Value-aware Approximate Attention

Ankit Gupta
Tel Aviv University
`ankitgupta.iitkanpur@gmail.com`

Jonathan Berant
Tel Aviv University,
Allen Institute for AI
`joberant@cs.tau.ac.il`

Abstract

Following the success of dot-product attention in Transformers, numerous approximations have been recently proposed to address its quadratic complexity with respect to the input length. However, all approximations thus far have ignored the contribution of the *value vectors* to the quality of approximation. In this work, we argue that research efforts should be directed towards approximating the true output of the attention sub-layer, which includes the value vectors. We propose a value-aware objective, and show theoretically and empirically that an optimal approximation of a value-aware objective substantially outperforms an optimal approximation that ignores values, in the context of language modeling. Moreover, we show that the choice of kernel function for computing attention similarity can substantially affect the quality of sparse approximations, where kernel functions that are less skewed are more affected by the value vectors.

1 Introduction

The Transformer architecture (Vaswani et al., 2017) has been widely successful in a wide range of natural language processing tasks, including machine translation (Edunov et al., 2018), language modeling (Roy et al., 2020), question-answering (Karpukhin et al., 2020), and many more. Transformers pre-trained on large amounts of text with a language modeling (LM) objective have become the standard in NLP, exhibiting surprising amounts of linguistic and world knowledge (Peters et al., 2018; Devlin et al., 2019; Petroni et al., 2019; Hewitt and Manning, 2019; Roberts et al., 2020).

The contextualizing component of the Transformer is the attention layer where all positions in an input sequence of length L aggregate information from the entire sequence in parallel. At its core, given L query, key and value vec-

tors, the *dot-product attention* function outputs[1] $\text{softmax}(QK^\top)V$ where the softmax function is applied row-wise on the matrix $QK^\top \in \mathbb{R}^{L \times L}$, consisting of similarity scores of the query-key pairs. Unfortunately, computing $\Omega(L \cdot L)$ similarity scores is prohibitive for long sequences.

To alleviate this, past work proposed to compute an approximation of $\text{softmax}(QK^\top)$. One major line of research focused on *sparse attention* variants, where only a few similarity scores are computed per position, and the rest are ignored. Methods differ by which query-key pairs are selected (Child et al., 2019; Ye et al., 2019; Qiu et al., 2019; Roy et al., 2020; Kitaev et al., 2020; Beltagy et al., 2020; Gupta and Berant, 2020; Vyas et al., 2020). A second line of research explored *dense* variants (Katharopoulos et al., 2020; Wang et al., 2020; Bello, 2021; Tay et al., 2020a) (cf. (Tay et al., 2020b) for a survey). E.g., instead of computing the attention scores exactly for only a few query-key pairs, (Choromanski et al., 2020) compute an approximation of scores for all pairs.

In this work, we point to a lacuna in current research on efficient Transformers. While recent work focused on approximating the attention scores $\text{softmax}(QK^\top)$, the true target of approximation should be the output of the attention sub-layer, namely $H = \text{softmax}(QK^\top)V$, which also includes the value vectors, V. We show that ignoring value vectors leads to unwarranted consequences both theoretically and empirically.

To demonstrate the importance of value-aware approximation, we analyze *optimal sparse attention*, that is, the case where, in hindsight, the model computes dot product similarity only with the most similar key vectors, while still ignoring the value vectors. We show that in the popular masked language modeling (MLM) setup, optimal sparse attention dramatically *under-performs* compared to

---

[1] Usually, the term is $\text{softmax}(QK^\top/\sqrt{d})V$ but $\sqrt{d}$ can be dropped via scaling of queries.

an optimal approximation of the true output of the attention sub-layer, H, leading to an error increase of 8-20 points. Next, by theoretically focusing on the case where queries compute similarity to the *single* most similar key vector, we show that approximating $\mathrm{softmax}(QK^\top)$ is equivalent to approximating H when the value vectors V satisfy strong orthogonality and norm constraints. Conversely, when they do not, ignoring V can lead to an unbounded approximation error.

