A Length-Extrapolatable Transformer

Yutao Sun[1]*, **Li Dong**[2], **Barun Patra**[2], **Shuming Ma**[2], **Shaohan Huang**[2]
Alon Benhaim[2], **Vishrav Chaudhary**[2], **Xia Song**[2], **Furu Wei**[2]
Tsinghua University[1]
Microsoft[2]
https://github.com/microsoft/torchscale

Abstract

Position modeling plays a critical role in Transformers. In this paper, we focus on length extrapolation, i.e., training on short texts while evaluating longer sequences. We define *attention resolution* as an indicator of extrapolation. Then we propose two designs to improve the above metric of Transformers. Specifically, we introduce a relative position embedding to explicitly maximize attention resolution. Moreover, we use blockwise causal attention during inference for better efficiency. The proposed architecture is named **L**ength-**Ex**trapolatable (LEX) Transformer. We evaluate different Transformer variants on language modeling. Experimental results show that our model achieves better performance in both interpolation and extrapolation settings. The code will be available at https://aka.ms/LeX-Transformer.

1 Introduction

Transformer (Vaswani et al., 2017) has shown strong performance in NLP and become a de-facto backbone (Dosovitskiy et al., 2020; Radford et al., 2021; Wang et al., 2022). However, most of them have a crucial shortcoming: they can only deal with the in-distribution size of inputs. Figure 1 shows that the perplexity of previous Transformers increases rapidly when the input sequence is getting longer. It is usually infeasible to train a model with all possible input lengths. Therefore, a length-extrapolatable Transformer is essential for wider usage.

In sequence modeling, position information plays a crucial role in building the correct representation and understanding of the latent meaning. For Recurrent Neural Networks such as LSTM (Hochreiter and Schmidhuber, 1997), the calculation is done along the sequence order in O(N) time. However, the parallel attention module

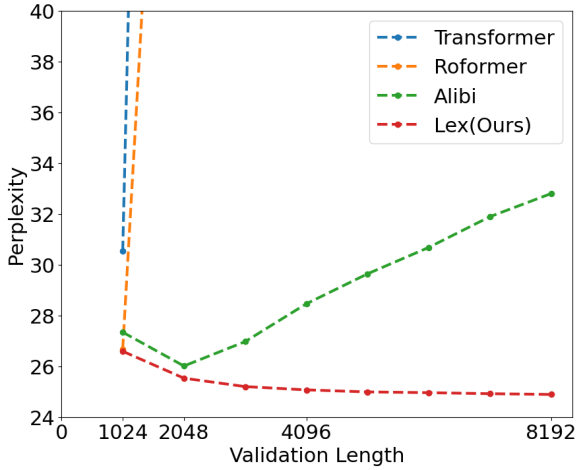

Figure 1: The perplexity of different Transformer designs with various input lengths.

makes it hard to encode position effectively. First, Vaswani et al. (2017) proposes absolute sinusoidal position embedding, and Devlin et al. (2019) adjusts it to a learnable one. The absolute design is computation-efficient, but not comparable with subsequent relative ones (Shaw et al., 2018; Su et al., 2021; Press et al., 2021). Among many relative position embeddings, ROPE (Su et al., 2021) shows better performance and is used to many PLMs such as PaLM (Chowdhery et al., 2022). However, it can't deal with sequences with exceeding length. Alibi (Press et al., 2021) mitigates the extrapolation problem but sacrifices the general performance.

Since different strategies concentrate on some part of the position feature, it is essential to build a comprehensive view and guide the Transformer's design systematically. First, a Transformer should be sensitive to order. Otherwise, it will degenerate into a bag-of-word model which confuses the whole meaning. Then, position translation can't hurt the representation, especially, when a prefix is added to the target sentence, the representation should stay the same with an attention mask on the prefix. After that, a good sequence model needs to

*Work done during internship at Microsoft Research.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 14590–14604
July 9-14, 2023 ©2023 Association for Computational Linguistics

Models	Translation Invariance	Length Extrapolation
Absolute Position Modeling		
Transformer (Sinusoidal)	✘	✘✘
GPT-2 (Learnable)	✘	✘✘
Relative Position Modeling		
PaLM / Roformer (RoPE)	✔	✘
T5	✔	✘
BLOOM / Alibi	✔	✔
LEX Transformer (Ours)	✔	✔✔

Table 1: Position modeling capabilities of Transformer variants for language modeling.

deal with any input length. As illustrated before, the length problem is not universal but special for Transformer. Especially, when a Transformer is pre-trained under a maximal length, it is not affordable to re-train for applying to tasks with longer sequences. Finally, when a Transformer satisfies the principles above, we evaluate its performance, which requires thorough experiments and empirical analysis.

Considering all the properties above, we propose Extrapolatable Position Embedding (XPOS), which is a universal-good design for Transformers. Based on ROPE's design, we propose *attention resolution* as a metric to measure position monotonicity. Then, we generalize its mathematical form, where an exponential decay is added to the rotation matrix. XPOS preserves the advantage of ROPE, and behaves stably at long-term dependency. Besides, inspired by sparse attention methods (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020; Xiong et al., 2021), we choose blockwise causal attention to increase attention resolution, which improves the performance of length extrapolation for language modeling.

