Latent Positional Information is in the Self-Attention Variance of Transformer Language Models Without Positional Embeddings

Ta-Chung Chi[†]
Carnegie Mellon University

Ting-Han Fan
Princeton University

Li-Wei Chen
Carnegie Mellon University

Alexander I. Rudnicky
Carnegie Mellon University

Peter J. Ramadge
Princeton University

Abstract

The use of positional embeddings in transformer language models is widely accepted. However, recent research has called into question the necessity of such embeddings. We further extend this inquiry by demonstrating that a randomly initialized and frozen transformer language model, devoid of positional embeddings, inherently encodes strong positional information through the shrinkage of self-attention variance. To quantify this variance, we derive the underlying distribution of each step within a transformer layer. Through empirical validation using a fully pretrained model, we show that the variance shrinkage effect still persists after extensive gradient updates. Our findings serve to justify the decision to discard positional embeddings and thus facilitate more efficient pretraining of transformer language models.

1 Introduction & Related Work

Transformer models have become the backbone of natural language processing applications (Vaswani et al., 2017; Devlin et al., 2019; Radford et al., 2019). Within the transformer architecture, there are two main categories: 1) bidirectional models, such as BERT (Devlin et al., 2019), that are trained using the masked language modeling objective, and 2) (causal) language models, such as GPT (Radford et al., 2019), that are trained using the traditional language modeling objective. Both of these categories share the common feature of using positional embeddings for encoding token distance.

Whether positional embeddings are truly essential has been a subject of ongoing research. While they have been considered necessary for bidirectional transformer models (Lee et al., 2019; Luo et al., 2021; Sinha et al., 2021; Haviv et al., 2022), the situation is different for transformer language models (Irie et al., 2019; Yang et al., 2019; Tsai et al., 2019; Scao et al., 2022; Haviv et al., 2022). In transformer language models, the removal of positional embeddings results in only a marginal decline in performance, while enabling more efficient training (Haviv et al., 2022). In addition to empirical evidence, it has been proven (Bhattamishra et al., 2020) that transformer language models without positional embeddings are Turing-complete and able to model sequences akin to recurrent neural networks (Rumelhart and McClelland, 1987; Jordan, 1986). Despite this, it remains an open question where positional information is stored in the absence of positional embeddings. This motivates further investigation into individual operations within a transformer layer.

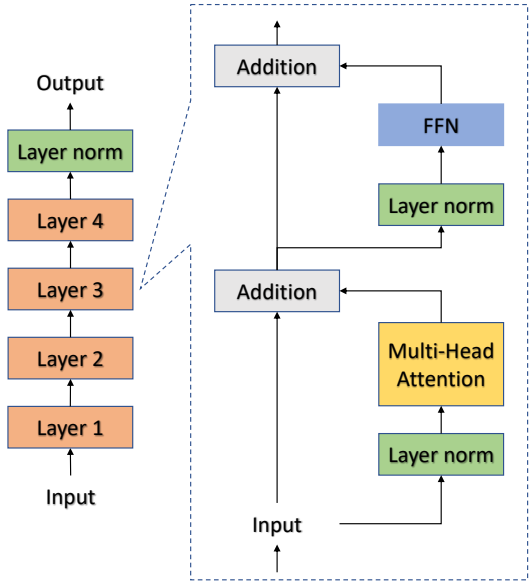

Figure 1: The architecture of a Pre-LN transformer language model. All the parameters are randomly initialized and randomly sampled input is used in this work.

The example architecture of a pre-LN (Xiong et al., 2020) multi-layer transformer language model with no positional embeddings used in this

[†]Correspondence to: tachungc@andrew.cmu.edu

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 2: Short Papers, pages 1183–1193
July 9-14, 2023

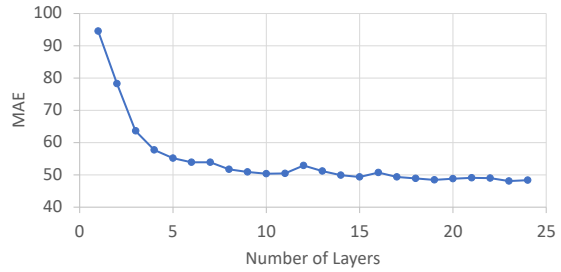

Figure 2: We plot the positions w.r.t their mean absolute error (MAE) for input sequence length $L = 512$. A naive baseline of predicting the middle point of $L = 256$ gives an MAE of 128. The numbers are the average of 5 seeds.

work is shown in Figure 1.[1] We hereinafter refer to this configuration as TLM. Our primary focus is on the multi-head attention (MHA) module of a randomly initialized TLM, as it is the only module that allows inter-token information exchange. To gain a deeper understanding, we compute the mean and variance of MHA outputs. To our surprise, we discover that the variance already encodes latent positional information, with later tokens in a sequence displaying smaller variance. This motivates us to quantify the variance by deriving the output distribution after MHA operations. Finally, through empirical validation using a fully pre-trained TLM, we confirm thatthe same variance shrinkage effect persists after extensive gradient updates.

