Attention Alignment and Flexible Positional Embeddings Improve Transformer Length Extrapolation

Ta-Chung Chi
Carnegie Mellon University
tachungc@andrew.cmu.edu

Ting-Han Fan
Independent Researcher
tinghanf@alumni.princeton.edu

Alexander I. Rudnicky
Carnegie Mellon University
air@cs.cmu.edu

Abstract

An ideal length-extrapolatable Transformer language model can handle sequences longer than the training length without any fine-tuning. Such long-context utilization capability relies heavily on a flexible positional embedding design. Upon investigating the flexibility of existing large pre-trained Transformer language models, we find that the T5 family deserves a closer look, as its positional embeddings capture rich and flexible attention patterns. However, T5 suffers from the dispersed attention issue: the longer the input sequence, the flatter the attention distribution. To alleviate the issue, we propose two attention alignment strategies via temperature scaling. Our findings show improvement on the long-context utilization capability of T5 on language modeling, retrieval, multi-document question answering, and code completion tasks without any fine-tuning. This suggests that a flexible positional embedding design and attention alignment can go a long way toward Transformer length extrapolation. The code is released at: https://github.com/chijames/T5-Attention-Alignment

1 Introduction

Pre-training large Transformer language models on long sequences is inherently expensive due to self-attention's quadratic complexity w.r.t the input sequence length (Vaswani et al., 2017). Even with the help of memory-efficient attention (Rabe and Staats, 2021; Dao et al., 2022), the maximum supported input length of current open-source pre-trained Transformer language models are capped at 4,096 tokens (Touvron et al., 2023), limiting their efficacy in handling long-context tasks.

One notable research topic aiming to lift the input length restriction is *Length Extrapolation* (Press et al., 2022). Ideally, a length-extrapolatable Transformer language model is trained on short sequences and can perform equally well on longer ones without any further fine-tuning. This is made possible with carefully designed positional embeddings (Press et al., 2022; Chi et al., 2022, 2023). Unfortunately, existing approaches are tailored for natural language modeling, a task known to have strong recency bias, and they often do not perform well on other seemingly simple tasks such as passkey, topic, and line retrieval (Mohtashami and Jaggi, 2023; Li et al., 2023).

Retrieval Tasks						
Criteria	Topic		Line		Passkey	
	512	15k	512	15k	512	15k
P_{max}	0.28	0.12	0.27	0.11	0.32	0.24
H	3.47	6.63	3.47	7.04	3.09	5.97

Table 1: **The Dispersed Attention Issue of Flan-T5-XL Encoder.** P_{max} is the average maximum probability and H is the average entropy. After increasing the sequence length from 512 to 15k, we observe larger entropy and smaller maximum probability, implying a flatter self-attention distribution.

To circumvent the recency bias, we sift through the positional embeddings of existing open-source large pre-trained Transformer language models, shown in Table 2, to find a flexible design, and the T5 family (Raffel et al., 2020) comes to our attention. As visualized in Figure 1, the flexibility of T5's positional embeddings allows it to encourage recency bias on one head and discourage that on another head. However, there is no free lunch: T5 suffers from the dispersed attention issue as shown in Table 1. That is, the attention distributions of long input sequences tend to be flatter than those of short input sequences. As a remedy, we propose two fine-tuning-free attention alignment strategies via Softmax temperature scaling (Yao et al., 2021; Su, 2021) to mitigate the dispersed attention issue: maximum probability (P_{max}) and entropy (H) alignment.

We validate the effectiveness of our alignment

Models	T5 (2020)	OPT (2022)	ChatGLM (2022)	LLaMA (2023)	Falcon (2023)	Pythia (2023)	XGen (2023)	BLOOM (2022)	MPT (2023)
PE.	Learned Relative	Learned Absolute	Rotary Relative	Rotary Relative	Rotary Relative	Rotary Relative	Rotary Relative	ALiBi Relative	ALiBi Relative

Table 2: **Open-source Transformer language models and their positional embeddings.** T5 is the only model equipped with learnable relative positional embeddings, which enable its long-context utilization capability.

strategies on tasks including language modeling, retrieval, multi-document question answering, and code completion. We also provide a theoretical analysis of how the alignment strategies work under the hood by investigating the relation between the Softmax temperature and data distribution.

2 Related Work

Transformer Positional Embeddings Transformer-based models rely on positional embeddings to encode positional information. We summarize open-source large pre-trained Transformer language models and their positional embeddings in Table 2. The relative variants are widely adopted due to their better empirical performance (Su et al., 2021) and possible length-extrapolation capability (Press et al., 2022). In this work, we place special focus on the T5 positional embeddings due to their flexibility as shown in Figure 1.

Transformer Length Extrapolation Existing research on Transformer length extrapolation is mostly confined to the task of natural language modeling (Press et al., 2022; Chi et al., 2022, 2023). Unfortunately, the reported positive results do not carry over to long-context retrieval (Mohtashami and Jaggi, 2023; Li et al., 2023). This contrastive observation can be explained by models' short empirical receptive field (Chi et al., 2023). In short, the strong decaying prior of positional embeddings prevents models from accessing distant tokens that may be necessary for retrieval tasks. In this work, we improve the flexible positional embeddings of T5 to get around this limitation.

Transformer Position Interpolation Instead of performing direct length extrapolation, a different line of research conducts model fine-tuning on long input sequences (Chen et al., 2023), where the main focus is to identify the most efficient fine-tuning scheme that can improve long-context utilization. Positive results have been reported on retrieval tasks (Li et al., 2023). However, we argue that fine-tuning incurs additional costs since it needs 1) GPU resources to perform long sequence fine-tuning with large models and 2) a pre-defined target sequence length, which still imposes a sequence length upper limit. Our proposed methods can circumvent these two limitations.

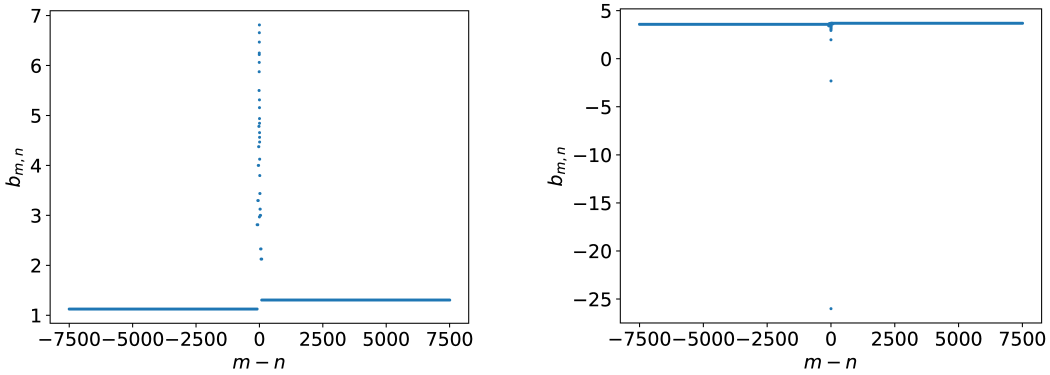
Retrieval Tasks with Transformers Transformer-based approaches often consist of a retriever and a reader to overcome the context length restriction (Guu et al., 2020; Lewis et al., 2020; Izacard and Grave, 2021; Borgeaud et al., 2022). The retriever retrieves relevant text snippets from a very large database and the reader digests the retrieved information to generate the correct output. Our proposed attention alignment strategy can be used to significantly increase the input sequence length of the reader, thereby allowing more retrieved information to participate in the decision process. For small-scale retrieval problems, our methods even obviate the need for context segmentation and the external key-value store used in prior work (Mohtashami and Jaggi, 2023), serving as a more elegant approach.

Softmax Temperature Scaling To increase the length extrapolation capability of Transformers, previous work (Yao et al., 2021; Su, 2021; Peng et al., 2023) scales the temperature of Softmax logarithmically w.r.t the sequence length. Our entropy alignment strategy is also inspired by this line of research except that we adopt a different procedure outlined below in Algorithm 1. Interestingly, our results in § 7 show that the logarithmic temperature scaling scheme is more similar to our proposed maximum probability alignment strategy.