We train different Transformers from scratch. We evaluate models on PG22 and QMSum (Zhong et al., 2021) with various input lengths. On the interpolation experiments, LEX Transformer reaches minimal perplexity. In the extrapolation experiments, our methods can continue decreasing the perplexity while other methods either can't extrapolate (i.e., perplexity increases) when the input length is very long. Figure 1 shows clearly that LEX Transformer has an opposite tendency compared with others.

We summarize our contributions as follows:

- We summarize the design principles of Transformers for position modeling.
- We define attention resolution to indicate a Transformer's capability on encoding position.
- We propose an extrapolatable position embedding and use blockwise causal attention to improve length extrapolation.
- We conduct experiments on language modeling and show that the proposed LEX Transformer achieves strong performance on both short and long texts.

2 Design Principles of Transformers for Position Modeling

2.1 Order Variance

A transformer without position information is actually a bag-of-word model. Although bag-of-words models can achieve comparable performance for some tasks (Wang et al., 2020a), position information is essential for sequence modeling. Most of the existing position modeling satisfies this goal (Vaswani et al., 2017; Devlin et al., 2019; Shaw et al., 2018; Wang et al., 2020a; Raffel et al., 2020; Su et al., 2021). With effective position information, Transformer models should be variant with permuting the order (Dufter et al., 2022). Give a permutation function $P_\pi(X) : [x_1, x_2, ..., x_n] \rightarrow [x_{\pi_1}, x_{\pi_2}, ..., x_{\pi_n}]$, where $[\pi_1, \pi_2, ..., \pi_n]$ is a random order, a Transformer model $f(input)$ should satisfy:

$$f(P_\pi(X)) \neq P_\pi(f(X)) \quad (1)$$

2.2 Translation Invariance

The representation of a sequence should be robust with the translation of its positions. For instance, a sentence's meaning is invariant with padding before or after the whole sentence. Similar

to (Wang et al., 2020a), we give a general form for translation invariance: given a Transformer model $f(\text{input}, \text{mask})$, any input sequence $X = [x_0, x_1, ..., x_n]$ with mask $M = [m_0, m_1, ..., m_n]$, the output should be same with the padding ones:

$$\begin{aligned} X_{\text{pad}} &= [0]_i \oplus X \oplus [0]_j \\ M_{\text{pad}} &= [0]_i \oplus M \oplus [0]_j \\ f(X, M) &= f(X_{\text{pad}}, M_{\text{pad}})[i : i + n] \end{aligned} \tag{2}$$

Relative positions (Shaw et al., 2018; Raffel et al., 2020; Wang et al., 2020a; Su et al., 2021) satisfy this condition, while most of the absolute positions do not (Vaswani et al., 2017; Devlin et al., 2019). Although sinusoidal embedding has a similar property (Vaswani et al., 2017): PE_{pos+k} can be represented as a linear function of PE_{pos}, the addition operation in the initial word embedding messes the attention weight, where the spread form of QK^T has 4 components whose geometric connection with position is unclear.

2.3 Length Extrapolation

As the cost of pre-training is getting bigger due to the larger model size and corpus, it is infeasible to retrain a model for a longer context. A Transformer model with a suitable design should be capable of dealing with any input length.

First, learnable absolute position embedding (Devlin et al., 2019) is not able to extrapolate because it does not have any pre-defined position knowledge. With the evaluation of perplexity on different lengths (Press et al., 2021), almost every position embedding's performance drops significantly (Vaswani et al., 2017; Raffel et al., 2020; Su et al., 2021). Alibi (Press et al., 2021) solves this problem by adding an exponential decay on the attention matrix, which lower the influence of out-of-distribution position like a soft sliding window. However, the absence of long-term dependency contributes to a performance drop compared with other relative strategies. Table 2 shows that Alibi's perplexity is larger than RoPE by about 0.3.

However, the extrapolation ability needs a systematic design where position embedding is a crucial but not only component. With the proper attention mask, the relative position can deal with long text. The ideal situation is to use the long context in the right way, in that case, the perplexity should decrease as the input length increases.

3 A Length-Extrapolatable Transformer

We define attention resolution as the indicator of the Transformer's capability on encoding position in Section 3.1. Then we propose two ways to maximize the resolution metric, i.e., improve the length interpolation and extrapolation of Transformers. First, we introduce a relative position encoding method (Section 3.2) to explicitly maximize attention resolution. Second, we propose to use blockwise causal masking (Section 3.3) during extrapolation inference for improved resolution.

In the following section, we denote d as the hidden dimension and l as the input length. For each attention layer, query, key, and value are calculated by input x: $q = W_Q x, k = W_K x, v = W_V x$.