To the best of our knowledge, we are the first to identify and quantify the latent positional information in TLMs. Our results provide theoretical insights into the removal of positional embeddings, enabling more efficient pretraining of future TLMs.

2 Probing Experiments

Given BERT and TLM (GPT) with positional embeddings removed, prior work (Haviv et al., 2022) shows that only TLM is able to maintain the same language modeling performance as its original version with positional embeddings. The discrepancy might be explained by the fact that only TLM encodes positional information within its layers, as shown by the position probing experiment in Haviv et al. (2022). Since both BERT and TLM have access to the same semantic input and the only difference is the use of causal attention masks in TLM, we hypothesize that the positional information may be attributed to the interaction between causal attention masks and the TLM architecture.

To further explore this hypothesis, we use a randomly initialized and frozen TLM to eliminate any semantic influence and focus solely on the architectural design. Additionally, to prevent the model from memorizing the order of input sequences, we do not perform embedding lookups and feed the model with randomly sampled input vectors. A trainable two-layer linear classifier with ReLU activation in between was appended to the TLM to probe the position of each token (further details can be found in Appendix B). We plot the mean absolute error (MAE) w.r.t the number of transformer layers in Figure 2. The plot indicates a randomly initialized and frozen TLM with randomly sampled input vectors inherently provides positional information, with an increase in the number of layers resulting in higher probing performance. This surprising outcome prompts further investigation into the encoding of latent positional information inside the TLM architecture.

3 Theoretical Analysis

We dissect the inner workings of a TLM by deriving the distribution of TLM operations in the hope that they elucidate where the latent positional information is stored. The derivation is made possible thanks to the usage of a randomly initialized and frozen TLM. We adopt the initialization settings in accordance with those employed in GPT (Radford et al., 2019). WLOG, our derivation is limited to the operations of the first layer in a TLM and the FFN component is omitted (justified in §3.4). The hyperparameters utilized in the simulations are: hidden dimension $d = 768$, number of attention heads $H = 12$, head dimension $d/H = 64$, sequence length $L = 512$, standard deviation for initialization $\sigma = 0.02$. All proofs of lemmas are deferred to Appendix A.

Given a sequence of randomly sampled input embeddings $\{\boldsymbol{x}_m\}_{m=1}^{L}$, where each element of $\boldsymbol{x}_m \in \mathbb{R}^d$ is sampled i.i.d from $N(0, \sigma^2)$, a TLM consists of the following operations:

3.1 Layer Normalization

For each input embedding $\boldsymbol{x}_m$, it computes the sample mean and (biased) sample variance:

$$\overline{\boldsymbol{x}}_{m,:} = \frac{\sum_{i=1}^{d} \boldsymbol{x}_{mi}}{d}, \; S(\boldsymbol{x}_{m,:}) = \frac{\sum_{i=1}^{d} (\boldsymbol{x}_{mi} - \overline{\boldsymbol{x}}_{m,:})^2}{d}$$

[1] Post-LN places layer norm at different positions. It is the configuration used in BERT (Devlin et al., 2019).

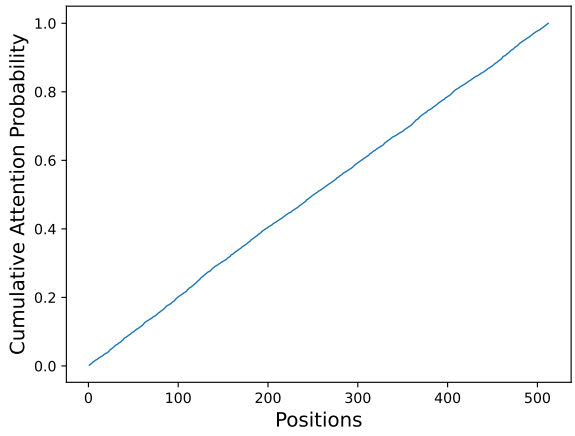

Figure 3: We plot the positions w.r.t their cumulative attention score for $L = 512$ averaged over 500 samples.

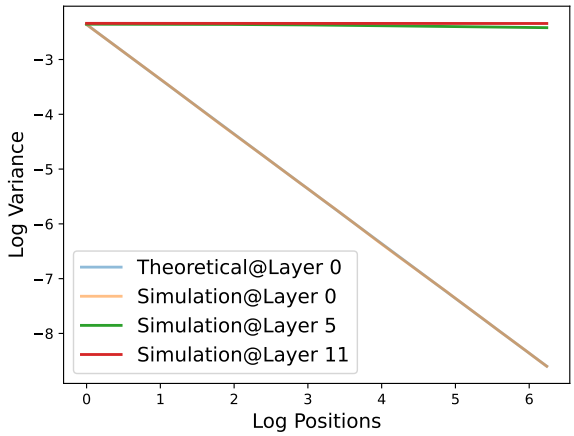

Figure 4: We plot the log positions (up to $L = 512$) w.r.t their log variance under the assumption of Property 1. The simulation aligns with the theoretical curve posited by Lemma 3 at the 0^{th} layer averaged over 500 samples.

