

# With Greater Text Comes Greater Necessity: Inference-Time Training Helps Long Text Generation

Y. Wang\* D. Ma\* D. Cai

Confidential Institutes

ywang462-c@my.cityu.edu.hk, tianjibai@foxmail.com, dcai@se.cuhk.edu.hk

## Abstract

Long text generation, such as novel writing or discourse-level translation with extremely long contexts, presents significant challenges to current language models. Existing methods mainly focus on extending the model’s context window through strategies like length extrapolation. However, these approaches demand substantial hardware resources during the training and/or inference phases.

Our proposed method, Temp-Lora, introduces an alternative concept. Instead of relying on the KV cache to store all context information, Temp-Lora embeds this information directly into the model’s parameters. In the process of long text generation, we use a temporary Lora module, progressively trained with text generated previously. This approach not only efficiently preserves contextual knowledge but also prevents any permanent alteration to the model’s parameters given that the module is discarded post-generation.

Extensive experiments on the PG19 language modeling benchmark and the GuoFeng discourse-level translation benchmark validate the effectiveness of Temp-Lora. Our results show that: 1) Temp-Lora substantially enhances generation quality for long texts, as indicated by a **13.2%** decrease in perplexity on a subset of PG19, and a **29.6%** decrease in perplexity along with a **53.2%** increase in BLEU score on GuoFeng, 2) Temp-Lora is compatible with and enhances most existing long text generation methods, and 3) Temp-Lora can greatly reduce computational costs by shortening the context window. While ensuring a slight improvement in generation quality (a decrease of 3.8% in PPL), it enables a reduction of 70.5% in the FLOPs required for inference and a 51.5% decrease in latency.

## 1 Introduction

Long text generation has become increasingly important in a variety of real-world applications, rang-

ing from creative writing assistance (Shi et al., 2022), chat-style AI assistant (OpenAI, 2023) to generative agents (Park et al., 2023). However, the generation of coherent and contextually relevant long text poses significant challenges to language models (LMs), particularly in terms of understanding and maintaining contexts that are longer than the model’s pre-defined context window size.

Existing methods, whether based on length extrapolation (Press et al., 2021; Su et al., 2023) or context window extension (Chen et al., 2023c; Han et al., 2023; Dao et al., 2022; Peng et al., 2023; Chen et al., 2023a), aims to store extensive text information within the KV cache, thereby improving the model’s long text comprehension. However, they demand significant hardware resources during training and/or inference. Consequently, in many applications where LMs are frequently queried for long text processing, users often resort to other strategies such as memory or summarization to reduce computational cost (Park et al., 2023).

In this paper, we propose an alternative method, Temp-Lora, which stores context information in the model’s parameters instead of the KV cache. The core idea of our approach is extremely simple: we store the context information in a temporary Lora module (Hu et al., 2021) that only exists during long text generation. We update this module in a streaming fashion during the generation process, using the generated content as training data, thereby achieving the goal of storing knowledge in the model parameters. Once inference is complete, this module is discarded to avoid permanently impacting the model’s parameters. This approach allows us to efficiently store nearly infinite context information without extending the context window.

We evaluate Temp-Lora on two benchmark datasets, PG19 (Rae et al., 2019) and GuoFeng (Wang et al., 2023b). We evaluate Temp-Lora across models with different context window sizes and find that Temp-Lora substantially

\*Equal Contribution

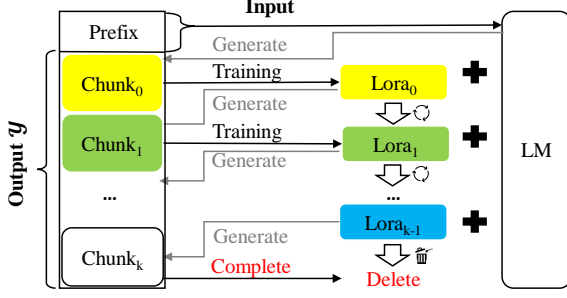


Figure 1: The framework of long text generation with Temp-Lora

reduces their perplexity on long texts with a large margin (-13.2% on a subset PG19 and -29.6% on a subset of GuoFeng). This trend of reduced perplexity becomes increasingly pronounced as the context length increased, which can be simply concluded as “**with greater text comes greater necessity**”. Further analysis also revealed that Temp-Lora, by shortening the maximum input length, can greatly reduce hardware consumption.

## 2 Temp-Lora

The Temp-Lora framework, depicted in Figure 1, presents a straightforward yet innovative approach for long text generation. At the heart of this method lies the progressive training of a temporary Lora module (Hu et al., 2021), which is named Temp-Lora, on previously-generated text in an autoregressive manner. The continuous adaptation and refinement of the Temp-Lora module ensure an evolving understanding of both recent and distant contexts. It is important to highlight the framework’s adaptability in handling cases where the initial input lengths are already substantial, such as novels or academic papers. In such scenarios, we may proactively pre-train the Temp-Lora module with the contexts, thereby laying a robust foundation for enhanced context comprehension in the generation process.