3.1 Attention Resolution

The monotonicity of attention scores is essential to represent distance in language models. In an attention layer of the vanilla Transformer, we measure the attention score expectation as $s[n]$ when the distance of two tokens is n:

$$s[n] = \mathop{\mathbb{E}}_{0 \le i \le N} \left(\frac{q_{i+n} k_i^T}{\sqrt{d}}\right) \tag{3}$$

We define *attention resolution* $R(s)$ as a metric to evaluate attention's ability to recognize position:

$$R(s) = \sum_{i=0}^{N} \frac{e^{s[i]}(e^{s[i]} - e^{s[i+1]})}{(\sum_{i=0}^{N} e^{s[i]})^2} \tag{4}$$

First, $s[i] > s[i+1]$ is preferred to ensure monotonicity. Besides, we implement $\mathrm{softmax}$ operation on $s[n]$ to simulate the attention probability. To mitigate the influence of long-tail distribution, the factor $e^{s[i]}$ is multiplied. We can estimate $s[n]$ and $R(s)$ quantitatively when we design Transformers.

3.2 Improve Resolution by Position Encoding

Su et al. (2021) propose that by adding absolute position embedding on query and key, the attention matrix is actually encoded with relative position information. RoPE shows a strong performance in interpolation tasks, but its $s[n]$ oscillates dramatically in Figure 2, which harms the *resolution*.

We use a similar but generalized strategy to improve *resolution*. First, a pseudo inner product is defined as $\langle x, y \rangle = \sum Re(x_i \cdot y_i^*)$, which is consistent with the exact inner product's definition when we map $\mathbb{C}^{d/2} \to \mathbb{R}^d$. Before calculating attention,

the query and key are encoded with position information. Generally, the attention function is as follows:

$$a_{ij} = \frac{\langle f_q(q_i, i), f_k(k_j, j)\rangle}{\sqrt{d}}$$
$$o_i = \sum_{j=0}^{i} \frac{e^{a_{ij}}}{\sum_{j=0}^{i} e^{a_{ij}}} v_j \tag{5}$$

Formally, the encoding must satisfy:

$$\langle f_q(q, n+r), f_k(k, n)\rangle = \langle f_q(q, r), f_k(k, 0)\rangle \tag{6}$$

A simple solution is as follows:

$$f_q(q, n) = A_q q e^{\lambda n}$$
$$f_k(k, n) = A_k k e^{-\lambda^* n} \tag{7}$$

The scaling factor A_q, A_k is unnecessary because q, k is obtained by a linear transformation. Since $\lambda \in \mathbb{C}^{d/2}$, it can be represented as $\lambda = \xi + i\theta$ where $\xi, \theta \in \mathbb{R}^{d/2}$:

$$f_q(q, n) = q e^{\xi n + i\theta n}$$
$$f_k(k, n) = k e^{-\xi n + i\theta n} \tag{8}$$

If $\xi = 0$, the form is the same as RoPE (Su et al., 2021). Geometrically, the transformation provides a rotation on vectors. If the relative angle between q and k is larger, the inner product is smaller. However, the cosine function is not monotonic if the rotating angle is large than π, which causes an unstable phenomenon in that the expectation of the inner product oscillates dramatically with the growth of relative distance. Following the parameters (Vaswani et al., 2017; Su et al., 2021) $\theta = \{\theta_i = 10000^{-2i/d}, i \in [0, 1, ..., d/2]\}$, we will calculate the expectation as follows. For generative models, we assume $\mathbb{E}(\angle q) \leq \mathbb{E}(\angle k)$ to ensure the monotonicity:

$$\begin{aligned}
&\mathbb{E}[\langle q e^{m\xi + im\theta}, k e^{-n\xi + in\theta}\rangle] \\
=&\sum_{x=0}^{d/2} \mathbb{E}[Re(\boldsymbol{q}_x \boldsymbol{k}_x e^{(m-n)\xi_x + i(m-n)\theta_x})] \\
\leq&\sum_{x=0}^{d/2} Re(\mathbb{E}[|\boldsymbol{q}_x \boldsymbol{k}_x|] e^{(m-n)\xi_x + i(m-n)\theta_x}) \\
\propto&\sum_{x=0}^{d/2} cos(m-n)\theta_x e^{(m-n)\xi_x}
\end{aligned} \tag{9}$$

The inference here is different from (Su et al., 2021) because of two reasons: 1) there is an additional assumption brought by generative language models where $\mathbb{E}(\angle q) \leq \mathbb{E}(\angle k)$; 2) the inequality scaling of (Su et al., 2021) is too strong to lose generality. We calculate expectation instead of the upper bound.

Now we define a function to represent the property of relative position:

$$g_\zeta[n] = \sum_{i=0}^{d/2} \cos n\theta_i \zeta_i^n \tag{10}$$

$g[n]$ simplifies Equation 9 by defining $\zeta_i = e^{\xi_i}$. Stabilizing the $g[n]$ curve is intuitive. Although attention bias can achieve this goal, we try to avoid additional position calculations. Instead, we can accomplish this goal using a good ζ to maximize $R(g_\zeta)$.

Obviously, the oscillation mainly comes from large θ_i. Manually setting ζ can achieve this goal:

$$\widetilde{\zeta}_i = \frac{i/(d/2) + \gamma}{1 + \gamma} \in [0, 1] \tag{11}$$

where $\widetilde{\zeta}_i$ becomes smaller when θ_i is larger. In this way, we punish the oscillation of unstable dimensions and keep the distribution of stable ones.

