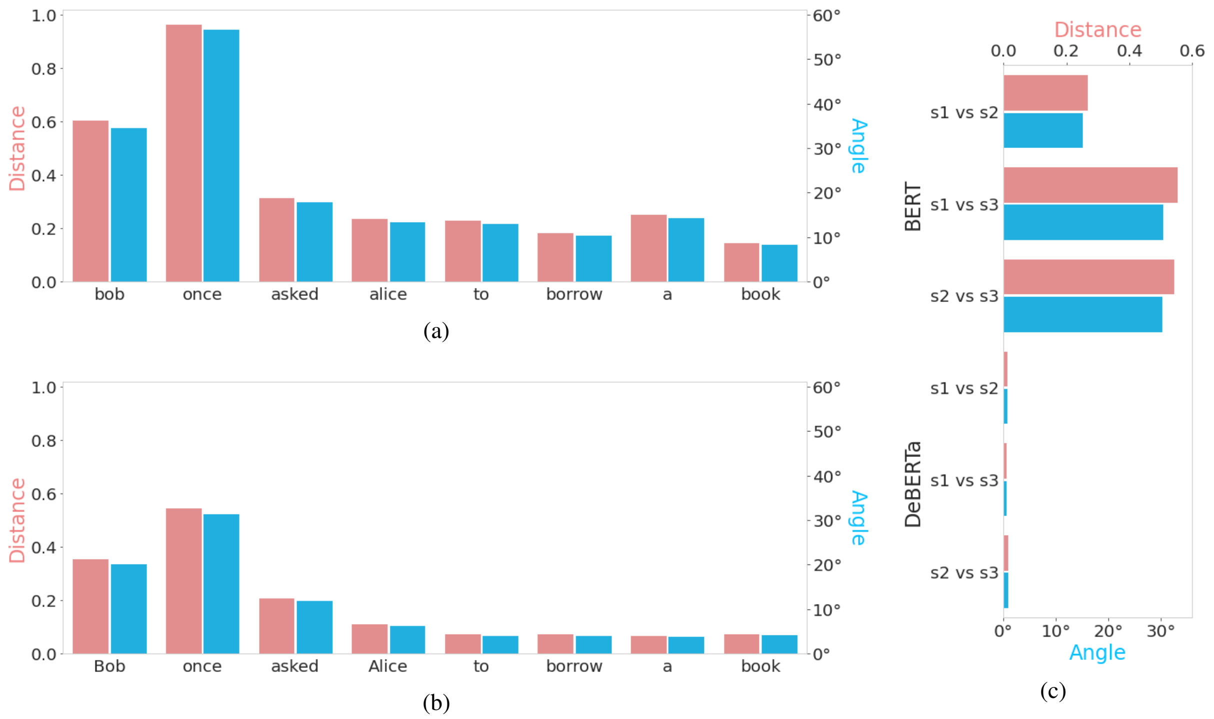

Figure 1: Differences between output representations of same words (tokens) from different sentences. (a). Differences in vector representation output by BERT-base between sentence 1 and 2 in terms of token. (b). Representation differences output by DeBERTa-base between sentence 1 and 2. (c). Differences between any two sentences via [CLS] representations output by BERT-base or DeBERTa-base. Two measures are computed to evaluate the difference between output hidden vectors of same words from two sentences. The first measure "Distance" is the Euclidean distance from one vector to another normalized by the minimum of vector lengths, which can be formulated as $d(\boldsymbol{v}_1, \boldsymbol{v}_2) = \frac{\|\boldsymbol{v}_1 - \boldsymbol{v}_2\|}{\min(\|\boldsymbol{v}_1\|, \|\boldsymbol{v}_2\|)}$. The second measure "Angle" is the angle between two vectors computed through cosine similarity as the formula $\theta(\boldsymbol{v}_1, \boldsymbol{v}_2) = \arccos \frac{\boldsymbol{v}_1 \cdot \boldsymbol{v}_2}{\|\boldsymbol{v}_1\| \|\boldsymbol{v}_2\|}$. For a given word (or token), $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ in both equations are the representation vectors of last layer produced by BERT or DeBERTa taking sentence 1 or 2 as input respectively.

tion embedding of a word is usually added to its word embedding and the summed embeddings of all words are then fed into a Transformer-based model. If a word moves forward or backward a little, the resulting embeddings may be much different and therefore change the final representations of a given sentence (Figure 1a). The other kind of approaches is relative positional encodings adopted by Transformer-XL (Dai et al., 2019) and DeBERTa (He et al., 2020). The relative position (distance) from a query to a key can be encoded by a scalar or vector and integrated into attention score between query and key contents through some transformations. Different from absolute encodings, there are no additional position embeddings added to word embeddings as initial input and relative positional embeddings are always shared across all layers. Since a relative position represents the distance of sequence order between two words, models of this kind are more sensitive to the changes of relative word orders rather than translation of all words. As seen in Figure 1b, the vector representations of "Bob" and "once" output by DeBERTa from sentence 2 greatly differ from those from sentence 1. All other words contrarily exhibit some characteristics of translation invariance which agrees with the essence of relative positions. Besides, BERT is able to roughly identify the difference between sentence 1 (or 2) and sentence 3 from the representation of [CLS] token, while DeBERTa apparently fails to distinguish three sentences, which is probably caused by the lack of an additional task (e.g., next sentence prediction) in pre-training (Figure 1c). Although word order is explicitly encoded by absolute or relative positional embeddings, pre-trained models of both kinds exhibit more or less fragility against minor permutations of an input sentence in sequence order.

To alleviate the intrinsic fragility in many pre-trained models, we propose several simple, yet effective modifications to current models, which leads to a new architecture called **T**ransformer with **A**ttention **CO**nvolution (**TACO**) for language modeling. The main improvements of TACO includes: 1). disentangled positional representation of each token throughout all layers of the model; 2). a novel Transformer block consisting of pseudo-Siamese structure consisting of feed-forward networks and layer normalizations, and a joint self-attention sub-layer; 3). 2-dimensional convolution over multi-source attention maps in self-attention; 4). an additional self-supervised task to predict masked positions in pre-training. Our model is illustrated in Figure 2.