Then each entry i of $\boldsymbol{x}_m$, denoted as $\boldsymbol{x}_{mi}$, is normalized by mean and variance to $\boldsymbol{e}_{mi}$:

$$\begin{aligned}\boldsymbol{e}_{mi} &= \frac{\boldsymbol{x}_{mi} - \overline{\boldsymbol{x}}_{m,:}}{\sqrt{S(\boldsymbol{x}_{m,:})}} * \gamma + \beta \\ &\overset{(*)}{\approx} \frac{\boldsymbol{x}_{mi} - \mathbb{E}[\boldsymbol{x}_{mi}]}{\sqrt{\mathbb{V}[\boldsymbol{x}_{mi}]}} \sim N(0,1),\end{aligned}$$

where $\mathbb{V}[\boldsymbol{x}]$ denotes the variance of $\boldsymbol{x}$. Since the initialization scheme sets $\gamma = 1$ and $\beta = 0$, $(*)$ holds with sufficiently large d by the Law of large numbers and the continuous mapping theorem.

3.2 Self Attention

Each attention head computes query, key, and value vectors in $\mathbb{R}^{\frac{d}{H}}$:

$$\boldsymbol{q}_m = \boldsymbol{W}_q \boldsymbol{e}_m, \quad \boldsymbol{k}_m = \boldsymbol{W}_k \boldsymbol{e}_m, \quad \boldsymbol{v}_m = \boldsymbol{W}_v \boldsymbol{e}_m,$$

where $\boldsymbol{W}_q$, $\boldsymbol{W}_k$, $\boldsymbol{W}_v \in \mathbb{R}^{\frac{d}{H} \times d}$ are matrices with each element sampled i.i.d from $N(0, \sigma^2)$.

To be precise, most matrices ($\boldsymbol{W}_q^{(h)}$, $\boldsymbol{W}_k^{(h)}$, $\boldsymbol{W}_v^{(h)}$), vectors ($\boldsymbol{q}_m^{(h)}$, $\boldsymbol{k}_m^{(h)}$, $\boldsymbol{v}_m^{(h)}$), and scalars ($l_{mn}^{(h)}$, $a_{mn}^{(h)}$) are associated with a head number h. For notation simplicity, we only show the dependency on h when we need it.

Lemma 1. *$\boldsymbol{q}_m$, $\boldsymbol{k}_m$, and $\boldsymbol{v}_m$ have zero mean and $(d\sigma^2) \cdot I$ covariance matrix.*

The resulting vectors are processed by the self-attention module for pre-Softmax logits:

$$l_{mn} = \begin{cases} \langle \boldsymbol{q}_m, \boldsymbol{k}_n \rangle, & \text{if } m \geq n \\ -\inf, & \text{otherwise} \end{cases}$$

followed by the scaled softmax normalization:

$$a_{mn} = \frac{\exp\left(l_{mn}/\sqrt{d/H}\right)}{\sum_{i=1}^{L} \exp\left(l_{mi}/\sqrt{d/H}\right)}$$

Lemma 2. *l_{mn} has zero mean and $\frac{d^3\sigma^4}{H^2}$ variance. $l_{mn}/\sqrt{d/H}$ has $\frac{d^2\sigma^4}{H}$ variance.*

The numerical variance of $l_{mn}/\sqrt{d/H}$ in our case is $\frac{768^2 \cdot 0.02^4}{12} \approx 0.0079$. Lemma 2 suggests the following approximation:

Property 1. *When $\sigma^4 \ll \frac{H}{d^2}$, $l_{m,:}$ has small variance, making the attention weights $a_{m,:}$ almost evenly distributed among all positions.*[2]

In Figure 3, we verify Property 1 by showing that $\boldsymbol{a}_{mn}$ is almost evenly distributed in simulation.

Observe that the output vector $\boldsymbol{o}_m$ at position m is:

$$\boldsymbol{o}_m = \boldsymbol{W}_o \left(\oplus_{h=1}^{H} \sum_{n=1}^{L} a_{mn}^{(h)} \boldsymbol{v}_n^{(h)} \right),$$

where $\oplus$ denotes the concatenation of vectors from all H attention heads. Assume that Property 1 is valid and that $\boldsymbol{W}_o \in \mathbb{R}^{d \times d}$ has elements i.i.d sampled from $N(0, \sigma^2)$, we derive the distribution of $\boldsymbol{o}_m$ below.

Lemma 3. *$\boldsymbol{o}_m$ has zero mean and $\frac{d^2\sigma^4}{m} I$ covariance matrix.*

[2]This approximation was also used in Xiong et al. (2020) except that they made a stronger assumption that $\boldsymbol{W}_q$ and $\boldsymbol{W}_k$ have to be initialized as zero matrices.