Numerical optimization methods are tried to find optimal values for ζ. However, the results rely on the initial value and lack control when the hidden dimension changes. Besides, the numerical precision should be considered because of fp16's range. Finally, we find a sub-optimal solution by manually setting γ to both satisfy the resolution is recognizable ($R(g_\zeta)$ is partially optimized) and ζ_i^n can be represented by fp16 when n is big (8192 in our setting). Besides, in implementation, the position is re-scaled with base B in the exponential calculation to avoid overflow and underflow ($e^{\xi n} \to e^{\xi n/B}$ in Equation (8)). We use $\gamma = 0.4$ and $B = 512$ as the final implementation in LEX Transformer.

The curves of $\zeta = \mathbf{1}, \hat{\zeta}$ are shown in Figure 2. The default rotary embedding contributes to a dramatic oscillation, especially in the large relative distance, which causes bad extrapolation performance and restricts the model's convergence speed. After adding a decay, the curve is almost stable, especially on long-term dependency. What's more, it does not hurt pure rotation's fitting ability because $\zeta_i^n \approx 1$ when i is large or n is small. In that way, short-term and long-term dependencies are divided continuously.

Finally, we have Extrapolatable Position Embed-

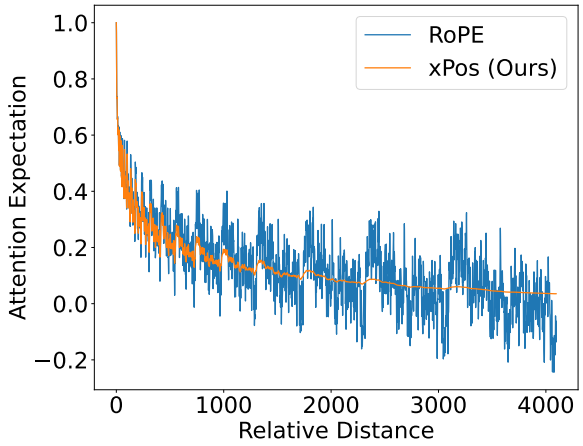

Figure 2: The long dependency curve of attention expectation. RoPE's dramatic oscillation confuses the attention resolution at long distances. In contrast, xPos provides stable and accurate position modeling.

Algorithm 1: Attention with xPos

def $\mathrm{rot}(x)$:
  return $[-x_1, x_0, -x_3, x_2, ...]$
Initialization:
$\theta_i = 1/10000^{2\lfloor i/2 \rfloor/d}, \theta \in \mathbb{R}^d$
$\hat{\zeta}_i = (2\lfloor i/2 \rfloor/d + \gamma)/(1+\gamma),\ \hat{\zeta} \in \mathbb{R}^d$
Input: $Q, K, V \in \mathbb{R}^{h \times l \times d}, M \in \mathbb{R}^{d \times d}$
$C_{mn} = \cos m\theta_n, C \in \mathbb{R}^{l \times d}$
$S_{mn} = \sin m\theta_n, S \in \mathbb{R}^{l \times d}$
$T_{mn} = \hat{\zeta}_n^m, T \in \mathbb{R}^{l \times d}$
$Q = Q \odot (C \odot T) + \mathrm{rot}(Q) \odot (S \odot T)$
$K = K \odot (C \odot T^{-1}) + \mathrm{rot}(K) \odot (S \odot T^{-1})$
$output = \mathrm{softmax}(\frac{QK^T}{\sqrt{d}} \cdot M)V$
return *output*

ding (xPos):

$$f_q(q,n) = \begin{pmatrix} q_1 \cos n\theta_1 \hat{\zeta}_1^{n/B} - q_2 \sin n\theta_1 \hat{\zeta}_1^{n/B} \\ q_2 \cos n\theta_1 \hat{\zeta}_1^{n/B} + q_1 \sin n\theta_1 \hat{\zeta}_1^{n/B} \\ \vdots \\ q_{n-1} \cos n\theta_{d/2} \hat{\zeta}_{d/2}^{n/B} - q_n \sin n\theta_{d/2} \hat{\zeta}_{d/2}^{n/B} \\ q_n \cos n\theta_{d/2} \hat{\zeta}_{d/2}^{n/B} + q_{n-1} \sin n\theta_{d/2} \hat{\zeta}_{d/2}^{n/B} \end{pmatrix}$$

$$f_k(k,n) = \begin{pmatrix} k_1 cosn\theta_1 \hat{\zeta}_1^{-n/B} - k_2 \sin n\theta_1 \hat{\zeta}_1^{-n/B} \\ k_2 \cos n\theta_1 \hat{\zeta}_1^{-n/B} + k_1 \sin n\theta_1 \hat{\zeta}_1^{-n/B} \\ \vdots \\ k_{n-1} \cos n\theta_{d/2} \hat{\zeta}_{d/2}^{-n/B} - k_n \sin n\theta_{d/2} \hat{\zeta}_{d/2}^{-n/B} \\ k_n \cos n\theta_{d/2} \hat{\zeta}_{d/2}^{-n/B} + k_{n-1} \sin n\theta_{d/2} \hat{\zeta}_{d/2}^{-n/B} \end{pmatrix} \tag{12}$$

In the implementation, the transformation for key and value can be easily calculated by parallel addition and multiplication as shown in Algorithm 1. Since position embedding's size $C, S, T \in \mathbb{R}^{l \times d}$ is much smaller than batched multi-head attention matrix and doesn't require gradients, the cost is almost the same with RoPE and bigger than Absolute Position with 6% additional time.