3 Long-context Retrieval Tasks with T5

3.1 Why Retrieval?

As suggested by recent work (Mohtashami and Jaggi, 2023; Li et al., 2023), the task of long-context retrieval serves as a controllable benchmark to measure how well a Transformer language model utilizes long-context inputs. One prominent characteristic of retrieval tasks is that only a subset of the input is of interest, requiring a model to accu-



(a) 1st Attention Head (b) 27nd Attention Head

Figure 1: **Visualization of T5 Positional Embeddings.** To plot figures of $b_{m,n}$, we set $m = 7500$ and vary the value of n from 0 to 15k. Each attention head of a Flan-T5-XL encoder learns a set of positional embeddings that capture different attention bias. For example, the positional embeddings in the left figure encourage the model to focus on nearby tokens. In contrast, the ones in the right figure let the model focus on only remote tokens.

rately pick up the necessary information. The other characteristic is that the key information can sit anywhere in an input, requiring a model to attend flexibly. Finally, the controllable aspect allows us to gradually increase the input sequence length to test the models' length extrapolation capability.

3.2 Why T5?

To solve retrieval tasks using Transformer language models, it is necessary to choose a positional embedding design that permits accurate and flexible length-extrapolatable attention. After checking through the existing positional embeddings in Table 2, we find that the T5 family (Raffel et al., 2020) fits our needs. As for other candidates, learnable absolute positional embeddings (Vaswani et al., 2017; Zhang et al., 2022) must be evaluated within the training length. ALiBi (Press et al., 2022) and Rotary (Su et al., 2021) have a recency bias; they cannot extrapolate easily without fine-tuning.

For each attention head, T5 encoder maintains a bucket (B) of 32 learnable parameters and assigns the relative positional bias (rpe bias) $b_{m,n}$ as[1]

$$b_{m,n} = \begin{cases} \mathrm{B}[m-n], \text{ if } 0 \le m-n < 8 \\ \mathrm{B}[n-m+16], \text{ if } -8 < m-n < 0 \\ \mathrm{B}[\min(15, 8 + \lfloor \frac{\log((m-n)/8)}{\log(128/8)} \cdot 8 \rfloor)], \text{ if } 8 \le m-n \\ \mathrm{B}[\min(31, 24 + \lfloor \frac{\log((n-m)/8)}{\log(128/8)} \cdot 8 \rfloor)], \text{ if } m-n \le -8, \end{cases}$$

where $0 \le m < L$ and $0 \le n < L$ are two position indices. $b_{m,n}$ will be added to the (m, n)-th entry of the $L \times L$ self-attention matrix. The summation becomes the input to the temperature-scaled Softmax. We plot the learned rpe bias of a T5 encoder in Figure 1. We can tell that its attention heads encode rich attention patterns. For example, head 1 learns to focus on the nearby tokens whereas head 27 learns to ignore the nearby tokens and allow access to faraway tokens.

3.3 The Dispersed Attention Issue of T5 Encoder

Unfortunately, directly applying T5 models on retrieval tasks does not yield perfect results. Upon inspecting the intermediate model states, we find that a longer input sequence consists of more tokens competing for the same amount (i.e., Softmax sums to 1) of attention, resulting in the dispersed attention issue. In Table 1, we see that the longer the input sequence, the flatter the self-attention distribution. The situation is not hopeless if the desired information still attains a higher attention weight than the remaining tokens. Our proposed solution in § 4 will let the key information stand out.

4 Proposed Methods

A natural solution to the dispersed attention issue described in § 3 is to sharpen the self-attention distribution. This can be achieved by reducing the temperature τ during extrapolation. We set the extrapolation temperature τ_{ex} such that the sharpness during training with $\tau_{tr} = 1$ and that during extrapolation with $\tau_{ex} < 1$ are roughly the same. As a measurement of sharpness, we explore the maxi-

[1] https://github.com/huggingface/transformers/blob/v4.33.2/src/transformers/models/t5/modeling_t5.py#L390

Algorithm 1 Attention Alignment Strategies

Require: A short sequence of length L_{tr} and a long sequence of length $L_{ex} > L_{tr}$. Encoder E. Alignment mode M.
Ensure: The Softmax temperature τ
  function FINDS(τ, M)
    Set temperature of all Softmax to τ
    $s \leftarrow [\,]$
    for operation in E **do**
      Perform the operation
      if operation is $\text{Softmax}_\tau(l)$ **then**
        if M is Maximum Probability **then**
          Append $\max(\text{Softmax}_\tau(l))$ to s
        else if M is Entropy **then**
          Append $\text{H}(\text{Softmax}_\tau(l))$ to s
        end if
      end if
    end for
    return $\text{avg}(s)$
  end function
  $\overline{S^{tr}}(1) \leftarrow \text{FINDS}(1.0, M)$
  for $\tau_{ex} = 1.0, 0.95, 0.9, \cdots, 0.5$ **do**
    $\overline{S^{ex}}(\tau_{ex}) = \text{FINDS}(\tau_{ex}, M)$
  end for
  return τ_{ex} s.t. $\overline{S^{ex}}(\tau_{ex}) \approx \overline{S^{tr}}(1)$

mum probability or entropy of a distribution. In other words, our proposed solution is to align the maximum probability or entropy of training and extrapolation distributions by adjusting τ_{ex}.

Concretely, let $l^{(i)} \in \mathbb{R}^L$ be the i-th pre-Softmax logit vector of a T5 encoder, where $L \in \{L_{tr}, L_{ex}\}$ is the sequence length. The post-Softmax distribution of $l^{(i)}$ is $\text{P}^{(i)}(\tau) = \text{Softmax}_\tau(l^{(i)})$. The maximum probability and entropy of $\text{P}^{(i)}(\tau)$ are $\text{P}^{(i)}_{\max}(\tau)$ and $\text{H}^{(i)}(\tau)$, respectively.

Take the maximum probability alignment strategy as an example: We first run the forward pass and compute the average maximum probability under temperature τ over all logit vectors: $\overline{\text{P}_{\max}}(\tau) = (1/N)\sum_i \text{P}^{(i)}_{\max}(\tau)$ where $N = R \times H \times L$ is the number of logit vectors in a T5 encoder with R layers, H heads, and length-L sequences. Since the temperature is 1 during training and τ_{ex} during extrapolation, we denote the average maximum probability during training as $\overline{\text{P}^{tr}_{\max}}(1)$ and that during extrapolation as $\overline{\text{P}^{ex}_{\max}}(\tau_{ex})$. Finally, to align the maximum probabilities, we adjust τ_{ex} s.t. $\overline{\text{P}^{ex}_{\max}}(\tau_{ex}) \approx \overline{\text{P}^{tr}_{\max}}(1)$. In practice, we do a grid search on τ_{ex} from 1.0 to 0.5. We outline the procedure of the alignment strategies in Algorithm 1.

Note that our proposed methods do not require any model fine-tuning or gradient computations. The only overhead is estimating the temperature τ_{ex} using Algorithm 1 and a few length L_{ex} sequences. Once the temperature is decided, it will be held fixed, rendering our methods simple and efficient. In addition, our fine-tuning free methods do not lead to performance regression on short L_{tr} sequences commonly observed on long-context fine-tuned models (Roziere et al., 2023).

5 Experiments

We compare the two alignment strategies against the length-only Softmax temperature scaling scheme $\tau = \log_{L_{ex}} L_{tr}$ (Yao et al., 2021; Su, 2021) and LongChat-13B-16K (Li et al., 2023). Note that LongChat-13B-16K (Li et al., 2023), the best baseline, was fine-tuned from LLaMA (Touvron et al., 2023) on long sequences of length 16k while our proposed methods do not need any fine-tuning. Our experiments are conducted on an A6000 GPU.