To validate the effectiveness of TACO, we therefore conduct a series of experiments and achieve comparable results on many downstream tasks. The TACO-base with fewer parameters (100M parameters) and less pre-training (only 46K) achieves 84.5 in GLUE benchmark, compared to 83.1 of BERT. For several question-answering tasks, the proposed TACO-base model also significantly outperforms BERT-base by +2.6% on SQuAD 1.1, +5.4% on SQuAD 2.0, and +3.4% on RACE. We further perform ablation studies to evaluate the effectiveness of different modifications.

2 Related Works

To this day, many works concerning positional encodings or position-involved attentions in Transformer block have been accomplished which emphasize the contribution of positions in resulting attention scores (Lin et al., 2021). Both Transformer (Vaswani et al., 2017) and BERT (Devlin et al., 2019) use absolute embeddings to represent different positions of input tokens. Different from absolute embeddings, Shaw et al. (2018) firstly introduce relative position representations to Transformer encoder. T5 (Raffel et al., 2020) then uses a scalar instead of a vector as relative position bias to encode the order of every word in a sentence. Transformer-XL, on the other hand, extends the work of Shaw et al. (2018) by integrating a global content bias and a global positional bias into attention score between every query and key vector (Dai et al., 2019). Following the idea of decomposed attention scores, DeBERTa (He et al., 2020) disentangles content-to-position and position-to-content terms from original attention score. Contrary to DeBERTa, TUPE (Ke et al., 2020a) adds a position-to-position term and a relative position bias instead of interactive attentions between token content and position. After a thorough analysis of these models, it is found that absolute or relative positional encodings basically act as supplementary parts in attention scores, and hence sequence order of words has not been paid enough attention to in pre-training language models.

As a novel trick in NLP, convolution is recently utilized to improve performance or efficiency of Transformer-based models. ConvBERT (Jiang et al., 2020) employs dynamic convolution conditioning on queries to capture local dependency among long sequences, and thus significantly decreases total size of the model and computational cost. Another application of convolution would be EA-Transformer (Wang et al., 2021), which adds a skip connection between adjacent self-attentions and uses convolution over attention maps of all heads to produce final attention weights. Chang et al. (2021) re-interpret relative embeddings as lightweight convolutions and incorporate them as composite attention. In summary, these modified models mostly benefit from the incorporation of convolutions and achieve better performance on downstream tasks.

3 Methodology

In this section, we introduce main architecture of TACO model and its detailed implementation.

3.1 Model Architecture

For a single sentence or a pair of sentences, we follow the procedure as BERT (Devlin et al., 2019) to tokenize them into tokens and pack them together. When the input sequence has been constructed, TACO firstly retrieves word embedding $\mathbf{x}_i$ and positional embedding $\mathbf{p}_i$ for each token i and then concatenates them together as its complete representation $\mathbf{e}_i = \text{concat}(\mathbf{x}_i, \mathbf{p}_i)$. Next, the initial representations of all tokens are fed to TACO model for further process.

As seen in Figure 2 left, we employ a multi-layer TACO encoder to encode contextual information of input representation. The main difference of each TACO layer from traditional Transformer lies in the pseudo-Siamese structure consisting of feed-forward networks and layer normalizations, as well as a joint self-attention sub-layer with several significant modifications. To resolve the ambiguity of Siamese components, we use **WE** and **PE** as