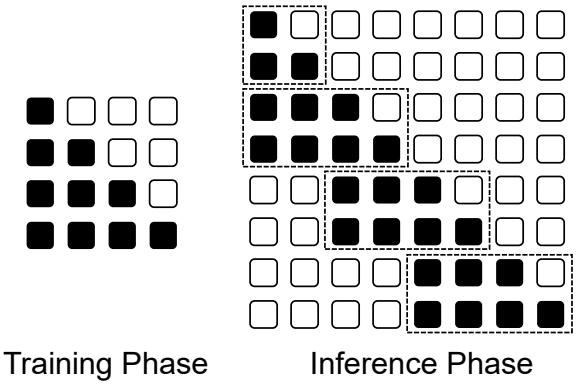

Figure 3: Our language model is trained on shorter texts in the same way as vanilla Transformers, i.e., using causal masking. During inference, we use blockwise causal attention for longer sequences, which recurrently reuses the overlapped parts (i.e., key and value vectors).

3.3 Blockwise Causal Attention

To deal with length extrapolation, a simple way to improve attention resolution (Section 3.1) is using windowed attention. During inference, we use blockwise masking (Dai et al., 2019; Zaheer et al., 2020; Xiong et al., 2021) for self-attention. Notice that other window strategies, such as sliding window (Child et al., 2019), also work. We use blockwise causal attention because it is cache-friendly and easy to implement.

As shown in Figure 3, if the pre-training length is l, we divide the query as blocks with $l/2$ length, and each query interacts with its own block and the last block. In this way, the context information can be delivered by the reuse of key and value. The window constraint helps models encode longer input with improved resolution.

Different from training a long-sequence model with a stop gradient, we use vanilla attention in the training phase, because the pre-training corpus is not very long on average. However, during the inference phase, when dealing with long sequences, we directly implement BCA to help the model to be more position-recognizable.

4 Experiments

4.1 Pre-training

To fairly evaluate different Transformer variants, we pre-train the Transformer from scratch. We use 1024 hidden dimensions, 16 heads, and 24 layers,

i.e., comparable to medium-size GPT-3 (Brown et al., 2020). The training corpus includes a subset of the Pile (Gao et al., 2020): Books3, OpenWebText2, Stack Exchange, PubMed Abstracts, Wikipedia, Gutenberg (PG-19), BookCorpus2, NIH ExPorter, and Pile-CC datasets. The training procedure is performed on 16×V100 GPUs. We use the tokenizer from GPT2 (Radford et al., 2019). The maximal length is 1024 for saving memory and extrapolation evaluation. The learning rate is 3×10^{-4} and polynomial decay is used to adjust the learning rate. The global batch size is 512 to follow GPT-3 (Brown et al., 2020), i.e., 0.5M token size. We use Adam (Kingma and Ba, 2015) optimizer with $\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^{-6}$. The code is based on TorchScale (Ma et al., 2022a).

4.2 Language Modeling

We measure perplexity on long document datasets, which can show the model's ability for long-dependency modeling. We use books from Project Gutenberg whose years are later than 2019 to ensure no overlap with PG19, and we name it as PG22. Besides, we pick QMSum (Zhong et al., 2021) from SCROLLS (Shaham et al., 2022) with above 9k length on average. We care about the performance on different input lengths to evaluate the model's interpolation and extrapolation capability. For experiment results in Table 2, we divide the same input into the target length to fairly compare the perplexity of different lengths.

For interpolation capability, we analyze the results where the length is no more than 1024. Since the validation distribution is very similar to training data, all Transformers' generalization capabilities are also close. XPOS have a stable advantage on others with a 0.09 perplexity drop on PG22, and 0.27 on QMSum, which proves that XPOS increases the interpolation ability.

For extrapolation lengths, we do not use BCA in other Transformers, and the following ablation study will discuss the performance with that. Press et al. (2021)'s experiment shows that most of the position strategies can't deal with input length longer than pre-training directly. XPOS shows a stable decrease when the sequence length increases, which satisfies the assumption that a longer context makes the prediction better. While others' perplexity increases when the input length is 4096.

To illustrate the tendency of perplexities, Figure 1 visualizes the relation between input length and perplexity. When the length is larger than 4096, Alibi's perplexity increases gradually. However, LEX's perplexity decreases continuously when the length extends to 8192.

The experiment shows that XPOS gets better performance on language modeling. With the stable advantage of any length, users can input any sentence freely without the concern of position. Besides, results also indicate that is not essential to build an explicit decay on the attention matrix, Instead, a proper design for an attention mask is actually better to deal with long-context tasks.

4.3 Measuring Resolution

We empirically evaluate the resolution of different Transformer variants. In the previous section, we define attention resolution as a quality indicator of position modeling in Transformers. The expectation of $s[n]$ is computed as:

$$\mathbb{E}[s[n]] = \frac{1}{N-n}\mathbb{E}[\sum_{i=n}^{N-1} a_{i(i-n)}] \tag{13}$$

where a_{ij} has the same meaning in Equation 5.