Second, we discuss the kernel-based view of attention, where efficiency is gained by replacing the exponential kernel (corresponding to $\mathrm{softmax}$) with other kernel functions (Katharopoulos et al., 2020). We theoretically show that while in the exponential kernel case (corresponding to $\mathrm{softmax}$), the effect of the norm of the value vectors is potentially small, switching to other kernels can dramatically increase the importance of the value vectors. We empirically test this by comparing optimal sparse attention given different kernel functions, and see that indeed approximation quality decreases when replacing the exponential kernel,

To conclude, we theoretically and empirically show that approximating the attention score matrix alone is insufficient, and propose that the research community should instead approximate the true output of the sub-attention layer, which importantly includes value vectors. Our code and trained models are available at https://github.com/ag1988/value_aware_attn.

2 Background

We review the kernel-based view of attention (Tsai et al., 2019), which will be instructive in §3.

Generalized Attention Let $\kappa(x,y) = \langle\Phi(x),\Phi(y)\rangle \geq 0$ be a kernel function with feature map $\Phi : \mathbb{R}^d \mapsto \mathcal{H}$ for some implicit reproducing kernel Hilbert space (RKHS) $\mathcal{H}$. Given a query vector q, keys $k_1,\ldots,k_L$, values $v_1,\ldots,v_L$, all in $\mathbb{R}^d$:

$$\mathrm{att}_\kappa(q,k_1,\ldots,v_1,\ldots) = \frac{\sum_{i=1}^L \kappa(q,k_i)v_i}{\sum_{i=1}^L \kappa(q,k_i)}, \quad (1)$$

where the normalization induces a probability distribution $\boldsymbol{\alpha}$ over the value vectors with $\alpha_i = \kappa(q,k_i)/\sum_i \kappa(q,k_i)$. The most popular use case is the exponential kernel $\kappa(x,y) = \exp(\langle x,y\rangle)$, referred to as dot-product attention in Transformers. Some other examples include the degree-2 *polynomial* kernel $\kappa(x,y) = \langle x,y\rangle^2$ and the

recently proposed *elu* kernel $\langle\Phi(x),\Phi(y)\rangle$ with $\Phi(\cdot) = 1 + \mathrm{ELU}(\cdot)$ (Katharopoulos et al., 2020).

Given $L \gg d$ queries, the attention function (Eq. 1) requires computing $L \cdot L$ similarity scores for the query-key pairs, which is prohibitive for long sequences. *Sparse attention* variants relax this requirement and compute only a few similarity scores, ignoring the rest:

$$\mathrm{att}_{\kappa,S} = \frac{\sum_{i\in S} \kappa(q,k_i)v_i}{\sum_{i\in S} \kappa(q,k_i)}, \quad (2)$$

for some $S \subseteq \{1,\ldots,L\}, |S| \ll L$. Methods differ in how S is determined given the queries and keys, and include use of locality bias (Beltagy et al., 2020), global memory (Gupta and Berant, 2020), and LSH hashing (Kitaev et al., 2020), among others. Conversely, instead of exactly computing the attention scores only on a few query-key pairs, *dense* variants compute an approximation of the true kernel values for all pairs. Such methods output $\sum_i \beta_i \cdot v_i$ for some approximation $\boldsymbol{\beta}$ of the true attention distribution $\boldsymbol{\alpha}$ (Choromanski et al., 2020; Peng et al., 2021).

3 Optimal Sparse Attention

Prior methods for approximating attention have ignored the contribution of the values vectors V. As the true output of the attention sub-layer also depends on V, a natural question is whether it is possible to design better approximation methods by incorporating V, and if so, how much improvement is even possible?

To answer this, we focus on sparse attention, and analyze the difference between an oracle sparse approximation that considers the value vectors, and an oracle approximation that does not. That is, we look at the difference between the two approximations from the perspective of *expressivity*, ignoring any memory and computational constraints. We denote an optimal value-aware approximation that uses r key vectors per query by *optimal-v-aware-r*, and an optimal approximation that ignores value vectors by *optimal-v-oblivious-r*. We define *optimal-v-oblivious-r* as the output of Eq. 2 in which S is selected to be the r indices with the highest attention scores α_i's. This is a natural baseline since this is what current sparse methods are trying to emulate. We now explicitly derive and analyze the value-aware objective.