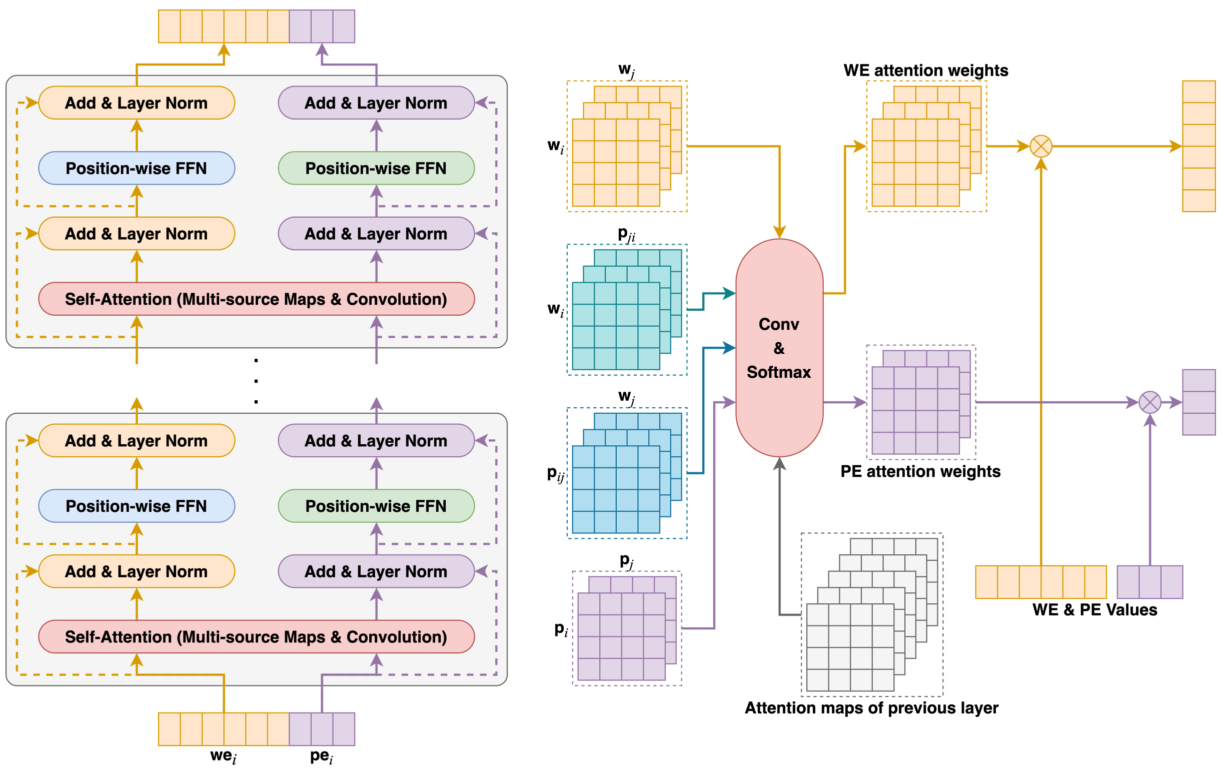

Figure 2: Basic architecture of TACO model. The left presents the main structure of TACO mostly like Transformer encoder. The primary difference is that TACO uses two feed-forward networks, separated residual connection and layer normalization in each layer for token content and positional vectors respectively. The self-attention of each layer is unified, in order to produce attention distributions using complete token and positional information. The right depicts the workflow of how TACO computes attention states for token content and position through attention convolution over multi-source attention maps.

identifiers to distinguish distinct vector representations, sub-layer components, model parameters and hyper-parameters with respect to token content and position. Generally, the pseudo-Siamese structure indicates that WE and PE input vectors are separately transformed through distinct FFN sub-layers and layer normalizations in each layer and they do not share internal parameters.

Here, we briefly describe the workflow of our model. After taking in the initial token vectors, TACO firstly computes multiple attention maps using WE or PE representations for queries and keys respectively. Then a unified convolution is performed over all attention maps stacked together in order to generate different attention weights for WE and PE heads (Figure 2 right). Subsequently, TACO aggregates and projects hidden states using vector representations and attention weights of all heads for WE and PE separately. Passing through corresponding residual connections and layer normalizations, WE and PE hidden vectors are then fed to different feed-forward sub-layers, followed by similar post-processing operations to those after self-attention. At last, TACO concatenates WE and PE output vector of each token as its complete representation, and feeds them to next layer. The complete representations of last layer are exploited to fine-tune downstream tasks through task-specific output layer.

3.2 Disentangled Positional Representation

We use a multi-layer TACO encoder to transform token and positional representations of an input sentence. Given the input sentence, the token vectors are denoted as $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ and the position vectors $\mathbf{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_N\}$. The hidden size of WE is denoted as d_{WE} and PE hidden size d_{PE}. Then an L-layer TACO model encodes the input as:

$$\begin{aligned}(\mathbf{H}^l_{\text{WE}}, \mathbf{H}^l_{\text{PE}}) &= \text{TACO}^l(\mathbf{H}^{l-1}) \\ \mathbf{H}^l &= \text{concat}(\mathbf{H}^l_{\text{WE}}, \mathbf{H}^l_{\text{PE}}) \end{aligned} \quad (1)$$

where $l \in [1, L]$, $\mathbf{H}^0 = \text{concat}(\mathbf{X}, \mathbf{P})$ and $\mathbf{H}^L = [\mathbf{h}^L_1, \dots, \mathbf{h}^L_N]$. Obviously, the contextualized representation $\mathbf{h}^L_i \in \mathbb{R}^{d_{\text{WE}}+d_{\text{PE}}}$ is achieved by concatenating WE and PE hidden vector of token i from last layer.

As shown in Figure 2 right, in order to produce WE and PE representations separately with the unified self-attention, we split multiple heads into two sets, h_{WE} WE heads and h_{PE} PE heads, to compute token and positional attention weights respectively. The computational process of self-attention is essentially analogous to Transformer except two major differences. One of them is that we use a similar process to compute two outputs from multi-head self-attention with respect to WE and PE. The other is that the attention weights are computed by more complex method than Transformer. We formulate the process as:

$$\begin{aligned}
\mathbf{H}'_{\mathrm{WE}} &= \mathrm{concat}(\mathbf{head}^{1}_{\mathrm{WE}}, \dots, \mathbf{head}^{h_{\mathrm{WE}}}_{\mathrm{WE}})\mathbf{W}^{O}_{\mathrm{WE}} \\
\mathbf{head}^{i}_{\mathrm{WE}} &= f(\mathbf{H}; i_{\mathrm{WE}})\mathbf{H}_{\mathrm{WE}}\mathbf{W}^{V_i}_{\mathrm{WE}} \\
\mathbf{H}'_{\mathrm{PE}} &= \mathrm{concat}(\mathbf{head}^{1}_{\mathrm{PE}}, \dots, \mathbf{head}^{h_{\mathrm{PE}}}_{\mathrm{PE}})\mathbf{W}^{O}_{\mathrm{PE}} \\
\mathbf{head}^{j}_{\mathrm{PE}} &= f(\mathbf{H}; j_{\mathrm{PE}})\mathbf{H}_{\mathrm{PE}}\mathbf{W}^{V_j}_{\mathrm{PE}} \qquad (2)
\end{aligned}$$

where $\mathbf{H}_{\mathrm{WE}}$ is the input matrix composed of WE hidden vectors of all tokens from previous layer as equation (1). $\mathbf{W}^{V_i}_{\mathrm{WE}} \in \mathbb{R}^{d_{\mathrm{WE}} \times d_v}$ is the projection matrix for the i-th head to project WE vectors into low-dimensional space. $\mathbf{W}^{O}_{\mathrm{WE}} \in \mathbb{R}^{h_{\mathrm{WE}} d_v \times d_{\mathrm{WE}}}$ is the output projection to fuse hidden states of all WE heads. $\mathbf{H}_{\mathrm{PE}}$, $\mathbf{W}^{V_j}_{\mathrm{PE}} \in \mathbb{R}^{d_{\mathrm{PE}} \times d_v}$ and $\mathbf{W}^{O}_{\mathrm{PE}} \in \mathbb{R}^{h_{\mathrm{PE}} d_v \times d_{\mathrm{PE}}}$ are corresponding matrices related to PE. $f(\cdot)$ is a unified function to generate attention weights for each WE and PE head. $\mathbf{H}'_{\mathrm{WE}}$ and $\mathbf{H}'_{\mathrm{PE}}$ are output vectors of self-attention, which will be fed to FFN sub-layers after residual connections and layer normalizations.