Then the attention resolution can be calculated by combining Equation (4) and Equation (13). The final expectation is averaged over input texts and different layers.

Table 3 reports the average resolution of various Transformer variants. The results show that XPOS makes the position more recognizable in both 1024 (i.e., training length) and 2048 (i.e., length extrapolation). For Alibi (Press et al., 2021), the stable resolution comes from explicit decay, but it prevents the model from learning position dependency itself. In addition, we ablate BCA in 1024 and 2048. The results support that BCA helps the model distinguish positions better, achieving higher attention resolution.

4.4 Ablation Studies

4.4.1 Rotation Computation

As shown in Table 4, we discuss the necessity of the combination of vector rotation and exponential decay. XPOS without rotation means Equation (12) degenerates to $\theta_i = 0$:

$$\dot{f}_q(q,n) = \begin{pmatrix} q_1\hat{\zeta}_1^n \\ q_2\hat{\zeta}_1^n \\ \vdots \\ q_{n-1}\hat{\zeta}_{d/2}^n \\ q_n\hat{\zeta}_{d/2}^n \end{pmatrix} \dot{f}_k(k,n) = \begin{pmatrix} k_1\hat{\zeta}_1^{-n} \\ k_2\hat{\zeta}_1^{-n} \\ \vdots \\ k_{n-1}\hat{\zeta}_{d/2}^{-n} \\ k_n\hat{\zeta}_{d/2}^{-n} \end{pmatrix}$$

Length	256	512	1024	2048	4096	8192
	Interpolation			Extrapolation		
			PG22			
Transformer	38.1	33.5	30.54	132.46	1446.95	12747.41
Alibi	34.25	30.01	27.34	26.01	28.46	32.8
Roformer	33.27	29.2	26.68	68.86	235.71	458.83
LEX Transformer (Ours)	**33.18**	**29.11**	**26.59**	**25.53**	**25.07**	**24.89**
			QMSum			
Transformer	24.25	18.81	16.05	86.56	1196.92	10781.38
Alibi	22.85	17.74	15.17	13.97	15.36	18.37
Roformer	22.66	17.65	15.12	36.54	146.61	331.56
LEX Transformer (Ours)	**22.01**	**17.24**	**14.85**	**13.92**	**13.56**	**13.48**

Table 2: Results of perplexity with different lengths. The language models are trained with a length of 1024 and then evaluated on various lengths. LEX obtains better performance not only on shorter texts (i.e., interpolation) but also on longer texts (i.e., extrapolation). The red color indicates that the perplexity begins increasing compared with the shorter length. LEX is the only method that has lower perplexity along with increased evaluation length.

Length	1024 Interpolation	2048 Extrapolation
Transformer	0.87	0.28
Alibi	0.81	0.88
Roformer	0.91	0.08
LEX (Ours)	**0.98**	**1.08**
− BCA	**0.98**	0.54

Table 3: Results of resolution with different Transformer variants. Higher resolution indicates that the architecture tends to better distinguish context tokens. "BCA" is short for blockwise causal attention.

Methods	1024 Interpolation	8192 Extrapolation
LEX	**26.59**	**24.89**
w/o Rotation	37.11	34.5
$\zeta = 0$	26.68	26.16
Scalar ζ	26.85	25.1

Table 4: Ablation results on the PG22 set show that rotation of XPOS is necessary for strong performance.

Moreover, the setting of $\zeta = 0$ is RoPE (Su et al., 2021), which can be viewed as a special case of our method. Besides, we discuss the situation when ζ is a scalar instead of a vector, where we choose $\zeta = \gamma/(1+\gamma)$ as the value.

After pre-training on 1024, we evaluate the perplexity of PG22 with 1024 and 8192 lengths. Table 4 shows that simple scaling operation cannot match the performance of LEX. The vector ζ also performs better than $\zeta = 0$ and $\gamma/(1+\gamma)$. Therefore, the combination of rotation and decay means the combination of in-distribution and out-of-distribution capability in terms of length.

4.4.2 Blockwise Causal Attention

As shown in Table 5, we run the evaluation using different position embeddings (i.e., Absolute, Alibi, ROPE, and XPOS) with or without blockwise causal attention.

First, Blockwise Causal Attention works for ROPE whose perplexity will explode without that. Alibi performs well without windowed attention because its "soft window" is broader than a hard block window. However, when the sequence length increases to 8192, windowed attention outperforms vanilla attention again (also shown in Figure 1). XPOS's perplexity without BCA increases by about 1.5 in 2048, and 40 in 8192. However, with its high resolution, XPOS can recognize position with BCA's constraint.