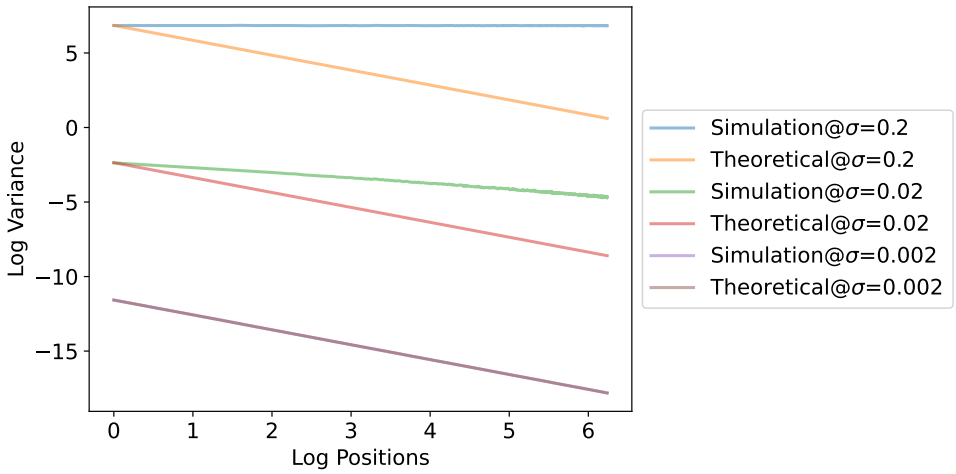

Figure 5: We vary the value of σ and show its effect at the 0^{th} layer. As we can see, a smaller value of σ brings Lemma 3 into alignment with the corresponding simulation more closely. Note that the two lines overlap completely when $\sigma = 0.002$. Average of 500 samples.

Figure 4 is a simulation that verifies Lemma 3 under the assumption of Property 1. We can see that *the variance of $\boldsymbol{o}_m$ already encodes the positional information m.*

3.3 Residual Connection

As denoted by the *Addition* block of Figure 1, the residual connection sets the output as $\boldsymbol{y}_m = \boldsymbol{x}_m + \boldsymbol{o}_m$. It allows the model to pass the first MHA output to later MHA modules as well as the final classifier. As the positional information has been passed by the residual connection, we omit the FFN part in our analysis.

3.4 The Final Layer Normalization

Layer normalization is an operation that might eliminate the positional information derived in Lemma 3, which happens before the MHA modules and position classifier. As mentioned in §3.1, $\text{LN}(\boldsymbol{y}_m)$ gives:

$$\boldsymbol{y}'_{mi} \approx \frac{\boldsymbol{y}_{mi} - \mathbb{E}[\boldsymbol{y}_{mi}]}{\sqrt{\mathbb{V}[\boldsymbol{y}_{mi}]}} \approx \frac{\boldsymbol{x}_{mi} + \boldsymbol{W}_o \boldsymbol{W}_v \frac{\sum_n^m \boldsymbol{e}_{ni}}{m}}{\sqrt{\sigma^2 + \frac{d^2\sigma^4}{m}}},$$

$$\mathbb{E}[\boldsymbol{y}_{mi}] = 0,\ \mathbb{V}[\boldsymbol{y}_{mi}] = \mathbb{V}[\boldsymbol{x}_{mi}] + \mathbb{V}[\boldsymbol{o}_{mi}] = \sigma^2 + \frac{d^2\sigma^4}{m}$$

Lemma 4. *The variance of the j-th dimension of $\boldsymbol{y}_m$ is:*

$$\frac{m\sigma^2 + \sum_i (\boldsymbol{W}_{o,j:} \boldsymbol{W}_{v,:i})^2}{m\sigma^2 + d^2\sigma^4},$$

where $\boldsymbol{W}_{o,j:} \in \mathbb{R}^{1\times d}$ is the j-th row of $\boldsymbol{W}_o$. $\boldsymbol{W}_{v,:i} \in \mathbb{R}^{d\times 1}$ is the i-th column of $\boldsymbol{W}_v$. As long as $\sum_i (\boldsymbol{W}_{o,j:} \boldsymbol{W}_{v,:i})^2 \neq d^2\sigma^4$, the classifier should be able to exploit the discrepancy to derive m.

Readers might wonder why $\boldsymbol{W}_{o,j:}$ and $\boldsymbol{W}_{v,:i}$ in the numerator cannot be treated as random variables. The reason is that we only focus on one dimension (j-th) at a time. This means we cannot use the law of large numbers to approximate the sample variance of $\boldsymbol{y}_{mj}$ as we did for the denominator.

3.5 Relaxing the Assumptions

We discuss possible relaxation of the assumptions used in §3.2.

What if Property 1 does not hold? Or equivalently, $\sigma^4 \not\ll \frac{H}{d^2}$. This prompts us to vary the value of σ. In Figure 5, we see that smaller σ better aligns Lemma 3 with the simulations, which is unsurprising as Lemma 3 assumes small σ. Even when σ is not too small (i.e., $\sigma = 0.2, 0.02$), the variance still encodes the positional information as the variance of $\boldsymbol{o}_m$ is negatively correlated with its position m.