3.3 Multi-source Attention Maps

As summarized by Lin et al. (2021), the prerequisite of attention weights is to calculate raw score of any query token q_i attending to any key k_j. The raw scores of all pairs of queries and keys form an attention map $\mathbf{A}$, each element $\mathbf{A}^{i,j}$ of which indicates how much attention q_i pays to k_j. We individually use query information from token content and position together with different key information to generate diverse attention scores. We mainly categorize resulting attention maps into three groups in terms of information sources of queries and keys. Content attention and positional attention employ query and key content or their positional information to compute attention scores, respectively. As for interactive attention, which includes content-to-position and position-to-content terms, we use relative position-based attentions as depicted in DeBERTa (Refer to Appendix A.1 for more details).

Despite of three groups of attention maps described above, we also create skip edges to connect multi-head attention modules in adjacent layers like RealFormer (He et al., 2021) and EA-Transformer (Wang et al., 2021). Instead of directly adding to the attention maps of current layer, we consider the attention logit matrices from previous layer as an additional source for further fusion. So far, we achieve $4h_{\mathrm{WE}} + 2h_{\mathrm{PE}}$ attention maps in total for each layer (except the first layer without residual attentions), including h_{WE} content maps, h_{PE} positional maps, $2h_{\mathrm{WE}}$ interactive maps and $h_{\mathrm{WE}} + h_{\mathrm{PE}}$ residual maps of previous layer.

3.4 Fusion of Attention Maps

Once the attention maps composed of raw scores have been computed, typical pre-trained models transform them into attention weights by applying softmax over the last dimension. However, our model slightly modifies the conventional process by fusing multi-source attention maps using convolution before softmax. Analog to images in computer vision, 2-dimensional convolution is a promising choice if we regard stacked attention maps as multi-channel inputs and attention weights of all heads as output feature maps. As a matter of fact, we adopt convolution followed by PReLU activation (He et al., 2015) to fuse multi-source maps for two reasons. One obvious reason is that the number of attention maps differs from pre-defined number of attention heads. It may cause redundant or insufficient usage of certain maps if we insist on matching them with attention weights of all heads. The other is that TACO essentially requires different attention weights for WE and PE heads in order to model inherent dependencies from the perspectives of semantic relation and word order.

For simplicity, we additionally specify kernel size and stride to be 1 in directions of both width and height in our model. At this point, the function $f(\cdot)$ in equation (2) indeed stands for the generation of multi-source attention maps, the 2d convolution with PReLU activation, and softmax.

3.5 Position Prediction Task

We here introduce two self-supervised tasks for pre-training TACO model. We firstly use masked language modeling (MLM) task to pre-train the model as introduced in BERT (Devlin et al., 2019), which has limited effect to drive the model to capture underlying patterns concerning word order of a sentence. Therefore, we propose an additional task similar to MLM for masked position prediction, as a complementary to MLM objective.

Token Masking First of all, we adopt the span-based masking scheme in SpanBERT (Joshi et al., 2020) to generate an input sequence by masking continuous segments of text instead of individual tokens. In practice, we set the probability of geometric distribution to 0.2 for span length sampling, and also restrict sampled spans to a maximum length of 3. The strategy to mask sampled spans is similar to BERT (Devlin et al., 2019), which is also clearly described in SpanBERT.

Position Masking Analog to token masking, we come up with a similar procedure to corrupt a position sequence for better modeling word order. For a given sentence X, the position of every token is denoted as $P = (p_1, p_2, \ldots, p_N)$. We create a similar random generator as token masking to sample spans of positions using a geometric distribution with $p = 0.2$ and clip the span length at 6. As a consequence, we select 15% of the positions in total through the sampling generator. The masking strategy for position spans is slightly different from token masking. Refer to Appendix A.2 for detailed procedure.

As described above, we independently employ these two masking schemes to generate training data, resulting in a pair of examples for one sentence. For a given sentence X and its position sequence P, we generate $\{\tilde{X}, P\}$ by following the procedure of token masking and $\{X, \tilde{P}\}$ by position masking, where $\tilde{X}$ and $\tilde{P}$ are denoted as corrupted sequences of tokens and positions after masking respectively.

Using masked tokens and original positions $\{\tilde{X}, P\}$, our model can try to reconstruct the original tokens through bi-directional contexts. This is the typical pre-training task known as MLM originally introduced in BERT (Devlin et al., 2019). Denote $\mathcal{K}$ as the set of indices of masked tokens. Then MLM pre-trains the model θ by maximizing the following objective

$$\mathcal{L}_{\text{MLM}} = -\sum_{i \in \mathcal{K}} \log Pr(x_i | \{\tilde{X}, P\}; \theta) \quad (3)$$

On the hand, TACO is also trained to recover the right order from masked positions together with original tokens. We refer this task to as masked position modeling (MPM) for convenience. MPM is essentially complementary to MLM task and is of vital importance in word order modeling (See Appendix A.3). Analog to MLM, the objective of MPM is formulated as below

$$\mathcal{L}_{\text{MPM}} = -\sum_{j \in \mathcal{Z}} \log Pr(p_j | \{X, \tilde{P}\}; \theta) \quad (4)$$

where $\mathcal{Z}$ is the set of indices of masked positions. TACO sums the losses from both MLM and MPM objectives to jointly learn WE and PE representations in a self-supervised way. Namely, a pair of examples derived from the same sentence are fed into the model to produce one summed loss for pre-training.

4 Experiments

In this section, we present fine-tuning results on 12 NLP tasks.