Value-aware objective Let $o = \sum_{i=1}^L \alpha_i v_i$ be a convex combination of $v_1,\ldots,v_L \in \mathbb{R}^d$, corre-

sponding to the true output of the attention sub-layer. Let $C_r = \{\sum_{i=1}^{L} \beta_i v_i : \forall i\ \beta_i \geq 0, \sum_i \beta_i = 1, |\{\beta_i : \beta_i > 0\}| \leq r\}$ denote the set of points in the polytope of v_i's that can be expressed as a convex combination of at most r value vectors v_i. The goal of value-aware approximation is to solve for the point in the constrained region C_r closest to the true output o, i.e. $\operatorname{argmin}_{\tilde{o} \in C_r} ||o - \tilde{o}||^2$. As mentioned, this solution is termed *optimal-v-aware-r*.

We consider two extreme cases of r: $r = 1$ and $r \geq d + 1$. For $r \geq d + 1$, the Carathéodory Theorem (Bárány and Karasev, 2012) states that $o = \sum_i \alpha_i v_i$ can be expressed as a convex combination of at most $d + 1$ v_i's. Hence, if $r \geq d + 1$ then $o \in C_r$ and the optimal approximation error is 0. In most popular architectures, such as BERT (Devlin et al., 2019), $d = 64 \ll L$. This means that from the point of expressivity, *optimal-v-aware-65* can obtain a perfect approximation. Conversely, we will show in §4 that the performance of *optimal-v-oblivious-65* is substantially lower.

At the other extreme, when $r = 1$ (a single value vector), the above objective is equivalent to $\operatorname{argmin}_{i \in (1, \dots, L)} ||o - v_i||^2$ and can be simplified as

$$\operatorname{argmin}_i ||o||^2 + ||v_i||^2 - 2\langle v_i, o \rangle$$
$$= \operatorname{argmin}_i ||v_i||^2 - 2\langle v_i, \sum_j \alpha_j v_j \rangle$$
$$= \operatorname{argmin}_i ||v_i||^2 (0.5 - \alpha_i) - \sum_{j \neq i} \alpha_j \langle v_i, v_j \rangle.$$
$$(3)$$

This equation induces a ranking over value vectors that *depends* on the value vectors themselves, in contrast to a value-oblivious ranking induced solely by attention weights $\boldsymbol{\alpha}$.

If $v_1, \dots, v_L$ are orthogonal, the above equation further simplifies to $\operatorname{argmin}_i ||v_i||^2 (0.5 - \alpha_i) - \sum_{j \neq i} \alpha_j \cdot 0 = \operatorname{argmin}_i ||v_i||^2 (0.5 - \alpha_i)$. In this case, if some $\alpha_i \geq 0.5$ or if $v_1, \dots, v_L$ have equal norms, this would further simplify to $\operatorname{argmax}_i \alpha_i$, and would therefore be independent of the value-vectors v_i's, implying that a value-oblivious approximation would work well.

But such assumptions on $v_1, \dots, v_L$ do not hold in general and thus an approximation that only depends on α_i's can be sub-optimal. E.g., let v_1, v_2, v_3 be orthogonal vectors $(1, 0, 0)$, $(0, 2, 0)$, $(0, 0, 3)$ respectively and let $\alpha_1, \alpha_2, \alpha_3$ be $0.25, 0.35, 0.4$. Then v_3 with the highest attention weight α_3 has a squared distance of 3.79 from the true output $\sum_i \alpha_i v_i$ whereas v_1 with the least atten-

tion weight α_1 has only 2.49. In this case, *optimal-v-aware-1* induces exactly the opposite ranking of value vectors compared to *optimal-v-oblivious-1*. Moreover, if we increase the value 3 in v_3 to infinity, the approximation error will also infinitely grow. This example and, in general, Eq. 3 also show that the optimal ranking can be significantly different from the one induced by $\alpha_i ||v_i||$ proposed recently by (Kobayashi et al., 2020) for obtaining better interpretability of attention models.