Language Modeling						
Models	Sequence Length (L_{ex})					
	1024	2048	4096	8192	15000	Avg.
T5-Large-LM	35.9	40.1	>1k	>1k	>1k	> 1k
w/ $\text{P}_{\max}$	34.7	45.5	45.2	45.5	52.7	**44.7**
w/ H	40.2	43.9	45.6	54.6	56.0	48.1
w/ $\log_{L_{ex}} L_{tr}$	39.8	38.2	47.4	45.3	55.9	45.3
T5-XL-LM	28.3	>1k	>1k	>1k	>1k	> 1k
w/ $\text{P}_{\max}$	30.2	36.0	31.6	41.7	50.0	37.9
w/ H	30.4	36.8	38.4	53.3	63.4	44.4
w/ $\log_{L_{ex}} L_{tr}$	27.3	29.4	31.7	39.3	45.8	**34.7**
T5-XXL-LM	109	>1k	>1k	>1k	>1k	> 1k
w/ $\text{P}_{\max}$	32.2	29.7	29.5	36.6	44.3	34.5
w/ H	26.8	28.1	34.2	37.8	43.8	**34.1**
w/ $\log_{L_{ex}} L_{tr}$	27.1	36.1	33.9	246	43.8	77.5

Table 3: **Language Modeling Performance.** We report the average perplexity of 500 sequences. The lower the better.

5.1 Language Modeling

We use the LM-Adapted T5 models[2] for this experiment. We set $L_{tr} = 512$. Following previous work on Transformer length extrapolation, we perform an intrinsic evaluation on language modeling (Press et al., 2022; Chi et al., 2022, 2023). Ideally, our proposed methods should alleviate the perplexity explosion problem during extrapolation. As we can see in Table 3, both alignment strategies dramatically improve (lower) the perplexity. We also observe that scaling the temperature solely based on sequence lengths is not the optimal strategy, as indicated by the sudden perplexity increase of the $\log_{L_{ex}} L_{tr}$ strategy. We will provide an in-depth

[2] https://github.com/google-research/text-to-text-transfer-transformer/blob/main/released_checkpoints.md#lm-adapted-t511lm100k

Retrieval Tasks																	
Models	Topic, # of topics					Line, # of lines						Passkey, # of sentences					Avg.
	5	10	15	20	25	200	300	400	500	600	680	20k	30k	40k	50k	55k	
Flan-T5-Large	99	100	97	97	83	97	100	92	96	93	92	62	47	31	16	9	76
w/ $\mathrm{P}_{\max}$	96	90	86	94	98	99	98	98	98	98	100	84	90	85	79	85	**92**
w/ H	59	32	16	0	3	97	90	94	83	93	88	29	25	21	15	22	48
w/ $\log_{L_{ex}} L_{tr}$	88	79	75	61	55	99	99	98	99	97	98	74	63	51	41	35	76
Flan-T5-XL	100	100	100	100	100	96	90	77	57	45	26	100	100	100	100	100	87
w/ $\mathrm{P}_{\max}$	100	100	100	100	100	97	90	89	80	70	62	100	99	100	100	100	**93**
w/ H	99	98	97	96	96	95	87	88	79	70	71	100	100	100	100	100	92
w/ $\log_{L_{ex}} L_{tr}$	99	100	100	100	100	98	88	81	86	60	67	100	100	100	100	99	92
Flan-T5-XXL	100	100	100	99	99	100	100	98	95	84	82	100	100	100	100	100	97
w/ $\mathrm{P}_{\max}$	100	100	100	99	99	97	99	96	97	94	95	100	98	100	100	100	**98**
w/ H	100	100	97	98	94	99	92	92	76	58	58	100	100	100	100	100	92
w/ $\log_{L_{ex}} L_{tr}$	100	100	99	98	92	100	98	94	93	84	90	100	100	100	100	100	97
LongChat	100	100	100	99	89	100	91	93	83	78	59	100	100	99	100	99	93

Table 4: **Performance of Retrieval Tasks.** Each number is the averaged accuracy computed over 100 sequences. The LongChat model corresponds to LongChat-13B-16K (Li et al., 2023). It is a LLaMA-13B (Touvron et al., 2023) model fine-tuned on sequences of length 16k using positional interpolation (Chen et al., 2023). Flan-T5-XXL has 11B parameters. The maximum sequence lengths (L_{ex}) of the three tasks are around 14.5k to 15.5k tokens.

discussion on this topic in § 7. Note that perplexity is not our primary focus since it often cannot accurately reflect the long-context utilization capability of Transformers on practical tasks (Li et al., 2023).

5.2 Long-context Retrieval

The tasks are formulated in the Question Answering (QA) format; therefore, we use the Flan-T5 models to leverage their instruction-following capability. We set $L_{tr} = 512$. Inspired by recently proposed retrieval tasks, we evaluate the proposed alignment strategies on three of these. Topic retrieval requires a model to retrieve the first topic in a long and multi-topic conversation (Li et al., 2023). Line retrieval has a long series of key-value pairs, and a model needs to retrieve the value corresponding to the given key (Li et al., 2023). Passkey retrieval hides a passkey in a long junk text snippet, and a model needs to return that passkey (Mohtashami and Jaggi, 2023).

As we can see in Table 4, the retrieval performance is greatly boosted after the Flan-T5 models are equipped with our proposed attention alignment strategies. In particular, the maximum probability alignment strategy provides better results across the board. Other baselines such as MPT (Team, 2023) and ChatGLM2 (Du et al., 2022) perform worse than LongChat. Please refer to Li et al. (2023) for more details. We also present the optimal temperature given by Algorithm 1 in Table 10 in Appendix A.5. In short, the temperature decreases when the input sequence length increases. We will provide additional temperature analysis below, in § 7.

5.3 Multi-document Question Answering

We again use the Flan-T5 models to leverage their instruction-following capability. We set $L_{tr} = 512$. We follow the multi-document question-answering task settings and data splits detailed in Liu et al. (2023). In short, the input consists of a question Q and multiple documents extracted from NaturalQuestions (Kwiatkowski et al., 2019) related to Q, where one of the documents (golden doc) contains the ground truth answer to Q. As shown in Table 5, when a model is equipped with the proposed maximum probability alignment strategy, it consistently outperforms the original model across model sizes and number of input documents.

Apart from the better task performance, we believe that the attention dispersed attention issue discussed in § 3 can help demystify the lost-in-the-middle phenomenon (Liu et al., 2023) of this task: Transformer models tend to perform worse when the ground truth sits near the middle of the input context. Let us recall the relative positional embedding of head 27 learned in Figure 1, if the ground truth answer sits in the middle, it will have long contexts from both sides competing for the attention weight. If this hypothesis is correct, we can

Multi-document Question Answering										
Models	10 Docs	20 Docs	30 Docs, golden doc at different positions							
	Avg.	Avg.	0	4	9	14	19	24	29	Avg.
Flan-T5-Large	52.4	43.3	52.6	42.0	36.5	34.0	33.9	33.9	37.9	38.7
w/ $\mathrm{P}_{\max}$	**53.1**	**44.2**	50.8	44.5	39.5	36.4	35.9	35.8	37.0	**40.0**
Improvement	+0.7	+0.9	-1.8	+1.5	+3.0	+2.4	+2.0	+1.9	-0.9	+1.3
w/ H	52.1	43.2	47.6	41.1	35.2	33.5	32.2	33.3	34.2	36.7
w/ $\log_{L_{ex}} L_{tr}$	53.2	44.5	50.6	44.1	39.3	36.3	35.8	35.8	37.2	39.9
Flan-T5-XL	59.4	51.2	58.4	44.6	40.0	39.9	41.7	46.4	54.8	46.5
w/ $\mathrm{P}_{\max}$	**61.1**	**53.6**	60.9	49.1	46.0	44.9	46.3	49.1	55.7	**50.3**
Improvement	+1.7	+2.4	+2.5	+4.5	+6.0	+5.0	+4.6	+2.7	+0.9	+3.8
w/ H	60.5	52.4	52.4	43.5	42.1	40.3	42.0	42.9	51.3	44.9
w/ $\log_{L_{ex}} L_{tr}$	60.9	**53.6**	61.0	49.1	46.1	44.7	46.1	48.7	55.4	50.2
Flan-T5-XXL	63.6	56.9	58.9	49.1	48.1	47.5	48.9	53.1	61.2	52.4
w/ $\mathrm{P}_{\max}$	63.7	**57.7**	60.4	52.5	51.0	50.2	51.3	53.5	59.1	**54.0**
Improvement	+0.1	+0.8	+1.5	+3.4	+2.9	+2.7	+2.4	+0.4	-2.1	+1.6
w/ H	63.6	57.1	61.0	53.4	50.8	50.3	50.7	51.9	55.7	53.4
w/ $\log_{L_{ex}} L_{tr}$	**63.9**	57.6	61.5	53.3	51.3	50.3	51.1	53.0	57.2	**54.0**

Table 5: **Performance of Multi-document QA.** Numbers are accuracy. Full score is 100. The maximum sequence length (L_{ex}) of 30 documents is around 5k. The improvement row represents the absolute accuracy improvement after a Flan-T5 model is equipped with our proposed maximum probability alignment strategy. For the full performance breakdown, please refer to Table 15 in Appendix A.7.

expect the performance boost to be more prominent when the answer appears near the middle. We reveal the performance breakdown when the number of input documents is 30. As we can see in the improvement row, those cases indeed achieve greater improvements.