As introduced in previous section, our model requires more hyper-parameters to determine the structure due to more complex design than conventional BERT and DeBERTa. We conduct several experiments to search for an optimal set of hyper-parameters (See Appendix A.5). As a consequence, we follow the optimal settings listed in Table 7 to pre-train a base-size TACO model. For training data, we use 160GB text corpora from English Wikipedia[1], BookCorpus (Zhu et al., 2015), OpenWebText[2] and CommonCrawl News (Liu et al., 2019). We use 6 machines (48 A30 GPUs) to train the models. It takes 5 weeks to train a TACO-base model with 1200 batch size and 500K steps.

4.1 Fine-tuning Results of Downstream Tasks

GLUE We compare TACO-base to several strong baselines, i.e., BERT (Devlin et al., 2019), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019) and DeBERTa (He et al., 2020), on 8 GLUE tasks, and report the results on the development set of each task in Table 1. All the models are in base-size for fair comparison. Although TACO has not completely converged (with only 46K pre-training steps so far), we still achieve comparable performance to classic BERT baseline (84.5 vs 83.1). Specifically, TACO slightly exceeds BERT on STS (+0.4%), MRPC (+3.1%), and RTE (+10.8%) tasks. However, our model performs slightly worse in the

[1]https://dumps.wikimedia.org/enwiki/
[2]http://Skylion007.github.io/OpenWebTextCorpus

Model	MNLI-m/mm Acc	QQP Acc	QNLI Acc	SST-2 Acc	CoLA MCC	STS PCC	MRPC Acc	RTE Acc	Avg
BERT	84.3/84.7	91.3	91.7	93.2	58.9	89.5	87.3	68.6	83.1
XLNet	86.8/-	91.4	91.7	94.7	60.2	89.5	88.2	74.0	84.6
RoBERTa	87.6/-	91.9	92.8	94.8	63.6	91.2	90.2	78.7	86.4
DeBERTa*	88.7/88.5	91.8	93.7	95.5	64.3	92.1	91.2	86.3	87.9
TACO (46K)	84.2/84.8	91.3	91.0	92.3	57.3	89.9	90.4	79.4	84.5

Table 1: Results of base-size models on the development set of the GLUE benchmark. * The DeBERTa-base models are fine-tuned using the code and model weights released in HuggingFace (Wolf et al., 2019).

Model	SQuAD 1.1 EM/F1	SQuAD 2.0 EM/F1	RACE Acc	SWAG Acc
BERT	80.8/88.5	73.7/76.3	65.0	81.6
RoBERTa	84.6/91.5	80.5/83.7	-	84.0*
DeBERTa	87.2/93.1	83.1/86.2	72.2*	86.3*
TACO (46K)	84.5/91.1	78.7/81.7	68.4	75.1

Table 2: Results of base-size models on the development set of SQuAD v1.1/v2.0, RACE and SWAG. * These results are collected by fine-tuning corresponding models using the code and model weights released in HuggingFace (Wolf et al., 2019).

natural language inference tasks including MNLI and QNLI, indicating that knowledge inference probably needs more training to learn high-level semantic patterns.

Question Answering We further validate TACO-base model on different types of question-answering tasks (Refer to Appendix A.4 for detailed description). The performance of TACO-base and other baselines are collected in Table 2. We observe significant improvements of TACO on four question-answering tasks compared to BERT, +2.6% (F1) on SQuAD 1.1 and +5.4% (F1) on SQuAD 2.0, +3.4% on RACE. These improvements over BERT partially demonstrate the effectiveness of several significant modifications introduced in the proposed TACO model.

According to Table 1 and 2, we inescapably discover a performance gap between our proposed TACO model and other two baselines, RoBERTa and DeBERTa. The primary reason responsible for these unexpected results is probably insufficient pre-training, since our fine-tuning models are derived from an intermediate checkpoint with only 46K pre-training steps. Based on the superior performance to BERT, we believe our model would achieve much better results when the pre-training is complete.

Even though the proposed TACO has not show superior performance than RoBERTa and DeBERTa yet, it is still a promising model with comparable ability to BERT and with great potential to outperform RoBERTa and DeBERTa.

4.2 Ablation Study

We present an ablation study to evaluate the effectiveness of different modifications introduced in TACO. We thus develop two variations by eliminating two particular components:

- The first shares the same architecture as TACO-base, except the exclusion of MPM loss in pre-training.

- Another variant is the TACO-base model without convolution in the computation of attention weights. Interactive attention maps are added to corresponding content maps as DeBERTa (He et al., 2020), while position maps are used to independently generate attention weights for PE heads. Attention maps from previous layer are still incorporated before softmax.

We follow the same settings as BERT-base to pre-train ablation models. Table 3 summarizes the results on four benchmark datasets. We use a check-

Model	#Param	MNLI-m/mm Acc	SQuAD 1.1 EM/F1	SQuAD 2.0 EM/F1	RACE Acc
BERT	108.9M	84.3/84.7	81.0/88.5	73.7/76.3	65.0
RoBERTa	124.1M	84.7/-	-/90.6	-/79.7	65.6
DeBERTa	138.6M	86.3/86.2	86.1/92.1	79.3/82.5	71.7
TACO (42K)	100.1M	84.0/84.3	83.9/90.5	77.2/80.3	65.9
– MPM loss	100.1M	83.8/84.2	83.5/90.4	76.7/79.8	65.4
– Attention Convolution	100.1M	83.7/84.0	83.4/90.2	76.0/79.5	64.7

Table 3: Ablation study of the TACO-base model. All models are pre-trained over Wikipedia and BookCorpus for 1M steps with a batch size of 256.

point pre-trained with 42K steps to evaluate the fine-tuning performance of ablation models. Obviously, we can affirm the importance of MPM task through the decreased performance of the first variant (-0.2% on MNLI-matched, -0.1% on SQuAD 1.1, -0.5% on SQuAD 2.0 and -0.5% on RACE). Despite that, the convolution also helps to better capture complex patterns with regard to both semantic relation and word order (84.0 vs 83.7 on MNLI-matched, 90.5 vs 90.2 on SQuAD 1.1, 80.3 vs 79.5 on SQuAD 2.0, and 65.9 vs 64.7 on RACE). We also observe significant improvements of TACO model over BERT on question-answering tasks, and achieve comparable results with RoBERTa except for MNLI task.