Besides, we compare BCA with Sliding Attention (Child et al., 2019). In this experiment, we set the window size as 1024 to align with pre-training. Sliding Attention performs better as shown in the last row of Table 5 because its interaction range is broader than Block Causal Attention. The reason to use block windows instead of sliding windows is efficiency. According to (Xiong et al., 2021), the training speed of Blockwise Attention is 1.5x

Methods	2048	8192
	Extrapolation	
Absolute	132.46	12747.41
Absolute + BCA	322.73	28787.01
RoPE	68.86	458.83
RoPE + BCA	26.37	26.16
Alibi	26.01	32.8
Alibi + BCA	27.53	31.82
xPos	27.29	63.99
xPos + BCA	25.53	24.89
xPos + Sliding Window	25.33	24.61

Table 5: Results of perplexity on PG22 dataset. "BCA" is short for blockwise causal attention.

faster than using sliding windows. Therefore, LEX makes a trade-off and uses BCA in our implementation. Without losing generality, our method is also compatible with Sliding Attention and other local attention variants.

5 Related Work

5.1 Long-Sequence Transformers

Long-sequence Transformers aim to solve two key problems. First, the computation or memory consumption is not efficient enough for long sequences. Second, there is a trade-off between performance and efficiency.

One popular solution (Wang et al., 2020b; Katharopoulos et al., 2020; Choromanski et al., 2020) is linear attention, i.e., using a kernel-based or low-rank approximation to replace vanilla attention. The methods typically target efficiency while underperforming vanilla Transformers for regular length. Another strand is sparse attention (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020; Xiong et al., 2021), which usually leverages structured sparsity to reduce computation. For causal sequence modeling, the recurrent-style designs (Dai et al., 2019; Hutchins et al., 2022; Ma et al., 2022b) are also competitive.

In comparison, we focus on length extrapolation (Press et al., 2021) for language modeling, i.e., training on short texts while evaluating long texts. The training process is kept the same as vanilla Transformers. The capability of long-sequence modeling is given for free during inference. So training efficiency (which is typically expensive for large-scale language models) is not affected compared with previous work. Moreover, the performance on regular length is perfectly retained, without trade-offs for long-sequence modeling.

5.2 Position Modeling

5.2.1 Absolute Position Embedding

Absolute sinusoidal position embedding is proposed by Vaswani et al. (2017), which is the initial design of the Transformer. For each dimension, different frequencies are encoded from 2π to $10000 \times 2\pi$:

$$\begin{aligned} \mathrm{PE}_{(pos,2i)} &= \cos(pos/10000^{2i/d_{\text{model}}}) \\ \mathrm{PE}_{(pos,2i+1)} &= \sin(pos/10000^{2i/d_{\text{model}}}) \end{aligned} \tag{14}$$

where PE_{pos+k} is represented as a linear function of PE_{pos} to restore a relative-position property.

5.2.2 Relative Position Embedding

Shaw et al. (2018) propose relative position embedding as an alternative approach. Denote a_{ij} as attention weight, $\alpha_{ij} = \mathrm{softmax}(a_{ij})$, o_i as output, we have:

$$\begin{aligned} a_{ij} = \frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d}} &\Longrightarrow \frac{\boldsymbol{q}_i \cdot (\boldsymbol{k}_j + \boldsymbol{p}_{ij}^K)}{\sqrt{d}} \\ o_i = \sum_j \alpha_{ij} \boldsymbol{v}_j &\Longrightarrow \sum_j \alpha_{ij} (\boldsymbol{v}_j + \boldsymbol{p}_{ij}^V) \end{aligned} \tag{15}$$

where $\boldsymbol{p}_{ij}^K = \omega^K_{\min(i-j,k)}, \boldsymbol{p}_{ij}^V = \omega^V_{\min(i-j,k)}$, and ω^K and ω^V are learnable parameters. The clipping strategy helps length generalization but cannot distinguish the positions that are larger than k. Yang et al. (2019) and He et al. (2020) further reparameterize the relative position vectors for better performance. T5 (Raffel et al., 2020) uses a simpler strategy to encode relative position:

$$a_{ij} = \frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d}} + p_{\text{bucket}(i-j)} \tag{16}$$

where log-bucket scalars are added to attention scores. Recently, pre-defined position embedding is brought back by RoPE (Su et al., 2021). Alibi (Press et al., 2021) proposes to explicitly build an exponential decay on the attention matrix, which contributes to length extrapolation:

$$a_{ij} = \frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d}} - \mathrm{m}(i-j), \quad \mathrm{m}(\cdot) > 0 \tag{17}$$

where the values of $\mathrm{m}(\cdot)$ are manually defined. However, Alibi (Press et al., 2021)'s performance tends to be inferior to RoPE for the context whose

length is shorter than the pre-training length. In this work, we propose a theoretically derived relative position embedding xPos that optimizes the attention resolution between tokens. The xPos method not only has the nice property of length extrapolation but also achieves strong performance.

6 Conclusion

We propose LEX Transformer to accurately capture position information for Transformers. We define attention resolution as the metric of length extrapolation and design a solution to improve the modeling. Extensive experiments on language modeling show that our method achieves lower perplexity on longer sequences while training on short texts. The simplicity also makes the method a go-to augmentation for Transformer-based language models. In addition, attention resolution provides a more principled view for position modeling, which sheds light on future architecture design.

Limitations

In this work, we focus on causal language modeling. It needs additional efforts to integrate the proposed methods into bidirectional attention, such as masked language modeling (Devlin et al., 2019). Moreover, xPos introduces about 6% inference cost compared with absolute position embeddings, although it accelerates training convergence.