Our strategies are not always perfect: The performance drops if the ground truth answer is at position 29. We believe T5 might have already handled this case pretty well due to the recency bias learned on some attention heads, and our additional temperature scaling sharpens the distribution too aggressively. We acknowledge this trade-off in § 9.

5.4 Code Key Retrieval and Completion

To test the generalizability of the alignment strategies, we apply our methods to the CodeT5+ model (Wang et al., 2023)[3] that was pre-trained on code data with 770M parameters. We set $L_{tr} = 768$. We do not experiment with larger CodeT5+ models since they do not follow the T5 architecture, but use other positional embeddings. We conduct two experiments on the LCC dataset (Guo et al., 2023), which is highly similar to the classic PY150 dataset (Raychev et al., 2016) except that the input context length is much longer.

For the code key retrieval experiment, we sample several code files from LCC along with a special function that only returns an integer from 1 to 100. We concatenate them and ask a model to generate the returned integer at the end (Roziere et al., 2023). Considering that this is essentially a passkey retrieval task in the code domain, we briefly report the average accuracy of 100 test cases when the input sequence length is around 16k: 0 (Original CodeT5+), **87** (w/ $\mathrm{P}_{\max}$), 80 (w/ H), and 85 (w/ $\log_{L_{ex}} L_{tr}$). We can see that the maximum probability alignment strategy performs the best.

For the code completion experiment, a model needs to generate the next line of code given some prior code as the context. The metrics are Exact Match (EM) and Edit similarity (ES) on a per line basis (Svyatkovskiy et al., 2020). We report the results in Table 6 using the context length bucketing format. While both alignment strategies improve the performance substantially, $\mathrm{P}_{\max}$ is better; however, its EM performance lags behind $\log_{L_{ex}} L_{tr}$ when the sequence length increases. We additionally include an extrapolation-free baseline, *truncation*, that truncates the long input context to the most recent $L_{tr} = 768$ tokens. Both $\mathrm{P}_{\max}$ and $\log_{L_{ex}} L_{tr}$ perform better than this baseline when $L_{ex} < 6000$, indicating that they can indeed benefit from longer ($6000/768 = 7.8$x) contexts without any fine-tuning.

[3] https://huggingface.co/Salesforce/codet5p-770m-py

Code Completion Exact Match						
Models	Sequence Length (L_{ex})					
	1k	2k	3k	4k	5k	6k
CodeT5+	19.6	19.0	11.3	2.6	0.1	0.0
w/ $\mathrm{P}_{\max}$	21.1	22.5	21.7	21.5	19.3	22.7
w/ H	19.5	18.7	13.7	9.0	7.9	9.0
w/ $\log_{L_{ex}} L_{tr}$	**21.6**	**23.0**	**22.1**	**22.0**	**20.6**	**24.3**
w/ *truncation*	20.0	19.2	19.3	19.2	17.1	21.4

Code Completion Edit Similarity						
Models	Sequence Length (L_{ex})					
	1k	2k	3k	4k	5k	6k
CodeT5+	62.4	59.6	53.1	38.9	18.3	10.4
w/ $\mathrm{P}_{\max}$	65.9	65.7	65.3	65.6	**63.1**	64.9
w/ H	64.8	62.5	54.1	43.0	43.0	44.8
w/ $\log_{L_{ex}} L_{tr}$	**66.3**	**66.1**	**65.2**	**66.4**	63.0	66.1
w/ *truncation*	65.3	64.2	64.2	65.6	62.2	**66.9**

Table 6: **Code Completion Performance.** Full score is 100. We set $L_{tr} = 768$. The bucket *n*k contains the data with length in [*n*k, (*n*+1)k), $n \in [1, 6]$. For example, the bucket 3k contains data with length in [3000, 4000). See Table 13 and 14 in Appendix A.6 for the full performance breakdown of L_{ex} up to 16k tokens.

5.5 Overall Observations

First, the maximum probability alignment strategy is the most reliable and best-performing method across most tasks and settings, echoing our discussion in § 3.1: For most data, only a subset of the input is useful for a model process at a time. The maximum probability alignment strategy captures this characteristic naturally, thereby outperforming the entropy alignment strategy that cares more about the holistic distribution.

Second, deciding the optimal temperature solely based on sequence lengths, e.g. $\tau = \log_{L_{ex}} L_{tr}$, is not robust enough. For example, the perplexity of $\log_{L_{ex}} L_{tr}$ suddenly increases (worse) on T5-XXL-LM, in Table 3, while the other strategies maintain stable results. As another example, it fails to improve the retrieval performance on the Flan-T5-Large model, shown in Table 4.

5.6 Application to Other Models

We also tried applying our proposed method to models using Rotary positional embeddings. However, we are not able to achieve the same length-extrapolatable performance without fine-tuning. Two immediate questions follow:

Why is fine-tuning needed when we apply our method to Rotary-based models? Because Rotary still suffers from the recency bias issue as we discussed in § 3.2, while T5 does not. To the best of our knowledge, such recency bias can only be overcome via long sequence fine-tuning. Take a concurrent work that focuses on models equipped with Rotary, YaRN (Peng et al., 2023), as an example: If we omit its fine-tuning step, its performance on the passkey retrieval task drops substantially as shown in Table 7.

How costly is the long sequence fine-tuning step? Let us take a look at the numbers reported by the authors of YaRN[4]: "Our run was around 300s/epoch on an 8x A100 node as well. Took about 24 hours to train for 400 steps." Using AWS, the fine-tuning expenses will be 32.77*24=786.48 USD.[5] In contrast, finding the optimal softmax scaling temperature of the longest inference sequence of a task using our method with a T5 model only takes 20 seconds on an A6000 GPU.

6 Theoretical Analysis

6.1 Assumptions

To shed more light on the underlying mechanisms of the two alignment strategies, we establish the connection between the softmax temperature τ and data distribution under empirically verified assumptions. We focus on the 0-th layer (closest to the input embeddings) and take the average over all logit vectors across attention heads. Note that this is just a crude approximation of Algorithm 1 for analysis purposes since 1) a Transformer language model typically encompasses multiple layers, and 2) in Algorithm 1, we take the maximum probability or entropy of individual logit vectors as opposed to the average one.

Assumption 1. *The length L average logit vector is normally distributed, i.e., its entry $l_i \sim N(0, \sigma^2)$.*

To compute the *average logit vector*, we start with a input sequence of length L. Using a Transformer model with H attention heads (specifically, a T5 Encoder in our context), we generate $H \times L$ pre-softmax logit vectors, each with a length of L. Here, the number of layers is 1 because we focus on the 0-th layer. These logit vectors are then individually sorted, and we subsequently calculate the

[4] https://github.com/jquesnelle/yarn/issues/32

[5] 32.77 is the on-demand price per hour of a p4d.24xlarge instance.

Model	Sequence Length				
	512	1024	2048	4096	8192
YaRN w/ fine-tuning	100	100	100	100	95
YaRN w/o fine-tuning	85	85	75	50	30
Ours Flan-T5-XL w/o fine-tuning	100	99	100	100	100

Table 7: **The Effect of Fine-tuning for Passkey Retrieval.** Numbers are accuracy. Full score is 100. The flexibility of T5 allows our method to extrapolate well on long sequences without any fine-tuning. In contrast, YaRN (Peng et al., 2023) requires a costly fine-tuning step on long sequences to regain performance.

average of all $H \times L$ sorted logit vectors, resulting in the average logit vector of length L.