Effect of kernel function Recently, Linear Transformer (Katharopoulos et al., 2020) proposed to replace the existing exponential kernel with more efficient kernels. We now show that replacing the exponential kernel with a polynomial kernel can lead to a drop in quality for current sparse approximation methods.

Intuitively, because the kernel function affects the skewness of $\boldsymbol{\alpha}$, it also affects the difference between the ranking induced by the optimal-value-aware approximation and the optimal-value-oblivious one. For simplicity, consider the case of orthogonal value vectors in which Eq. 3 simplifies to $\operatorname{argmin}_i ||v_i||^2 (0.5 - \alpha_i)$. From Eq. 1, we have $\alpha_i = \kappa(q, k_i)/\sum_j \kappa(q, k_j)$ which is $\langle q, k_i \rangle^C / \sum_j \langle q, k_j \rangle^C$ for the degree-C polynomial kernel. For $C = 0$, we have $\alpha_i = 1/L$, which gives $\operatorname{argmin}_i ||v_i||^2$. In this case, the value vectors become crucial when $\boldsymbol{\alpha}$ is uniform. On the other hand, assuming distinct inner products, for $C \gg 0$ we will obtain $\max_i \alpha_i \geq 0.5$, thereby reducing us to $\operatorname{argmax}_i \alpha_i$, where value vectors do not affect the approximation. The complexity of the Transformer grows exponentially with the degree C and thus in practice a low C must be used (e.g., degree-2 polynomial). In such case, $\boldsymbol{\alpha}$ is likely to be less skewed compared to the exponential kernel and more likely to induce a sub-optimal ranking.

In the next section, we empirically verify the above observations and show a significant performance gap between value-oblivious approximations and value-aware ones.

4 Experiments

We empirically verify our observations in the context of training causal and masked language models, which are known to strongly correlate with performance on downstream applications (Radford et al., 2019; Devlin et al., 2019).

Masked LM task   We form examples by sampling sequences and replacing sub-words with `<mask>` following the procedure in (Devlin et al., 2019). The model is trained to maximize the log probability of the masked out tokens and we evaluate the *error* of the model as the percentage of masked tokens predicted incorrectly. As approximate attention becomes increasingly relevant for long sequences, we train ROBERTA-4096 on sequences of length 4096 (Fig. 1). Training was warm-started using ROBERTA-base (Liu et al., 2019). Full details on the experimental setup are in §A.1. After training the model for $\sim$ 2.5M steps, the error of the model (that is, proportion of incorrect predictions) on the evaluation set was 24.2 (compared to 26.6 for an analogous training on 512-long sequences), ensuring that tokens in ROBERTA-4096 indeed attend over longer distances and result in higher quality representations. We then replace the attention function of the trained model with various approximation schemes and evaluate the resulting model on the evaluation set.
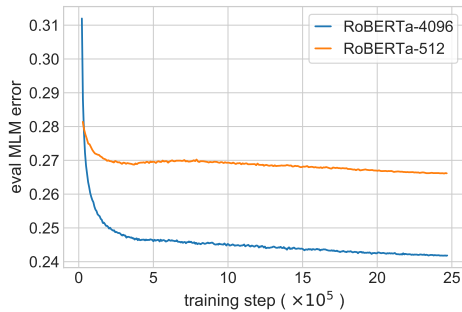

Figure 1: Evaluation error on the MLM task using vanilla attention (computing the full attention matrix).