References

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

Tom Brown, 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.

Ta-Chung Chi, Ting-Han Fan, Peter J Ramadge, and Alexander I Rudnicky. 2022a. Kerple: Kernelized relative positional embedding for length extrapolation. *arXiv preprint arXiv:2205.09921*.

Ta-Chung Chi, Ting-Han Fan, and Alexander I Rudnicky. 2022b. Receptive field alignment enables transformer length extrapolation. *arXiv preprint arXiv:2212.10356*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *URL https://openai.com/blog/sparse-transformers*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek B Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Oliveira Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. PaLM: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Jacob Devlin, 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, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Philipp Dufter, Martin Schmitt, and Hinrich Schütze. 2022. Position information in transformers: An overview. *Computational Linguistics*, 48(3):733–763.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. 2022. Block-recurrent Transformers. In *Advances in Neural Information Processing Systems*.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, San Diego, CA.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Shuming Ma, Hongyu Wang, Shaohan Huang, Wenhui Wang, Zewen Chi, Li Dong, Alon Benhaim, Barun Patra, Vishrav Chaudhary, Xia Song, and Furu Wei. 2022a. TorchScale: Transformers at scale. *CoRR*, abs/2211.13184.

Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, and Luke Zettlemoyer. 2022b. Mega: Moving average equipped gated attention. *arXiv preprint arXiv:2209.10655*.

Ofir Press, Noah A Smith, and Mike Lewis. 2021. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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(140):1–67.

Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, et al. 2022. Scrolls: Standardized comparison over long language sequences. *arXiv preprint arXiv:2201.03533*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. 2020a. On position embeddings in bert. In *International Conference on Learning Representations*.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020b. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, et al. 2022. Image as a foreign language: BEiT pretraining for all vision and vision-language tasks. *arXiv preprint arXiv:2208.10442*.

Wenhan Xiong, Barlas Oğuz, Anchit Gupta, Xilun Chen, Diana Liskovich, Omer Levy, Wen-tau Yih, and Yashar Mehdad. 2021. Simple local attentions remain competitive for long-context tasks. *arXiv preprint arXiv:2112.07210*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, et al. 2021. Qmsum: A new benchmark for query-based multi-domain meeting summarization. *arXiv preprint arXiv:2104.05938*.

A Additional Experiments

Besides the experiments in Section 4, we run language modeling evaluation on Arxiv and NarrativeQA (Kočiskỳ et al., 2018). The results are shown in Table 6. These datasets have their shortcomings. The article length of Arxiv is usually less than 8192, and part of NarrativeQA's corpus is sampled from PG19, which is in the training dataset. Therefore, we show them in the appendix instead of the main content.

B Hyperparameters for Pre-Training

As shown in Table 7, we present the hyperparameters for pre-training. The setting keeps the same among all Transformer variants. We follow medium-size GPT3 (Brown et al., 2020), 24 layers, 1024 hidden size, 4096 FFN inner hidden size, and 16 attention heads. The number of batch tokens is 0.5M, for pre-training 1024, and the number of batch sentences is 512. We use Adam (Kingma and Ba, 2015) optimizer with $\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^{-6}$. The warmup steps are 20k, and we use 50k checkpoints for evaluation.

Length	256	512	1024	2048	4096
	Interpolation			Extrapolation	
			arXiv		
Transformer	29.74	23.6	19.59	102.09	1240.77
Alibi	26.53	21.07	17.53	**15.38**	16.88
Roformer	25.89	20.6	17.24	49.29	199.25
LEX Transformer (Ours)	**25.73**	**20.48**	**17.14**	15.81	**15.19**
			NarrativeQA		
Transformer	16.74	14.42	13.02	58.95	574.91
Alibi	15.58	13.45	12.15	11.4	12.09
Roformer	15.21	13.16	11.93	20.72	35.14
LEX Transformer (Ours)	**14.82**	**12.86**	**11.67**	**11.14**	**10.93**

Table 6: Results of perplexity with different lengths. The language models are trained with a length of 1024 and then evaluated on various lengths. LEX obtains better performance not only on shorter texts (i.e., interpolation) but also on longer texts (i.e., extrapolation). The red color indicates that the perplexity begins increasing compared with the shorter length. LEX is the only method that has lower perplexity along with increased evaluation length.

Hyperparameters	Value
Layers	24
Hidden size	1024
FFN inner hidden size	4096
Attention heads	16
Training steps	50K
Batch tokens per task	0.5M
Adam ϵ	1e-6
Adam β	(0.9, 0.98)
Learning rate	3e-4
Learning rate schedule	Polynomial
Warmup steps	20,000
Gradient clipping	2.0
Weight decay	0.01

Table 7: Hyperparameters used for language model pre-training.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
In the final part after conclusion

☐ A2. Did you discuss any potential risks of your work?
Not applicable. It is fundamental research and not tied to particular applications.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4

☑ B1. Did you cite the creators of artifacts you used?
Section 4

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. These tools and models are publicly available and free of use for research purposes.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
The use is consistent with their intended use.

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Section 4

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4

C ☑ Did you run computational experiments?

Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section4.1 and Appendix B

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section4.1 and Appendix B

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4.1

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.