To assess whether the average logit entries follow a Gaussian distribution, we make use of QQ plots, as illustrated in Figure 2. The linearity of the plot serves as an indicator – the closer the points are to the identity line, the more Gaussian the distribution.

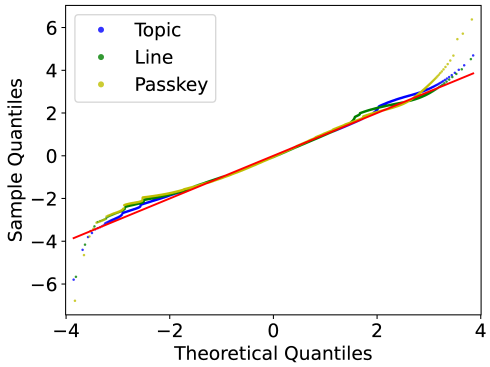


Figure 2: **QQ plots of Flan-T5-XL.** We experiment with short and long sequences. The red reference line is y=x. We use sequences of length around 512 for this plot. The plot for sequences of length around 15k looks highly similar. Please refer to Appendix A.1 for details.

Assumption 2. *The largest logit entry of the average logit vector during training and extrapolation is the same:* $l^{ex}_{\max} = l^{tr}_{\max}$. *See Table 8.*

Criteria	Retrieval Tasks					
	Topic		Line		Passkey	
	512	15k	512	15k	512	15k
$l_{\max}$	8.61	8.80	8.71	8.97	8.75	8.85

Table 8: **Largest Logit Entry of Flan-T5-XL.** $l_{\max}$ is the largest logit entry of the average logit vector.

6.2 Maximum Probability Alignment

Proposition 1. *Under Assumption 1 and 2, we can adjust the temperature* τ *to align the maximum probability* $\mathrm{P}^{tr}_{\max} = \mathrm{P}^{ex}_{\max}$

$$\tau \approx \frac{\log L_{tr} + \log \mathrm{P}^{tr}_{\max} + \sigma^2_{tr}/2}{\log L_{ex} + \log \mathrm{P}^{tr}_{\max} + \sigma^2_{ex}/(2\tau^2)}.$$
$$= \frac{B}{A + \frac{C}{\tau^2}} = \frac{B\tau^2}{A\tau^2 + C}.$$

Assuming $\tau \neq 0$, *we solve the quadratic equation* $A\tau^2 - B\tau + C = 0$ *to get* τ. *We pick the larger root as our final solution. See proof in Appendix A.2.*

6.3 Entropy Alignment

Proposition 2. *Under Assumption 1, we can adjust the temperature* τ *to align the entropy* $H_{tr} = H_{ex}$

$$\tau \approx \frac{\sigma_{ex}}{\sqrt{\sigma^2_{tr} + 2\log \frac{L_{ex}}{L_{tr}}}}$$

See proof in Appendix A.3.

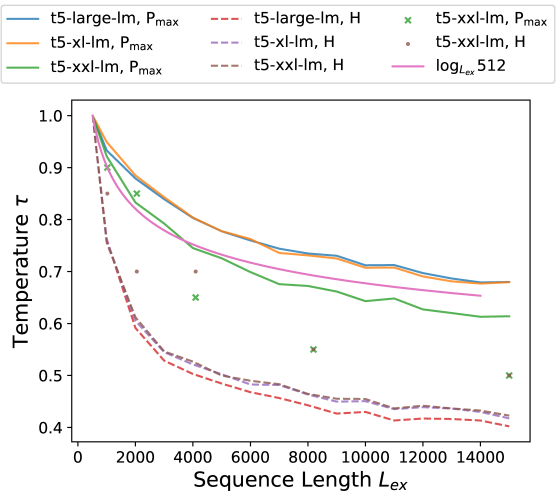


Figure 3: **Language Modeling Temperature Analysis.** Curves are from Proposition 1 & 2. Dots and crosses are from Algorithm 1.

7 Discussion

The goal of this section is to explain the observations made in § 5 via the lens of temperature

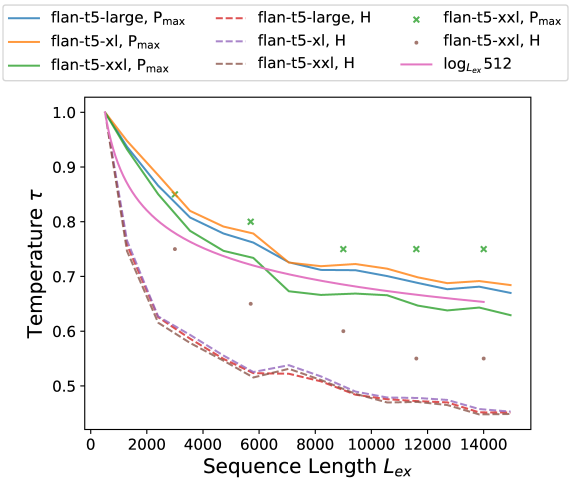


Figure 4: **Topic Retrieval Temperature Analysis.** Curves are from Proposition 1 & 2. Dots and crosses are from Algorithm 1.

analysis. We visualize Proposition 1 and 2 by plotting the temperature curves in Figure 3 and 4. We evaluate $\mathrm{P}_{\max}^{tr}$ and σ_{tr} at the training length and σ_{ex} at every extrapolation length considering only the 0-th layer. You may find the temperature curves for the other tasks in Appendix A.4.

First, while both proposed strategies lower the temperature when the input sequence length increases, the entropy alignment strategy does so more aggressively, possibly leading to its inferior performance observed in Table 4 and 5 (w/ H). This can be seen by comparing the curves from Propositions 1 and 2 or dots from Algorithm 1.

Second, deciding the optimal temperature based on sequence lengths, e.g. $\tau = \log_{L_{ex}} L_{tr}$, is not the most robust method. It gives too high of a temperature in Figure 3 compared to Algorithm 1. In other words, it does not sharpen the distribution enough, possibly explaining its perplexity spike in Table 3. On the other hand, it overly lowers the temperature in Figure 4, thereby failing to improve the retrieval performance on Flan-T5-Large in Table 4.

8 Conclusion

In this paper, we show that the T5 model family has great potential when it comes to Transformer length extrapolation. We propose the maximum probability and entropy alignment strategies to fix T5's dispersed attention issue without model fine-tuning. We conduct experiments on natural language modeling, retrieval, multi-document question answering, and code completion tasks to demonstrate the effectiveness of our proposed methods. Finally, we present a simplified theoretical analysis to elucidate how the temperature is scaled to achieve attention alignment. We hope that our work can inspire future length-extrapolatable Transformer designs.

9 Limitations

We base our theoretical analysis on a simplified Transformer language model, which might be further improved by taking all the layers and their interactions into account. In addition, we find that different layers have different degrees of distribution flatness, which could be leveraged in future work to perform per-layer fine-grained attention alignment. Finally, our temperature scaling scheme sometimes sharpens a distribution too aggressively in the multi-document question-answering and code completion experiments. This drawback could be possibly improved by designing a more fine-grained attention alignment strategy.

10 Ethics Statement

Our work improves the amount of context a Transformer language model can process. Inappropriate usage of the proposed technique might lead to negative societal impacts including the potential loss due to wrong predictions and ethical challenges on the improper use of the model. However, these implications apply to most language model research and are not unique to this specific work.

References

Salesforce AI. 2023. Long sequence modeling with xgen: A 7b llm trained on 8k input sequence length.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

Ta-Chung Chi, Ting-Han Fan, Peter J Ramadge, and Alexander I Rudnicky. 2022. KERPLE: Kernelized Relative Positional Embedding for Length Extrapolation. In *Advances in Neural Information Processing Systems (NeurIPS)*, New Orleans, USA.

Ta-Chung Chi, Ting-Han Fan, Alexander I Rudnicky, and Peter J Ramadge. 2023. Dissecting Transformer Length Extrapolation via the Lens of Receptive Field Analysis. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, Toronto, Canada.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335.