5 Conclusion

We present a novel architecture TACO by explicitly disentangling positional representations and incorporating convolution over multi-source attention maps before softmax in self-attention. We pre-train a language model based on this architecture with an additional MPM task to jointly learn positional representations concerning word order. MPM is essentially complementary to the classic MLM objective by introducing word order modeling to enhance robustness of learned representations against minor permutations of a natural sentence. Experiments show that our TACO model integrating above novelties consistently improves the end-task results on several language understanding benchmarks.

References

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, et al. 2020a. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *International Conference on Machine Learning*, pages 642–652. PMLR.

Siqi Bao, Huang He, Fan Wang, Hua Wu, and Haifeng Wang. 2020b. Plato: Pre-trained dialogue generation model with discrete latent variable. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 85–96.

Tyler A Chang, Yifan Xu, Weijian Xu, and Zhuowen Tu. 2021. Convolutions and self-attention: Re-interpreting relative positions in pre-trained language models. *arXiv preprint arXiv:2106.05505*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. *Advances in neural information processing systems*, 28:3079–3087.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G 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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Ruining He, Anirudh Ravula, Bhargav Kanagal, and Joshua Ainslie. 2021. RealFormer: Transformer likes residual attention. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 929–943, Online. Association for Computational Linguistics.

Jeremy Howard 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.

Zi-Hang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. 2020. Convbert: Improving bert with span-based dynamic convolution. *Advances in Neural Information Processing Systems*, 33.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Guolin Ke, Di He, and Tie-Yan Liu. 2020a. Rethinking positional encoding in language pre-training. In *International Conference on Learning Representations*.

Pei Ke, Haozhe Ji, Siyang Liu, Xiaoyan Zhu, and Minlie Huang. 2020b. Sentilare: Linguistic knowledge enhanced language representation for sentiment analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6975–6988.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794.

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2021. A survey of transformers. *arXiv preprint arXiv:2106.04554*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. 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 3730–3740.

Yinhan Liu, 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. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Thang M Pham, Trung Bui, Long Mai, and Anh Nguyen. 2020. Out of order: How important is the sequential order of words in a sentence in natural language understanding tasks? *arXiv preprint arXiv:2012.15180*.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26.

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

Colin Raffel, 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*, 21:1–67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Peter Shaw, 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, Volume 2 (Short Papers)*, pages 464–468.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. 2019. Structbert: Incorporating language structures into pre-training for deep language understanding. In *International Conference on Learning Representations*.

Yujing Wang, Yaming Yang, Jiangang Bai, Mingliang Zhang, Jing Bai, J. Yu, Ce Zhang, Gao Huang, and Yunhai Tong. 2021. Evolving attention with residual convolutions. In *ICML*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

A Appendix

A.1 Multi-source Attention Maps

Content Attention The first group is content attention by employing query and key content to measure the attention level between a query and a key. The query content is obviously represented by its WE hidden vector, so is the key content. This type of attention maps, which are included in nearly all pre-trained language models such as BERT (Devlin et al., 2019), DeBERTa (He et al., 2020) and TUPE (Ke et al., 2020a), is the most fundamental and crucial source to model global and local dependencies within the input sentence. We compute content attention map of each WE head following the equation below.

$$\mathbf{Q}_{\text{WE}} = \mathbf{H}_{\text{WE}}\mathbf{W}^{Q}_{\text{WE}}, \quad \mathbf{K}_{\text{WE}} = \mathbf{H}_{\text{WE}}\mathbf{W}^{K}_{\text{WE}}$$
$$\mathbf{A}_{\text{W}} = \frac{\mathbf{Q}_{\text{WE}}\mathbf{K}^{T}_{\text{WE}}}{\sqrt{d_k}} \tag{5}$$

where $\mathbf{W}^{Q}_{\text{WE}}, \mathbf{W}^{K}_{\text{WE}} \in \mathbb{R}^{d_{\text{WE}}\times d_k}$ are parameter matrices to project WE vectors into the same d_k-dimensional space.

Position Attention Analog to content attention, the second group is positional attention, which uses query's and key's positional information to compute attention scores as employed by TUPE (Ke et al., 2020a). Specifically, positional information of a query and a key is explicitly expressed by their corresponding PE vectors. We believe that positional attention is very helpful to model word orders of a given sentence. The positional attention map of each PE head is also computed via scaled dot-product similar to content attention.

$$\mathbf{Q}_{\text{PE}} = \mathbf{H}_{\text{PE}}\mathbf{W}^{Q}_{\text{PE}}, \quad \mathbf{K}_{\text{PE}} = \mathbf{H}_{\text{PE}}\mathbf{W}^{K}_{\text{PE}}$$
$$\mathbf{A}_{\text{P}} = \frac{\mathbf{Q}_{\text{PE}}\mathbf{K}^{T}_{\text{PE}}}{\sqrt{d_k}} \tag{6}$$

where $\mathbf{W}^{Q}_{\text{PE}}, \mathbf{W}^{K}_{\text{PE}} \in \mathbb{R}^{d_{\text{PE}}\times d_k}$ are parameter matrices to project PE vectors. The dimension d_k of PE queries and keys of each head is set the same as that of WE queries.

Interactive Attention The last group is interactive attention, which has two different forms. One form is to use query content and key's position to calculate content-to-position attention score as DeBERTa (He et al., 2020). Accordingly, the other is position-to-content term using query's positional information and key content. Since both terms integrate token content and position in a same low-dimensional space and then assess the interaction between a query and a key, we unite them as interactive attention implying that it bridges the gap between the representation subspaces of token content and position.