Other Initialization Schemes So far we assume the weight matrices ($\boldsymbol{W}_q$, $\boldsymbol{W}_k$, $\boldsymbol{W}_v$, $\boldsymbol{W}_o$) are initialized i.i.d from $N(0, \sigma^2)$. However, we can relax the assumption to i.i.d. samples from a distribution with zero mean and finite variance. This is because the proof in Appendix A calculates the covariance. The variance calculation relies on $\mathbb{E}[\boldsymbol{r}_i \boldsymbol{r}_i^\top] = \sigma^2 I$ where $\boldsymbol{r}_i^\top$ is the i-th row vector of a weight matrix. This property holds for any distribution with zero

mean and σ^2 variance.

4 Discussions

Why are the positions of later tokens in a sequence harder to be predicted in Figure 3 of Haviv et al. (2022)? Lemma 3 states the variance is inversely proportional to the position m, so the variance of later tokens (large m) plateaus, resulting in a harder numerical optimization problem. This also suggests a potential downside of removing positional embeddings: It might be challenging for the model to infer positional information of the later tokens in extremely long input sequences.

Why do lower layers (closer to input) give worse probing performances in both Figure 2 and Haviv et al. (2022)? This can be explained by Figure 4. Most of the positions at the 0^{th} layer have tiny variance ($\exp(-10) = 4.5e^{-5}$), which again poses a difficult numerical optimization problem.

Why does BERT fail to converge without positional embeddings? In a BERT model (Devlin et al., 2019), each token has access to all the other tokens, making the variance at all positions $\frac{d^2\sigma^4}{L}$. Therefore, a BERT model cannot utilize variance differences as its positional indicator.

5 Post-Training Results

Our derivations only apply to the initial stage where the TLM and input embeddings are randomly initialized, which may not hold true after gradient updates. It is essential to verify the existence of variance properties and lemmas on a fully pre-trained TLM on OpenWebText2 (details in Appendix C).

We expect that the properties of lower layers of a pre-trained TLM should align more closely with the theoretical results for two reasons: 1) There are more steps between the lower layers and the final language modeling loss, resulting in smaller gradients and thereby fewer parameter updates, and 2) Lower layers typically encode more low-level information dependent on positional information (Vulić et al., 2020; de Vries et al., 2020). Figures 6 and 7 demonstrate that the 0^{th} (lowest) layer exhibits highly similar cumulative attention probability and decay-with-position variance as the theoretical results. In contrast, higher layers deviate from the analyses in § 3. We posit that the model learns to rely more heavily on semantic rather than positional information. This also explains why predicting positions using outputs of higher transformer layers is more challenging as demonstrated in Figure 2 of Haviv et al. (2022).

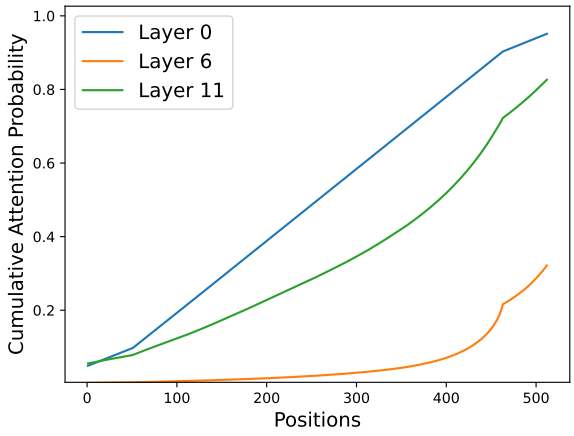

Figure 6: We plot the positions w.r.t their cumulative attention probability for $L = 512$ of a pre-trained TLM. We average over all heads in a layer and 500 samples.

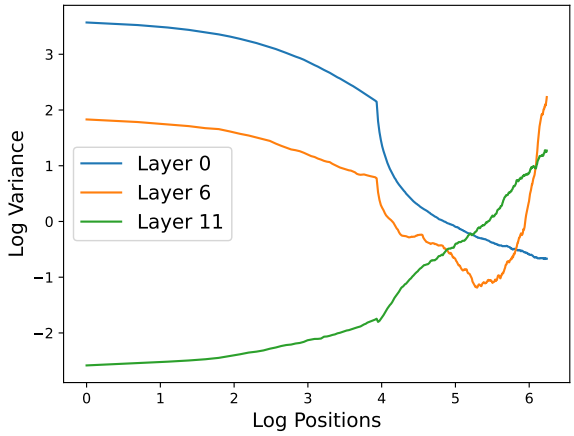

Figure 7: We plot the log positions w.r.t their log variance for $L = 512$ of a pre-trained TLM. We average over 500 samples.

6 Conclusion

We mathematically analyzed a randomly initialized transformer language model without positional embeddings. We showed that the variance of the self-attention output decreases as the position increases, which serves as an indicator for positional information. We validated that, after extensive gradient updates, the low layers of a pretrained language model still exhibit highly similar variance reduction behaviors. Our results pave the way for the pretraining of more efficient and positional embedding-free transformer language models.

Limitations

The limitations of this work mostly come from our assumptions: 1) A randomly initialized and frozen TLM, and 2) Input tokens are all different and randomly sampled. These two assumptions obviously do not hold true for human languages and pre-trained TLMs. Therefore, we attempted to empirically verify the existence of lemmas and properties on a pre-trained TLM without positional embeddings in §5.