Daya Guo, Canwen Xu, Nan Duan, Jian Yin, and Julian McAuley. 2023. Longcoder: A long-range pre-trained language model for code completion. In *International Conference on Machine Learning*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph E Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. 2023. How long can open-source llms truly promise on context length.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*.

Amirkeivan Mohtashami and Martin Jaggi. 2023. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*.

Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*.

Markus N Rabe and Charles Staats. 2021. Self-attention does not need o (n^2) memory. *arXiv preprint arXiv:2112.05682*.

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. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Veselin Raychev, Pavol Bielik, and Martin Vechev. 2016. Probabilistic model for code with decision trees. *ACM SIGPLAN Notices*, 51(10):731–747.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François

Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Jianlin Su. 2021. Scaling attention via the lens of entropy invariance.

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

Alexey Svyatkovskiy, Shao Kun Deng, Shengyu Fu, and Neel Sundaresan. 2020. Intellicode compose: Code generation using transformer. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1433–1443.

The MosaicML NLP Team. 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms. *https://www.mosaicml.com/blog/mpt-30b*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi DQ Bui, Junnan Li, and Steven CH Hoi. 2023. Codet5+: Open code large language models for code understanding and generation. *arXiv preprint arXiv:2305.07922*.

Martin B Wilk and Ram Gnanadesikan. 1968. Probability plotting methods for the analysis for the analysis of data. *Biometrika*, 55(1):1–17.

Shunyu Yao, Binghui Peng, Christos Papadimitriou, and Karthik Narasimhan. 2021. Self-attention networks can process bounded hierarchical languages. 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 3770–3785, Online. Association for Computational Linguistics.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

A Appendix

A.1 QQ Plots for Assumption 1

A QQ plot (Wilk and Gnanadesikan, 1968) is a graphical technique used for comparing two probability distributions by plotting their quantiles against each other. A point (x, y) corresponds to a quantile from the second distribution (y-coordinate) plotted against the same quantile from the first distribution (x-coordinate). When the two distributions under comparison are similar, the points in the QQ plot will roughly align with the identity line, y = x. In our case, where we aim to determine the degree of Gaussian behavior in the average logit vector, the linearity of the plot serves as an indicator – the closer the points are to the identity line, the more Gaussian the distribution.

We present the QQ plots for two lengths, 512 and 15k, on the three retrieval tasks in Figure 5. They are all close to the red reference line, indicating that their form is highly Gaussian.

A.2 Detailed Derivation of Proposition 1

Let $l_{\max}$ be the largest value in the logit vector l. Let τ be the temperature of the Softmax function. The probability of the largest entry is

$$\mathrm{P}_{\max} = \frac{e^{l_{\max}/\tau}}{\sum_{i=1}^{L} e^{l_i/\tau}}.$$

Since Softmax is shift-invariant, the logit vector can always be made zero-mean: $\sum_i l_i = 0$. Next, according to Assumption 1, the denominator of Softmax can be approximated as

$$\sum_{i=1}^{L} e^{l_i/\tau} \approx L \cdot \mathbb{E}[e^{l_i/\tau}] = L \cdot e^{\sigma^2/(2\tau^2)} \tag{1}$$

This implies $\mathrm{P}_{\max}$ is approximately

$$\mathrm{P}_{\max} \approx \frac{e^{l_{\max}/\tau}}{L e^{\sigma^2/(2\tau^2)}}$$

During the training stage, the temperature τ is 1

$$\mathrm{P}^{tr}_{\max} \approx \frac{e^{l^{tr}_{\max}}}{L_{tr} e^{\sigma^2_{tr}/2}},$$

which gives an expression of the largest logit entry during the training stage

$$l^{tr}_{\max} \approx \log\left(\mathrm{P}^{tr}_{\max} L_{tr} e^{\sigma^2_{tr}/2}\right) \tag{2}$$

According to Assumption 2, the largest probability during the extrapolation stage can be simplified as

$$\mathrm{P}^{ex}_{\max} \approx \frac{e^{l^{ex}_{\max}/\tau}}{L_{ex} e^{\sigma^2_{ex}/(2\tau^2)}} \overset{\text{A. 2}}{=} \frac{e^{l^{tr}_{\max}/\tau}}{L_{ex} e^{\sigma^2_{ex}/(2\tau^2)}} \overset{(2)}{\approx} \frac{\left(\mathrm{P}^{tr}_{\max} L_{tr} e^{\sigma^2_{tr}/2}\right)^{1/\tau}}{L_{ex} e^{\sigma^2_{ex}/(2\tau^2)}}$$

Since τ is a free parameter during extrapolation, we adjust it to carry out the maximum probability alignment strategy. Rearranging the terms gives Proposition 1.

A.3 Detailed Derivation of Proposition 2

The entropy of a discrete probability computed by Softmax is

$$H = -\sum_i \frac{e^{l_i/\tau}}{D} \log \frac{e^{l_i/\tau}}{D} = \log D - \frac{\sum_i \frac{l_i}{\tau} e^{l_i/\tau}}{D},$$

where $D = \sum_i e^{l_i/\tau}$ is the denominator of Softmax, which can be approximated using Eq. (1). On the other hand, we note that $\sum_i l_i e^{l_i} \approx L\mathbb{E}[le^l]$. When $l \sim N(0, \sigma^2)$, $\mathbb{E}[le^l]$ is approximated as

$$\begin{aligned}
\mathbb{E}[le^l] &= \int_{-\infty}^{\infty} \frac{le^l}{\sigma\sqrt{2\pi}} e^{-\frac{l^2}{2\sigma^2}} dl \\
&= \int_{-\infty}^{\infty} \frac{l}{\sigma\sqrt{2\pi}} e^{\frac{2\sigma^2 l - l^2}{2\sigma^2}} dl \\
&= \int_{-\infty}^{\infty} \frac{l}{\sigma\sqrt{2\pi}} e^{-\frac{(l-\sigma^2)^2 - \sigma^4}{2\sigma^2}} dl \\
&= e^{\sigma^2/2} \int_{-\infty}^{\infty} \frac{l}{\sigma\sqrt{2\pi}} e^{-\frac{(l-\sigma^2)^2}{2\sigma^2}} dl \\
&= e^{\sigma^2/2}\sigma^2
\end{aligned} \tag{3}$$

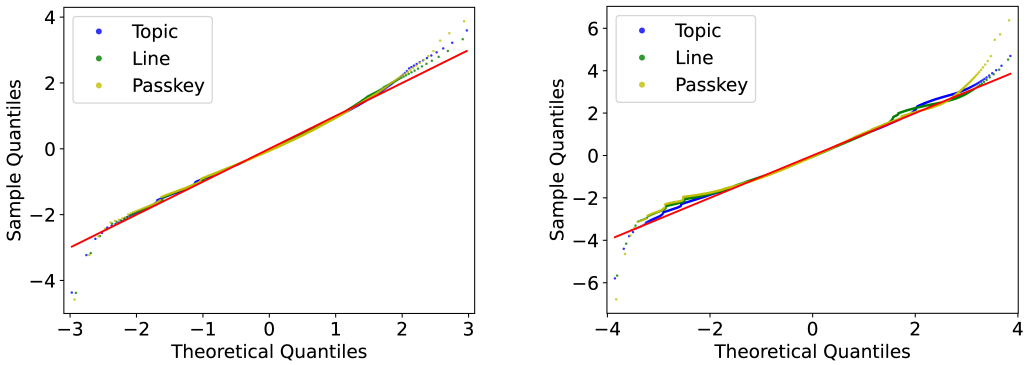
Thus, combining Eq. (1) and (3), the entropy H is approximated as

$$\begin{aligned}
H &\approx \log L + \frac{\sigma^2}{2\tau^2} - \frac{L e^{\sigma^2/(2\tau^2)} \frac{\sigma^2}{\tau^2}}{L e^{\sigma^2/(2\tau^2)}} \\
&= \log L - \frac{\sigma^2}{2\tau^2}
\end{aligned}$$

Since τ is set to 1 during the training stage, we have $H_{tr} \approx \log L_{tr} - \frac{\sigma^2_{tr}}{2}$. During extrapolation, we align the entropy (i.e., $H_{ex} = H_{tr}$) by adjusting τ.

$$\log L_{ex} - \frac{\sigma^2_{ex}}{2\tau^2} \approx H_{ex} = H_{tr} \approx \log L_{tr} - \frac{\sigma^2_{tr}}{2}.$$

Since τ is a free parameter during extrapolation, we adjust it to apply the entropy alignment strategy. Rearranging the terms gives Proposition 2.