Intuitively, we use relative positional encodings to compute interactive attention maps as DeBERTa (He et al., 2020), which can be expressed as:

$$\mathbf{K}_{\text{R}} = \mathbf{R}_{\text{PE}}\mathbf{W}^{K}_{\text{R}}, \quad \mathbf{A}^{i,j}_{\text{QR}} = \frac{\mathbf{Q}^{i,\cdot}_{\text{WE}}\left(\mathbf{K}^{r_{ij}}_{\text{R}}\right)^{T}}{\sqrt{d_k}}$$
$$\mathbf{Q}_{\text{R}} = \mathbf{R}_{\text{PE}}\mathbf{W}^{Q}_{\text{R}}, \quad \mathbf{A}^{i,j}_{\text{RK}} = \frac{\mathbf{Q}^{r_{ji}}_{\text{R}}\left(\mathbf{K}^{j,\cdot}_{\text{WE}}\right)^{T}}{\sqrt{d_k}} \tag{7}$$

where r_{ij} is the relative distance from a position i to j, defined as $r_{ij} = \max(-k, \min(k, j-i))$ as in Shaw et al. (2018). $\mathbf{R}_{\text{PE}} \in \mathbb{R}^{(2k+1)\times d_{\text{PE}}}$ is relative positional encodings for total $2k{+}1$ relative distances. $\mathbf{W}^{K}_{\text{R}}, \mathbf{W}^{Q}_{\text{R}} \in \mathbb{R}^{d_{\text{PE}}\times d_k}$ are parameter matrices to project relative positional vectors. $\mathbf{K}^{r_{ij}}_{\text{R}}$ represents the projected vector of distance r_{ij}, i.e., the r_{ij}-th row vector of $\mathbf{K}_{\text{R}}$. The maximum relative distance k is set to 511 due to the absolute positions ranging from 0 to 512 (exclusive) in our model. Besides, relative positional encodings are shared across all layers of TACO model.

Despite of relative encodings, we in fact design two more forms of interactive attention using positional representations $\mathbf{H}_{\text{PE}}$ of each layer. We will investigate them in future work due to paper limit.

A.2 Position Masking

Slightly different from the masking strategy for tokens, 15% positions are randomly masked by performing the following procedure at the span level instead of for each position individually:

- Suppose that we want to mask a span (2, 3) in the original position sequence (1, 2, 3, 4, 5, 6) for a sentence with six tokens.
- 80% of the time: Replace the positions with a mask position such as -1 in our model, e.g., (1, 2, 3, 4, 5, 6) $\rightarrow$ (1, -1, -1, 4, 5, 6).
- 10% of the time: Shuffle the positions, e.g., (1, 2, 3, 4, 5, 6) $\rightarrow$ (1, 3, 2, 4, 5, 6). If the span just contains one position such as (2), a random position in the range from 0 to maximum sequence length is sampled to replace the single position, e.g., (1, 2, 3, 4, 5, 6) $\rightarrow$ (1, 8, 3, 4, 5, 6).
- 10% of the time: Keep the positions unchanged, e.g., (1, 2, 3, 4, 5, 6) $\rightarrow$ (1, 2, 3, 4, 5, 6).

The shuffle operation in the procedure above coincidently accords with the word structural objective adopted by StructBERT (Wang et al., 2019), which is to reconstruct the right order of certain number of intentionally shuffled tokens.

A.3 Relation between MLM and MPM tasks

Despite of analogy in masking schemes and training objectives, the MLM and MPM tasks essentially supplement each other from the linguistic perspective. We take the sentence "*Bob once asked Alice to borrow a book.*" as an example to expound the complementary effect of these two tasks. For a corrupted sentence such as "*Bob once asked Alice to* [MASK] *a book.*", MLM requires a model to fill in the masked token. In this task, the model knows which position the masked token locates in and thus predicts what is correct in this position. Contrarily, MPM urges the model to find correct position for each given candidate (e.g. "*asked*" and "*borrow*") in the corrupted sentence like "*Bob once* _ *Alice to* _ *a book .* _". Owing to loose restriction on the range of predicted positions, there is an inevitable shortcoming inherent in this MPM task. That is, the predicted position of each candidate may be overlapped with existing positions occupied by known tokens or far beyond the sentence scope. For example, TACO predicts position 2 for "*asked*" which conflicts with "*once*" in the same position, or 12 for "*borrow*" which exceeds the sentence length. Therefore, MPM is also a crucial pre-training task in spite of less challenge than MLM because of smaller number of pre-defined possible positions than the size of token vocabulary. To sum up, MLM and MPM together encourage a model to learn contextualized representations of token content and position, and consequently enhance positional robustness against changes of word orders in language modeling.

A.4 Datasets

Datasets	#Train/#Dev/#Test
Single-Sentence Classification	
CoLA (Acceptability)	8.5k/1k/1k
SST-2 (Sentiment)	67k/872/1.8k
Pairwise Text Classification	
MNLI (NLI)	393k/20k/20k
RTE (NLI)	2.5k/276/3k
QNLI (NLI)	105k/5.5k/5.5k
WNLI (NLI)	634/71/146
QQP (Paraphrase)	364k/40k/391k
MRPC (Paraphrase)	3.7k/408/1.7k
Text Similarity	
STS-B (Similarity)	7k/1.5k/1.4k

Table 4: Statistics of GLUE datasets.

GLUE The General Language Understanding Evaluation (GLUE) benchmark is a collection of 9 natural language understanding (NLU) tasks. Table 4 summarizes the dataset details for GLUE. We don't use WNLI dataset for evaluation to keep consistent with previous work (Devlin et al., 2019).

SQuAD The Standford Question Answering Dataset (SQuAD) is a collection of 100k crowdsourced question/answer pairs (Rajpurkar et al., 2016, 2018). For a given question and paragraph, the task is to predict the answer span from the paragraph. We fine-tune on two versions of SQuAD:

Hidden size (WE, PE)	**#Param**	**MNLI-m/mm** Acc	**SQuAD 1.1** EM/F1	**SQuAD 2.0** EM/F1	**RACE** Acc
(648, 120)	80.4M	**82.0/82.7**	**82.1/89.0**	73.2/76.5	**62.0**
(576, 192)	71.1M	82.0/82.1	81.3/88.6	**73.2/76.5**	61.2
(504, 264)	64.3M	81.4/82.0	81.2/88.4	72.8/75.8	59.3

Table 5: Performance of TACO models with different WE & PE hidden sizes.

v1.1 and v2.0. In v1.1, the context always contains an answer, whereas in v2.0 some questions are not answered in the provided context, making the task more challenging.