That being said, several methods could be attempted to remove these assumptions. Firstly, we can analyze the training dynamics of a TLM to shed light on the model parameter distribution after pre-training. Secondly, Zipf's law or a simple n-gram language model could be used to quantify the degree of input token duplication in human languages. This might give us a more accurate estimate of the variance at different positions. We leave these ideas as future work.

Ethics Statement

Our work provides a deeper understanding of why a transformer language model can still perform well without positional embeddings, potentially enabling the application of developing future transformers that are greener and more cost-efficient. Inappropriate usage of our technique might have negative societal impacts though. These include the ethical challenges of improper text generation and privacy issues inherent in the data collection process. These implications apply to any natural language processing research and are not unique to this specific work.

Acknowledgment

The authors acknowledge the support from Boeing (2019-STU-PA-259), Amazon (CC ADV 00474341 2021 TR), NSF MRI Award 1919452, and Princeton Research Computing.

References

Alex Andonian, Quentin Anthony, Stella Biderman, Sid Black, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Michael Pieler, Shivanshu Purohit, Tri Songz, Wang Phil, and Samuel Weinbach. 2021. GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch.

Satwik Bhattamishra, Arkil Patel, and Navin Goyal. 2020. On the computational power of transformers and its implications in sequence modeling. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 455–475, Online. Association for Computational Linguistics.

Wietse de Vries, Andreas van Cranenburgh, and Malvina Nissim. 2020. What's so special about BERT's layers? a closer look at the NLP pipeline in monolingual and multilingual models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4339–4350, Online. Association for Computational Linguistics.

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.

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

Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. 2022. Transformer language models without positional encodings still learn positional information. *arXiv preprint arXiv:2203.16634*.

Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019. Language modeling with deep transformers. In *INTERSPEECH*.

M I Jordan. 1986. Serial order: a parallel distributed processing approach. technical report, june 1985-march 1986.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. Cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753. PMLR.

Ziyang Luo, Artur Kulmizev, and Xiaoxi Mao. 2021. Positional artefacts propagate through masked language model embeddings. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5312–5327, Online. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

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.

David E. Rumelhart and James L. McClelland. 1987. *Learning Internal Representations by Error Propagation*, pages 318–362.

Teven Le Scao, Thomas Wang, Daniel Hesslow, Lucile Saulnier, Stas Bekman, M Saiful Bari, Stella Biderman, Hady Elsahar, Jason Phang, Ofir Press, Colin Raffel, Victor Sanh, Sheng Shen, Lintang Sutawika, Jaesung Tae, Zheng Xin Yong, Julien Launay, and Iz Beltagy. 2022. What language model to train if you have one million GPU hours? In *Challenges & Perspectives in Creating Large Language Models*.

Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. 2021. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2888–2913, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2019. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. 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 4344–4353, Hong Kong, China. Association for Computational Linguistics.

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.

Ivan Vulić, Edoardo Maria Ponti, Robert Litschko, Goran Glavaš, and Anna Korhonen. 2020. Probing pretrained language models for lexical semantics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7222–7240, Online. Association for Computational Linguistics.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*.

Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Zhaopeng Tu. 2019. Assessing the ability of self-attention networks to learn word order. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3635–3644, Florence, Italy. Association for Computational Linguistics.

A Proofs

The proof of Lemma 1 and 2 are head-dependent while that of Lemma 3 is head-independent. For notation simplicity, at Lemma 1 and 2, we drop the head dependency on matrices ($\boldsymbol{W}_q^{(h)}$, $\boldsymbol{W}_k^{(h)}$, $\boldsymbol{W}_v^{(h)}$), vectors ($\boldsymbol{q}_m^{(h)}$, $\boldsymbol{k}_m^{(h)}$, $\boldsymbol{v}_m^{(h)}$), and scalars ($l_{mn}^{(h)}$, $a_{mn}^{(h)}$).

Proof of Lemma 1 Here, we use $\boldsymbol{r}_i^\top$ to denote the i-th row vector of $\boldsymbol{W}_v$.

$$
\begin{aligned}
\text{cov}(\boldsymbol{v}_m, \boldsymbol{v}_n) &= \mathbb{E}[\boldsymbol{v}_m \boldsymbol{v}_n^\top] \\
&= \mathbb{E}[\boldsymbol{W}_v \boldsymbol{e}_m \boldsymbol{e}_n^\top \boldsymbol{W}_v^\top] \\
&= \mathbb{E}\left[\begin{bmatrix} \boldsymbol{r}_1^\top \boldsymbol{e}_m \\ \vdots \\ \boldsymbol{r}_{\frac{d}{H}}^\top \boldsymbol{e}_m \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_n^\top \boldsymbol{r}_1 & \dots & \boldsymbol{e}_n^\top \boldsymbol{r}_{\frac{d}{H}} \end{bmatrix}\right] \\
&= \left[\mathbb{E}[\boldsymbol{r}_i^\top \boldsymbol{e}_m \boldsymbol{e}_n^\top \boldsymbol{r}_j]\right]_{i,j=1}^{\frac{d}{H}} \\
&= \left[\mathbb{E}[\text{Tr}(\boldsymbol{r}_j \boldsymbol{r}_i^\top \boldsymbol{e}_m \boldsymbol{e}_n^\top)]\right]_{i,j=1}^{\frac{d}{H}} \\
&= \left[\text{Tr}(\mathbb{E}[\boldsymbol{r}_j \boldsymbol{r}_i^\top] \mathbb{E}[\boldsymbol{e}_m \boldsymbol{e}_n^\top])\right]_{i,j=1}^{\frac{d}{H}} \\
&\overset{(*)}{=} \left[\text{Tr}((\mathbb{1}_{i=j}\sigma^2) \cdot I_d \cdot \mathbb{1}_{m=n} \cdot I_d)\right]_{i,j=1}^{\frac{d}{H}} \\
&= \left[\mathbb{1}_{i=j} \mathbb{1}_{m=n} d\sigma^2\right]_{i,j=1}^{\frac{d}{H}} \\
&= (\mathbb{1}_{m=n} d\sigma^2) \cdot I_{d/H}
\end{aligned}
$$