We first compare *optimal-v-oblivious-r* to *optimal-v-aware-r*. We know that the approximation error of value-aware approximation is 0 for $r > 64$. For $r = 1$, we exhaustively go through all possible values and choose the one that minimizes the value-aware objective. As seen in Fig. 2 and Table 1, there is substantial gap between the two approximations. For instance, *optimal-v-oblivious-65* gives an MLM error of 43.5 whereas the error of *optimal-v-aware-65* is 24.2, since it can perfectly approximate full attention. Moreover, we compare *optimal-v-oblivious-r* to existing approximations: (a) *sliding-window-r*, where a position attends to $r/2$ positions to its left and right), (b) LSH attention (Kitaev et al., 2020) and (c) Performer attention (Choromanski et al., 2020). Fig. 2 shows that *sliding-window-r* trails behind *optimal-*

v-oblivious-r. LSH attention, which tries to emulate *optimal-v-oblivious-r*, either requires a large number of hash rounds or a large chunk size. Similarly, the Performer attention provides an unbiased approximation of the exponential kernel but suffers from high variance in practice.

exact	OVO 1	OVO 65	OVA 1	OVA 65	Performer 256 features	LSH $r = 64$ 4-rounds	LSH $r = 512$ 16-rounds
24.2	96.6	43.5	88.6	24.2	89.79	90.39	26.11

Table 1: MLM error of ROBERTA-4096 on the evaluation set using approximate attention described in §4. OVO r: *optimal-v-oblivious-r*, OVA r: *optimal-v-aware-r*. In LSH, each query attends to a total of r keys per hash round.
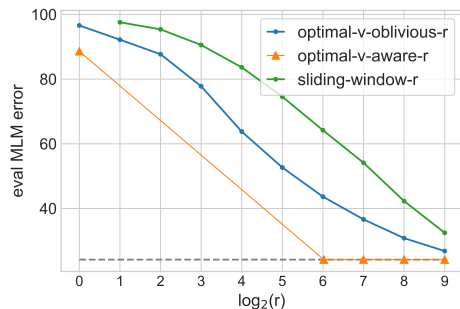

Figure 2: Evaluation of MLM error of ROBERTA-4096 after replacing vanilla attention with approximation schemes. Dashed line denotes error using vanilla attention.

Causal LM task   To investigate the effect of the kernel function on the quality of value-oblivious methods, we train a 6-layer Transformer LM over 512 tokens on WikiText-103 (Merity et al., 2017) (details in §A.2). We train 3 models with identical hyperparameters using the exponential, degree-2 polynomial, and elu kernels respectively and evaluate the trained models with value-aware and value-oblivious approximations. Again, *optimal-v-aware-r* substantially outperforms *optimal-v-oblivious-r* (Table 2), pointing to the potential of working on approximating the value-aware objective.

	exact	OVO-1	OVO-65	OVA-1	OVA-65
exponential	30.5	1031.1	33.5	280.3	30.5
polynomial (deg 2)	34.2	6700.2	310.2	1005.4	34.2
elu	35.3	1770.6	62.7	837.4	35.3

Table 2: Evaluation perplexity of models using approximate attention. OVO-r: *optimal-v-oblivious-r*, OVA-r: *optimal-v-aware-r*.

More importantly, comparing the approximation quality across different kernel functions (Fig. 3), we see that the gap between the three kernels is small when using full attention (512 keys) vectors. However, convergence is much slower for the

elu kernel, and especially the degree-2 polynomial, demonstrating that the approximation based on the top-r key vectors is sub-optimal when switching to a less skewed kernel, which is more affected by the value vectors.
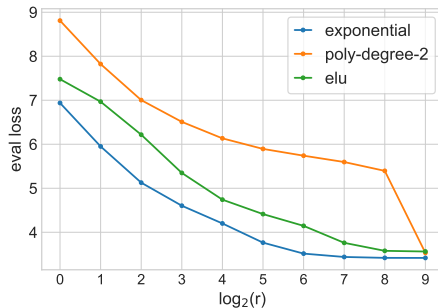

Figure 3: Evaluation loss (base e) of *optimal-v-oblivious-r* oracle on the causal LM task for distinct kernel functions.

5   Conclusions

In this work, we provide theoretical and empirical evidence against current practice of focusing on approximating the attention matrix in Transformers, while ignoring the value vectors. We propose a value-aware objective and argue that the efforts to develop more efficient Transformers should consider this objective function as a research target.

Acknowledgments

References

I. Bárány and R. Karasev. 2012. Notes about the carathéodory number. *Discrete & Computational Geometry*, 48:783–792.

Irwan Bello. 2021. Lambdanetworks: Modeling long-range interactions without attention. In *International Conference on Learning Representations*.