RACE The ReAding Comprehension from Examinations (RACE) is a large-scale machine learning comprehension dataset, and is collected from English examinations in China (Lai et al., 2017). In RACE, each passage is associated with multiple questions, and the task is to select one correct answer from four options for every question.

SWAG The Situations With Adversarial Generations (SWAG) dataset is a large-scale adversarial dataset for the task of grounded commonsense inference, which unifies natural language inference and physically grounded reasoning (Zellers et al., 2018). SWAG contains 113k sentence-pair completion examples that evaluate grounded commonsense inference. For a given sentence, the task is to predict the most plausible continuation among four choices.

A.5 Comparison of Different Structures

To fairly and quickly compare different structures, we conduct several light-weight experiments following the settings of BERT-base model (Devlin et al., 2019), except that we use AdamW optimizer (Loshchilov and Hutter, 2018) to pre-train for 120k steps in total and learning rate warmup over the first 10000 steps. For training data, we only use Wikipedia and BookCorpus (Zhu et al., 2015), which leads to a total data size of 16GB after preprocessing. As in RoBERTa (Liu et al., 2019), the benchmark datasets include MNLI (matched/mismatched), SQuAD v1.1/v2.0 and RACE to validate the performance of pre-trained models. We carefully evaluate a number of configurations when pre-training TACO model, including hidden size and the number of attention heads.

Hidden sizes We first evaluate the effect of different WE and PE hidden size. In order to match the settings of base-size classic models such as BERT and DeBERTa, we attempt to choose hidden sizes for WE and PE so that the summation equals to 768, which indicates that the complete representation of a token from TACO has the same dimension as BERT-base or DeBERTa-base hidden vectors. For this experiment, we specify the attention heads of WE and PE to 6 and use relative position encodings with dimension d_{PE} to compute interactive maps. The fine-tuning results are presented in Table 5.

It is found that that TACO with 120-dimensional position encodings outperforms other models in MNLI (matched & mismatched), SQuAD v1.1 and RACE datasets, and achieves comparable results in SQuAD v2.0 to the model of 192 PE hidden size. According to Table 5, TACO basically performs worse on downstream tasks when the representation space of token content with dimension d_{WE} becomes smaller. It implies that a pre-trained model with more free parameters usually has a greater potential to capture complex semantic patterns. Besides, the increase of position-related parameters can not compensate the performance degradation caused by reduced WE hidden size. Although small-sized positional space results in a large model, we can not choose too small values for PE hidden size, which may lead to the inability to model word orders through the MPM task. We use PE hidden size of 120 for the following TACO models of base size.

Attention heads Next, we intend to choose proper numbers of WE and PE attention heads. We follow most settings as previous TACO model with $d_{\text{PE}} = 120$, except for different numbers of attention heads.

As seen in Table 6, our model exhibits superior performance on all five tasks if we increase the number of WE heads from 6 to 12. Contrarily, more PE heads tends to impede the improvement of learning capability of TACO model. This abnor-

Attention Heads (WE, PE)	**#Param**	**MNLI-m/mm** Acc	**SQuAD 1.1** EM/F1	**SQuAD 2.0** EM/F1	**RACE** Acc
(6, 6)	80.4M	82.0/82.7	82.1/89.0	73.2/76.5	62.0
(12, 6)	87.5M	82.5/82.7	**82.8/89.4**	**75.5/78.6**	**62.6**
(12, 12)	88.6M	**82.5/82.7**	82.2/89.0	74.0/77.6	62.2

Table 6: Performance of TACO models with different numbers of WE & PE attention heads.

mal phenomenon of decreased performance when increasing model parameters may be induced by insufficient pre-training, namely the model does not fully converge after 120k steps. We simply leave this uncertainty to future work and use 12 WE heads and 6 PE heads as default base settings.

In summary, we finally choose 648 and 120 as WE and PE hidden sizes, 12 WE attention heads and 6 PE heads to pre-train TACO-base model. Additionally, we enlarge the dimension of relative position embeddings to 768 since relative distance may enhance the capability of our model in both MLM and MPM tasks.

A.6 Pre-training details

As shown in Table 7, the main hyperparameters used for pre-training TACO-base is presented. The hidden size and number of attention heads for WE and PE are chosen according to comparison results in section A.5.

Number of Layers	12
Hidden size (WE, PE)	(648, 120)
FFN inner hidden size (WE, PE)	(2592, 480)
Attention heads (WE, PE)	(12, 6)
Attention head size	64
Dropout	0.1
Attention Dropout	0.1
Warmup Steps	10000
Peak Learning Rate	1e-4
Batch Size	1200
Weight Decay	0.01
Max Steps	500K
Learning Rate Decay	Linear
Adam ϵ	1e-6
Adam β_1	0.9
Adam β_2	0.999
Gradient Clipping	1.0

Table 7: Hyperparameters for pre-training TACO-base.

A.7 Fine-tuning details

Table 8 reports the hyperparameters used for fine-tuning TACO-base on GLUE, SQuAD, RACE, and SWAG benchmarks. We report the median result of five runs starting from different seeds for each set of hyperparameters for every task.

Datasets	**Batch Size**	**Learning Rate**	**Max Epochs**	**Weight Decay**
GLUE	{16, 32}	{1e-5, 2e-5, 3e-5, 5e-5}	{3, 4, 5, 6}	0.01
SQuAD	32	{1e-5, 2e-5, 3e-5, 5e-5}	{2, 3, 4, 5}	0.01
RACE	16	1e-5	6	0.1
SWAG	{16, 32}	{1e-5, 1.5e-5, 2e-5}	{4, 5}	0.01

Table 8: Hyperparameters for fine-tuning TACO-base and ablation models on GLUE, SQuAD, RACE, and SWAG.