$(*)$ holds because $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$ are independent when $i \neq j$ (similarly for $\boldsymbol{e}_m$ and $\boldsymbol{e}_n$) and the covariance of a Gaussian random vector is an identity matrix. I_d and $I_{d/H}$ denote $d \times d$ and $\frac{d}{H} \times \frac{d}{H}$ identity matrices.

Proof of Lemma 2 Here, we use $\boldsymbol{r}_i^\top$ to denote the i-th row vector of $\boldsymbol{W}_q$ and $\boldsymbol{W}_k$.

$$
\begin{aligned}
&\text{cov}(l_{mn}, l_{mp}) \\
&= \mathbb{E}[(\boldsymbol{e}_m^\top \boldsymbol{W}_q^\top \boldsymbol{W}_k \boldsymbol{e}_n)(\boldsymbol{e}_m^\top \boldsymbol{W}_q^\top \boldsymbol{W}_k \boldsymbol{e}_p)^\top] \\
&= \mathbb{E}[\text{Tr}(\boldsymbol{e}_m^\top \boldsymbol{W}_q^\top \boldsymbol{W}_k \boldsymbol{e}_n \boldsymbol{e}_p^\top \boldsymbol{W}_k^\top \boldsymbol{W}_q \boldsymbol{e}_m)] \\
&= \mathbb{E}[\text{Tr}(\boldsymbol{e}_m \boldsymbol{e}_m^\top \boldsymbol{W}_q^\top \boldsymbol{W}_k \boldsymbol{e}_n \boldsymbol{e}_p^\top \boldsymbol{W}_k^\top \boldsymbol{W}_q)] \\
&= \text{Tr}(\mathbb{E}[\boldsymbol{e}_m \boldsymbol{e}_m^\top] \mathbb{E}[\boldsymbol{W}_q^\top \boldsymbol{W}_k \boldsymbol{e}_n \boldsymbol{e}_p^\top \boldsymbol{W}_k^\top \boldsymbol{W}_q]) \\
&= \mathbb{E}[\text{Tr}(\boldsymbol{e}_n \boldsymbol{e}_p^\top \boldsymbol{W}_k^\top \boldsymbol{W}_q \boldsymbol{W}_q^\top \boldsymbol{W}_k)] \\
&= \text{Tr}(\mathbb{E}[\boldsymbol{e}_n \boldsymbol{e}_p^\top] \mathbb{E}[\boldsymbol{W}_k^\top \boldsymbol{W}_q \boldsymbol{W}_q^\top \boldsymbol{W}_k)]) \\
&= (\mathbb{1}_{n=p}) \text{Tr}(\mathbb{E}[\boldsymbol{W}_q \boldsymbol{W}_q^\top] \mathbb{E}[\boldsymbol{W}_k \boldsymbol{W}_k^\top]) \\
&\overset{(*)}{=} (\mathbb{1}_{n=p}) \text{Tr}((\frac{d}{H}\sigma^2 \cdot I)(\frac{d}{H}\sigma^2 \cdot I)) \\
&= (\mathbb{1}_{n=p}) \frac{d^3\sigma^4}{H^2}
\end{aligned}
$$

$(*)$ holds since:

$$
\begin{aligned}
\mathbb{E}[\boldsymbol{W}_q \boldsymbol{W}_q^\top] &= \mathbb{E}\left[\begin{bmatrix} \boldsymbol{r}_1^\top \\ \vdots \\ \boldsymbol{r}_{\frac{d}{H}}^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_1 & \dots & \boldsymbol{r}_{\frac{d}{H}} \end{bmatrix}\right] \\
&= \left[\mathbb{E}[\boldsymbol{r}_i^\top \boldsymbol{r}_j]\right]_{i,j=1}^{\frac{d}{H}} = \frac{d}{H}\sigma^2 \cdot I
\end{aligned}
$$