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

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, A. Gane, Tamás Sarlós, P. Hawkins, J. Davis, Afroz Mohiuddin, L. Kaiser, D. Belanger, Lucy J. Colwell, and Adrian Weller. 2020. Rethinking attention with performers. *ArXiv*, abs/2009.14794.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, W. Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *NAACL-HLT*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Association for Computational Linguistics (NAACL)*, pages 4171–4186, Minneapolis, Minnesota.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Ankit Gupta and Jonathan Berant. 2020. Gmat: Global memory augmentation for transformers. *arXiv preprint arXiv:2006.03274*.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *North American Association for Computational Linguistics (NAACL)*, pages 4129–4138.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

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*.

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

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online. Association for Computational Linguistics.

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*.

Stephen Merity, Caiming Xiong, James Bradbury, and R. Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. 2021. Random feature attention. In *International Conference on Learning Representations*.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. 2018. Deep contextualized word representations. In *North American Association for Computational Linguistics (NAACL)*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen-tau Yih, Sinong Wang, and Jie Tang. 2019. Blockwise self-attention for long document understanding. *arXiv preprint arXiv:1911.02972*.

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

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *ArXiv*, abs/2002.08910.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2020. Efficient content-based sparse attention with routing transformers. *arXiv preprint arXiv:2003.05997*.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020a. Sparse sinkhorn attention. In *International Conference on Machine Learning*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020b. Efficient transformers: A survey.

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 Conference on Empirical Methods in Natural Language Processing*.

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 (NIPS)*, pages 5998–6008.

A. Vyas, A. Katharopoulos, and F. Fleuret. 2020. Fast transformers with clustered attention.

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

Zihao Ye, Qipeng Guo, Quan Gan, Xipeng Qiu, and Zheng Zhang. 2019. Bp-transformer: Modelling long-range context via binary partitioning. *arXiv preprint arXiv:1911.04070*.

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 *The IEEE International Conference on Computer Vision (ICCV)*.

A Supplemental Material

A.1 Masked LM task

The instances for the MLM task (§4) were formed separately using the corpora listed in Table 3. For each dataset, after appending `</s>` token at the end of each document, the documents were arranged in a random order and concatenated into a single long text which was then tokenized into a list of sub-words. Depending upon the final input sequence length L of the experiment (512/4096) this list was chunked into full length $L - 2$ sequences which were then masked randomly following (Devlin et al., 2019) and enclosed within `<s>` and `</s>` tokens. To handle sequences longer than 512 tokens, the positional embeddings were used following (Gupta and Berant, 2020). The learning curves of ROBERTA-512 and ROBERTA-4096 are in Fig. 1.

corpus	all	training	evaluation
Wikipedia (10/2017)	2.67B	1.53B	1.02M
BookCorpus (Zhu et al., 2015)	1.06B	1.02B	1.02M
ArXiv (Cohan et al., 2018)	1.78B	1.53B	1.02M
PubMed (Cohan et al., 2018)	0.47B	510M	1.02M
PG19 (Rae et al., 2020)	3.06B	510M	1.02M

Table 3: Number of tokens in the datasets used for MLM training.

Hyperparameters For convenience, we denote the training hyperparameters using the following abbreviations, INS: number of training instances, BSZ: number of instances in a batch, ISZ: instance size, SQL: final input sequence length after rearranging BSZ instances each of length ISZ, LR: learning rate, WRM: linear LR warm-up proportion, EP: number of epochs, STP: number of optimizer steps, GAC: gradient accumulation steps, POSq: whether (y/n) q part is included in positional embeddings. The hyperparameters are listed in Table 4.

model	init	BSZ	ISZ	SQL	LR	WRM	EP	STP	POSq
ROBERTA-512	ROBERTA	8	512	512	5e-6	0.1	2	2.476M	n
ROBERTA-4096	ROBERTA	8	512	4096	5e-6	0.1	2	2.476M	y

Table 4: Training hyperparameters. Common parameters: INS=10M, dropout-rate=0.0, optimizer=Bert-Adam, β_1=0.9, β_2=0.98, weight-decay=0.01, max-grad-norm=5.0, seed=42, GAC=1.