(a) Short sequences around 512 (b) Long sequences around 15k

Figure 5: **QQ plots of Flan-T5-XL.** We experiment with short and long sequences. The red reference line is y=x. The more closely the scatter plots follow the red reference line, the more Gaussian they are.

A.4 More Real-world Temperature Plots

We verify Proposition 1 and 2 on the remaining tasks by plotting the temperature curves in Figure 6, 7, 8, and 9. We empirically evaluate σ_{tr} at the training length and σ_{ex} every extrapolation length considering only the 0-th layer.

The real temperatures given by Algorithm 1 are usually higher than those derived from the two propositions. After checking the per-layer attention distributions, we find that the 0-th layer has flatter distributions compared to higher layers. Because the two propositions are derived based on the 0-th layer and a flatter distribution needs a lower temperature to correct, the temperatures given by them tend to be lower than the ones given by Algorithm 1 that takes the average of temperatures across all layers.

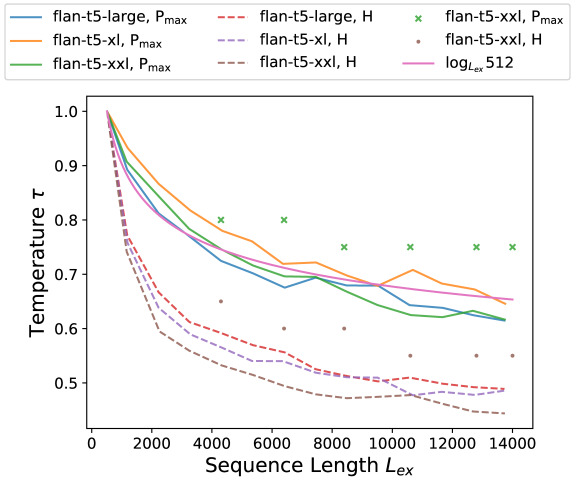


Figure 6: **Line Retrieval Temperature Analysis.** Curves are given by Proposition 1 and 2. Cross signs and dots are given by Algorithm 1. $\log_L 512$ is given by Yao et al. (2021); Su (2021).

A.5 Detailed Temperature Breakdown

We report the temperatures for all tasks across model sizes given by Algorithm 1 in Table 9, 10, 11, and 12.

A.6 Performance Breakdown of Code Completion

We report the performance breakdown of Exact Match and Edit Similarity across lengths in Table 13 and 14.

A.7 Performance Breakdown of Multi-document Question Answering

We report the performance breakdown of different numbers of input documents in Table 15.

B Scientific Artifacts

The pretrained models we used belong to the T5 model family, which is released under the Apache 2.0 license. The models are used in this work for research purposes only. For the data used to train T5 models, please refer to Raffel et al. (2020); Lester et al. (2021); Chung et al. (2022) for details. Except for the LCC Python data, other task data is written in English. We already report the number of data instances in § 5 for the language modeling and retrieval tasks. As for the multi-doc QA and code related tasks, we follow the original data splits.

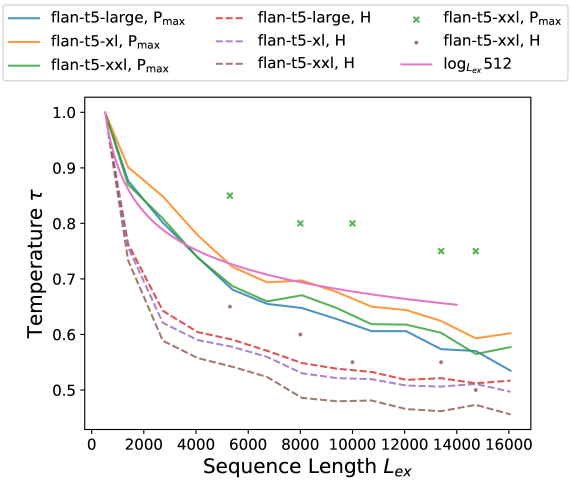


Figure 7: **Passkey Retrieval Temperature Analysis.** Curves are given by Proposition 1 and 2. Cross signs and dots are given by Algorithm 1. $\log_L 512$ is given by Yao et al. (2021); Su (2021).

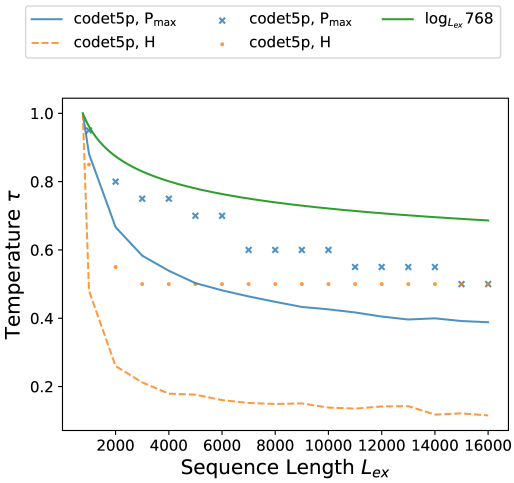


Figure 9: **Code Completion Temperature Analysis.** Curves are given by Proposition 1 and 2. Cross signs and dots are given by Algorithm 1. $\log_L 768$ is given by Yao et al. (2021); Su (2021).

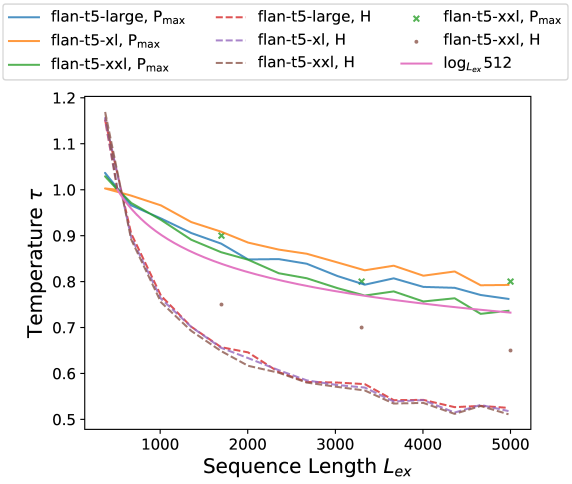


Figure 8: **Multi-doc QA Temperature Analysis.** Curves are from Proposition 1 & 2. Dots and crosses are from Algorithm 1.

Language Modeling					
Models	Sequence Length (L_{ex})				
	1024	2048	4096	8192	15000
T5-Large-LM					
w/ $\mathrm{P}_{\max}$	0.9	0.85	0.8	0.75	0.7
w/ H	0.8	0.7	0.6	0.5	0.5
T5-XL-LM					
w/ $\mathrm{P}_{\max}$	0.9	0.85	0.75	0.7	0.6
w/ H	0.85	0.7	0.55	0.5	0.5
T5-XXL-LM					
w/ $\mathrm{P}_{\max}$	0.9	0.85	0.65	0.55	0.5
w/ H	0.85	0.7	0.7	0.55	0.5
w/ $\log_{L_{ex}} L_{tr}$	0.9	0.82	0.75	0.69	0.65

Table 9: **Temperatures of Language Modeling.** We search the optimal temperature from 1.0, 0.95, 0.9, $\cdots$, 0.5. We set $L_{tr} = 512$.