Proof of Lemma 3 Because $\boldsymbol{W}_o \in \mathbb{R}^{d \times d}$ is applied on a concatenation of vectors at all heads, we take $\boldsymbol{v}_i = \oplus_{h=1}^H \boldsymbol{v}_i^{(h)}$. $\boldsymbol{v}_i$ here is head-independent while $\boldsymbol{v}_i$ at Lemma 1 is head-dependent. Here, we use $\boldsymbol{r}_i^\top$ to denote the i-th row vector of $\boldsymbol{W}_o$.

$$
\begin{aligned}
&\text{cov}(\boldsymbol{o}_m, \boldsymbol{o}_m) \\
&\overset{\text{Property 1}}{\approx} \mathbb{E}\left[\boldsymbol{W}_o \frac{\sum_{i=1}^m \boldsymbol{v}_i}{m} \frac{\sum_{j=1}^m \boldsymbol{v}_j^\top}{m} \boldsymbol{W}_o^\top\right] \\
&= \frac{1}{m^2} \sum_{i,j=1}^m \mathbb{E}[\boldsymbol{W}_o \boldsymbol{v}_i \boldsymbol{v}_j^\top \boldsymbol{W}_o^\top] \\
&= \frac{1}{m^2} \sum_{i,j=1}^m \mathbb{E}\left[\begin{bmatrix} \boldsymbol{r}_1^\top \boldsymbol{v}_i \\ \vdots \\ \boldsymbol{r}_d^\top \boldsymbol{v}_i \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_j^\top \boldsymbol{r}_1 & \dots & \boldsymbol{v}_j^\top \boldsymbol{r}_d \end{bmatrix}\right] \\
&= \frac{1}{m^2} \sum_{i,j=1}^m \left[\mathbb{E}[\boldsymbol{r}_k^\top \boldsymbol{v}_i \boldsymbol{v}_j^\top \boldsymbol{r}_l]\right]_{k,l=1}^d \\
&= \frac{1}{m^2} \sum_{i,j=1}^m \left[\mathbb{E}[\text{Tr}(\boldsymbol{r}_l \boldsymbol{r}_k^\top \boldsymbol{v}_i \boldsymbol{v}_j^\top)]\right]_{k,l=1}^d \\
&= \frac{1}{m^2} \sum_{i,j=1}^m \left[\text{Tr}(\mathbb{E}[\boldsymbol{r}_l \boldsymbol{r}_k^\top] \mathbb{E}[\boldsymbol{v}_i \boldsymbol{v}_j^\top])\right]_{k,l=1}^d \\
&\overset{(*)}{=} \frac{1}{m^2} \sum_{i,j=1}^m \big[\text{Tr}((\mathbb{1}_{k=l}\sigma^2) \cdot I \\
&\qquad \cdot (\mathbb{1}_{i=j} d\sigma^2) \cdot I)\big]_{k,l=1}^d \\
&= \frac{d^2\sigma^4}{m} I
\end{aligned}
$$

$(*)$ follows from Lemma 1: because $\text{cov}(\boldsymbol{v}_i^{(h)}, \boldsymbol{v}_j^{(h)}) = (\mathbb{1}_{i=j} d\sigma^2) \cdot I_{d/H}$, a concatenation for all $h \in H$ gives $\mathbb{E}[\boldsymbol{v}_i \boldsymbol{v}_j^\top] = (\mathbb{1}_{i=j} d\sigma^2) \cdot I_d$.

B Probing Experiment Details

We train a randomly initialized and frozen TLM with 12 layers, $d = 768$, $H = 12$, $L = 512$, and $\sigma = 0.02$. We use the Adam optimizer (Kingma and Ba, 2014) with learning rate $1e-3$ and 5000 gradient updates. The batch size is set to 32. We implement our model using PyTorch (Paszke et al., 2019).

# Layers	Hidden Size	# Attention Heads	Train Seq. Len.	# Trainable Params.
12	64	12	512	162M
Optimizer	Batch Size	Train Steps	Precision	Dataset
Adam (lr 6e-4)	32	50,000	bfloat16	OpenWebText2

Table 1: **Pre-trained Model Configurations.**

C Pre-trained Transformer Language Model Details

We use the gpt-neox library (Andonian et al., 2021) to train a TLM with no positional embeddings. Detailed hyperparameters are listed in Table 1. The pretraining takes 5 hours on one NVIDIA A100-40GB.

D Scientific Artifacts

We use the gpt-neox library (Andonian et al., 2021) under Apache-2.0 license. OpenWebText2 (Gao et al., 2020) is released by the authors of gpt-neox. The codebase and dataset are publicly released for research purposes. The steps taken to protect privacy and anonymization are discussed in Section 6 and 7 of Gao et al. (2020). The distribution and statistics of OpenWebext2 are also discussed in Gao et al. (2020).

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations section

☑ A2. Did you discuss any potential risks of your work?
Ethics Statement

☑ 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 2 and 5

☑ B1. Did you cite the creators of artifacts you used?
appendix D

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
appendix D

☑ 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)?
appendix D

☑ 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?
appendix D

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

☑ 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.
appendix D

C ☑ Did you run computational experiments?

section 2, 3, and 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?
appendix C

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?
appendix C

☑ 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?
In figure captions scattered across all sections

☐ 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.)?
Not applicable. Left blank.

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.