Details of LSH attention Given L queries and L keys in $\mathbb{R}^d$, in each hash round, we sample a new matrix $R \in \mathbb{R}^{\frac{C}{2} \times d}$ of standard gaussians and hash the queries and keys as $H_R(x) = \text{argmax}([-Rx; Rx]) \in \{1, \dots, C\}$. We rearrange the queries (and similarly keys) according to their hash value, breaking ties using the original position, and then chunk them into L/B chunks of B vectors each. Denoting these chunks as $Q_1, \dots, Q_{L/B}$ and $K_1, \dots, K_{L/B}$, for each query in Q_i we compute its similarity scores with respect to all keys in K_{i-1}, K_i. I.e. in each hash round a query attends to $r = 2B$ keys. For each query, these similarity scores are accumulated over different hash rounds, and at the end normalized by their sum to get normalized attention scores over the keys. As recommended in the original paper (Kitaev et al., 2020), we use $C = 2L/B = 4L/r$ which in practice can be sub-optimal as rearrangement destroys the original locality structure.

Details of Performer attention Given L queries and L keys in $\mathbb{R}^d$ we divide each vector by $d^{\frac{1}{4}}$ to account for the temperature term in dot-product attention. For a given number F of features, we sample a random orthogonal matrix $R \in \mathbb{R}^{F \times d}$ as described in (Saxe et al., 2013) and provided as a tensor initialization option in PyTorch. We then map each vector to the feature space as $\Phi(x) = \frac{1}{\sqrt{F}} \exp\left(Rx - \frac{||x||^2}{2}\right) \in \mathbb{R}^F$ where $(-)$ and $\exp$ operations are applied element-wise. Similarity score of a query-key pair (q, k) is computed as $\langle \Phi(q), \Phi(k) \rangle$ and and is normalized by the sum of the similarity scores of q with all the keys. Computing this directly leads to numerical instability so we instead compute $\Phi(q) = \frac{1}{\sqrt{F}} \exp\left(Rq - \frac{||q||^2}{2} - \max(Rq)\right)$ for queries and $\Phi(k) = \frac{1}{\sqrt{F}} \exp\left(Rk - \frac{||k||^2}{2} - \max(RK)\right)$ where K is the matrix of all keys and $\max$ is over all elements of input.

The main idea behind Performer attention is that, for a vector w of standard gaussians, $\langle w, x \rangle \sim \mathcal{N}(0, ||x||^2)$ and from the properties of log-normal distributions, $\mathbb{E}_w[\exp(\langle w, x \rangle)] = \exp(\frac{||x||^2}{2})$. So, $\mathbb{E}_w[\exp(\langle w, q \rangle) \cdot \exp(\langle w, k \rangle)] = \mathbb{E}_w[\exp(\langle w, q + k \rangle)] = \exp(\frac{||q+k||^2}{2}) = \exp(\langle q, k \rangle + \frac{||q||^2}{2} + \frac{||k||^2}{2})$. Appropriately scaling both sides gives, $\mathbb{E}_w[\exp(\langle w, q \rangle - \frac{||q||^2}{2}) \cdot \exp(\langle w, k \rangle - \frac{||k||^2}{2})] = \exp(\langle q, k \rangle)$, which is exactly the term for the exponential kernel.

A.2 Causal LM task

For this task, we used the language modeling framework provided by Faiseq[2].

---

[2] https://github.com/pytorch/fairseq

Model and training details number of decoder layers: 6, hidden size: 512, head size: 64, number of model parameters: 156M, dataset: WikiText-103, training examples: 1801350, input sequence length: 512, β_1=0.9, β_2=0.98, weight-decay: 0.01, gradient clip-norm: none, learning rate: 0.0005, learning rate schedule: inverse square root, number of warmup updates: 4000, batch size: 128, epochs: 20, number of steps: 31520, minimum context-window during evaluation on test-set: 400.