Retrieval Tasks																
Models	Topic, # of topics					Line, # of lines						Passkey, # of sentences				
	5	10	15	20	25	200	300	400	500	600	680	20k	30k	40k	50k	55k
Flan-T5-Large																
w/ $\mathrm{P}_{\max}$	0.85	0.8	0.75	0.75	0.75	0.85	0.8	0.8	0.75	0.75	0.75	0.85	0.80	0.80	0.75	0.75
w/ H	0.7	0.6	0.55	0.5	0.5	0.65	0.55	0.55	0.5	0.5	0.5	0.6	0.55	0.5	0.5	0.5
Flan-T5-XL																
w/ $\mathrm{P}_{\max}$	0.8	0.75	0.7	0.65	0.65	0.8	0.75	0.75	0.7	0.70	0.7	0.85	0.8	0.75	0.75	0.75
w/ H	0.7	0.55	0.55	0.5	0.5	0.6	0.55	0.55	0.5	0.5	0.5	0.7	0.65	0.6	0.6	0.6
Flan-T5-XXL																
w/ $\mathrm{P}_{\max}$	0.85	0.8	0.75	0.75	0.75	0.8	0.8	0.75	0.75	0.75	0.75	0.85	0.8	0.8	0.75	0.75
w/ H	0.75	0.65	0.6	0.55	0.55	0.65	0.6	0.6	0.55	0.55	0.55	0.65	0.6	0.55	0.55	0.5
w/ $\log_{L_{ex}} L_{tr}$	0.79	0.72	0.69	0.67	0.65	0.74	0.71	0.69	0.67	0.66	0.65	0.73	0.69	0.67	0.66	0.65

Table 10: **Temperatures of Retrieval Tasks.** We search the optimal temperature from 1.0, 0.95, 0.9, $\cdots$, 0.5. The maximum lengths of the three tasks are all around 14.5k to 15.5k tokens (L_{ex}). We set $L_{tr} = 512$.

Multi-document Question Answering			
Models	10 Docs	20 Docs	30 Docs
	$L_{ex} = 1700$	$L_{ex} = 3300$	$L_{ex} = 5000$
Flan-T5-Large			
w/ Max.	0.9	0.85	0.8
w/ Ent.	0.75	0.65	0.6
Flan-T5-XL			
w/ Max.	0.85	0.75	0.75
w/ Ent.	0.75	0.65	0.55
Flan-T5-XXL			
w/ Max.	0.9	0.8	0.8
w/ Ent.	0.75	0.7	0.65
w/ $\log_{L_{ex}} L_{tr}$	0.84	0.77	0.73

Table 11: **Temperatures of Multi-document Question Answering.** We search the optimal temperature from 1.0, 0.95, 0.9, $\cdots$, 0.5. Different golden document positions have the same temperature. We set $L_{tr} = 512$.

Code Completion																
Models	Sequence Length (L_{ex})															
	1k	2k	3k	4k	5k	6k	7k	8k	9k	10k	11k	12k	13k	14k	15k	16k
CodeT5+																
w/ $\mathrm{P}_{\max}$	0.95	0.8	0.75	0.75	0.7	0.7	0.6	0.6	0.6	0.6	0.55	0.55	0.55	0.55	0.5	0.5
w/ H	0.85	0.55	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
w/ $\log_{L_{ex}} L_{tr}$	0.96	0.87	0.83	0.8	0.78	0.76	0.75	0.74	0.73	0.72	0.71	0.71	0.7	0.7	0.69	0.69

Table 12: **Temperatures of Code Completion.** We search the optimal temperature from 1.0, 0.95, 0.9, $\cdots$, 0.5. The maximum length is around 16k tokens (L_{ex}). We set $L_{tr} = 768$.

Code Completion Exact Match															
Models	Sequence Length (L_{ex})														
	1k	2k	3k	4k	5k	6k	7k	8k	9k	10k	11k	12k	13k	14k	15k
CodeT5+	19.6	19.0	11.3	2.6	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
w/ $\mathrm{P}_{\max}$	21.1	22.5	21.7	21.5	19.3	22.7	16.1	14.4	13.4	20.6	16.0	15.3	12.3	16.7	4.5
w/ H	19.5	18.7	13.7	9.0	7.9	9.0	10.3	8.8	10.8	12.1	11.7	10.2	9.2	11.1	2.3
w/ $\log_{L_{ex}} L_{tr}$	21.6	23.0	22.1	22.0	20.6	24.3	20.7	18.6	19.1	22.4	13.8	20.3	15.4	19.4	11.4
w/ *truncation*	20.0	19.2	19.3	19.2	17.1	21.4	21.1	18.0	19.1	25.2	18.1	20.3	16.9	27.8	15.9

Table 13: **Full Exact Match Breakdown of Code Completion Edit Similarity.** We set $L_{tr} = 768$. Numbers in red are higher than their counterpart in the w/*truncation* row. The bucket *n*k contains the data with length in [*n*k, (*n*+1)k), $n \in [1, 15]$.

Code Completion															
Models	Sequence Length (L_{ex})														
	1k	2k	3k	4k	5k	6k	7k	8k	9k	10k	11k	12k	13k	14k	15k
CodeT5+	62.4	59.6	53.1	38.9	18.3	10.4	6.1	4.0	4.5	5.0	6.7	5.1	6.4	4.4	3.5
w/ $\mathrm{P}_{\max}$	65.9	65.7	65.3	65.6	63.1	64.9	60.0	60.0	58.1	57.5	56.2	56.0	52.1	56.9	39.9
w/ H	64.8	62.5	54.1	43.0	43.0	44.8	47.7	47.0	47.6	51.2	44.3	49.7	50.3	57.4	42.0
w/ $\log_{L_{ex}} L_{tr}$	66.3	66.1	65.2	66.4	63.0	66.1	61.9	58.8	61.6	57.8	54.2	57.9	48.7	52.2	48.6
w/ *truncation*	65.3	64.2	64.2	65.6	62.2	66.9	66.8	61.8	64.1	65.1	63.5	63.9	61.5	67.6	60.8

Table 14: **Full Edit Similarity Breakdown of Code Completion.** We set $L_{tr} = 768$. Numbers in red are higher than their counterpart in the w/*truncation* row. The bucket *n*k contains the data with length in [*n*k, (*n*+1)k), $n \in [1, 15]$.

Multi-document Question Answering															
Models	10 Docs			20 Docs					30 Docs						
	0	4	9	0	4	9	14	19	0	4	9	14	19	24	29
Flan-T5-Large	60.6	48.5	48.0	54.5	44.0	39.6	38.0	40.2	52.6	42.0	36.5	34.0	33.9	33.9	37.9
w/ Max.	60.9	49.8	48.6	53.5	45.6	40.8	39.7	41.3	50.8	44.5	39.5	36.4	35.9	35.8	37.0
w/ Ent.	58.9	50.1	47.3	52.4	45.2	40.4	38.0	40.0	47.6	41.1	35.2	33.5	32.2	33.3	34.2
w/ $\log_{L_{ex}} L_{tr}$	60.2	51.1	48.4	53.8	46.0	41.4	39.4	41.7	50.6	44.1	39.3	36.3	35.8	35.8	37.2
Flan-T5-XL	64.0	55.4	58.9	60.6	47.9	45.1	47.3	55.3	58.4	44.6	40.0	39.9	41.7	46.4	54.8
w/ Max.	65.3	57.3	60.8	62.2	51.6	49.0	49.4	56.0	60.9	49.1	46.0	44.9	46.3	49.1	55.7
w/ Ent.	64.7	56.7	60.0	59.3	50.1	47.9	49.8	55.1	52.4	43.5	42.1	40.3	42.0	42.9	51.3
w/ $\log_{L_{ex}} L_{tr}$	65.1	57.0	60.6	62.2	51.7	48.8	49.5	56.0	61.0	49.1	46.1	44.7	46.1	48.7	55.4
Flan-T5-XXL	65.1	61.0	64.6	61.1	53.9	52.4	54.7	62.4	58.9	49.1	48.1	47.5	48.9	53.1	61.2
w/ Max.	66.2	61.8	63.2	62.8	55.9	54.4	55.6	59.6	60.4	52.5	51.0	50.2	51.3	53.5	59.1
w/ Ent.	67.3	62.1	61.3	63.2	56.1	54.1	54.3	57.6	61.0	53.4	50.8	50.3	50.7	51.9	55.7
w/ $\log_{L_{ex}} L_{tr}$	66.7	61.9	63.1	63.1	56.0	54.7	55.1	59.0	61.5	53.3	51.3	50.3	51.1	53.0	57.2

Table 15: **Full Performance Breakdown of Multi-document Question Answering.** The numbers are accuracy. Full score is 100. 0, 4, 9... indicate the position of the golden document that contains the answer to a question.