The algorithm is detailed in Algorithm 1. During the generation process, tokens are generated chunk-by-chunk. At each chunk generation, we use **the most recent  $L_{\mathcal{X}}$  tokens** as the input  $\mathcal{X}$  for generating the subsequent tokens. Once the number of generated tokens hits a predefined **chunk size  $\Delta$** , we initiate the training of the Temp-Lora module using the latest chunk and start the next chunk generation afterwards. It is important to note that  $L_{\mathcal{X}} + \Delta \leq \mathcal{W}$ , where  $\mathcal{W}$  is the model’s context window size. In our experiments, we set  $\Delta + L_{\mathcal{X}}$  equal to  $\mathcal{W}$  to take full advantage of the model’s

---

### Algorithm 1: Long Text Generation with Temp-Lora

---

**Input:** Input  $\mathcal{X}$ , Input Length  $L_{\mathcal{X}}$ , Training Input  $\mathcal{X}_T$ , Training Input Length  $L_T$ ; Chunk size  $\Delta$ ; Learning rate  $\alpha$ ; Epoch number  $n$ ; LM  $\Theta$ .

**Output:** Long sequence  $\mathcal{Y}$

```

1 if  $\text{len}(\mathcal{X}) > L_{\mathcal{X}}$  then
2   Pre-train  $\Theta$  on  $\mathcal{X}$  to obtain LM with an
   initial Temp-Lora module  $\Theta_0$ ;
3 end
4 Total token number:  $i \leftarrow \text{len}(\mathcal{X})$ ;
5 Temp-Lora ID:  $k \leftarrow 0$ ;
6 Token number in this chunk:  $m \leftarrow 0$ ;
7 Output sequence  $\mathcal{Y} = \mathcal{X}$ ;
8  $\mathcal{X} = \mathcal{X}[-(\min(L_{\mathcal{X}}, \text{len}(\mathcal{X})) : )]$ ;
9 while In Generation do
10   if  $m < \Delta$  then
11      $\mathcal{Y}[i] = \Theta_k(\mathcal{X} + \mathcal{Y}[-m : ])$ ;
12      $i ++, m ++$ ;
13   end
14   else
15      $\text{Chunk}_k = \mathcal{Y}[-m : ]$ ;
16      $\mathcal{X}_T = \mathcal{Y}[-(L_T + m) : -m]$ ;
17      $k ++$ ;
18     Train  $\Theta_k$ : In  $\mathcal{X}_T \rightarrow \text{Out } \text{Chunk}_k$ 
19     (n epochs, learning rate  $\alpha$ );
20      $\mathcal{X} = \mathcal{Y}[-L_{\mathcal{X}} : ]$ ;
21      $m = 0$ ;
22   end
23 end
24 Destroy Temp-Lora module:  $\Theta_k \leftarrow \Theta$ ;
```

---

context window size.

For the training of the Temp-Lora module, it is crucial to recognize that learn to generate the new chunk without any condition may not constitute a meaningful training objective and potentially lead to significant overfitting. To address this concern, we incorporate the preceding  $L_T$  tokens of each chunk into our training process, using them as the input and the chunk as the output.

**Cache Reuse** For more efficient inference, we propose a strategy called **cache reuse**. In the standard framework, after updating the Temp-Lora module, we need to re-compute the KV states with the updated parameters. Alternatively, we can also reuse the existing cached KV states while employing the updated model for subsequent text genera-

tion. Concretely, we only recompute the KV states with the latest Temp-Lora module when the model generates up to the maximum length (context window size  $\mathcal{W}$ ). We empirically find that cache reuse can accelerate the generation speed without significantly compromising the generation quality.

### 3 Experiments

We evaluate the proposed Temp-Lora framework using the Llama2 (Touvron et al., 2023) families considering its wide adoption and popularity. Its effectiveness is evaluate in two different long text generation tasks: 1) Novel Generation; and 2) Discourse-Level Literary Translation.<sup>1</sup>

**Dataset:** The first dataset we adopt is a subset of the long text language modeling benchmark, PG19 (Rae et al., 2019). It is a well-established benchmark that consists of more than 28K books which were published before 1919. Since long texts are required to evaluate the effectiveness of the Temp-Lora framework, and considering that some of the PG-19 data might already be included in Llama2’s training set, we selected the 100 books with the highest PPL from those whose lengths range between 200K and 600K tokens. From these, we randomly sampled 40 books as our test set.

We also evaluate the effectiveness of Temp-Lora on a downstream task, Discourse-Level Literary Translation, with a randomly sampled subset of GuoFeng dataset from WMT 2023 (Wang et al., 2023b,a). This subset contains 20 web novels, originally written in Chinese by various novelists and subsequently translated into English by professional translators. Their lengths (EN + ZH) range from a minimum of 84K to a maximum of 370K.

We designate Chinese as the source language and English as the target language. We merge sentences from the GuoFeng dataset into segments, each approximately 512 tokens in length. The model’s input includes the most recent source and target texts, along with the current source segment, and it is required to output the translation of this segment.

**Baselines:** We apply the Temp-Lora framework to three Llama2 variants: the standard **Llama2-7B-4K**<sup>2</sup>, standard **Llama2-13B-4K**<sup>3</sup>, **Llama2-**

**7B-32K**<sup>4</sup> whose context window is extended via position interpolation (Chen et al., 2023c), and a chat-style model **Yi-Chat-6B**<sup>5</sup>.

**Evaluation Metric:** The primary metric is perplexity (PPL), a standard measure in language modeling to assess the model’s prediction capability. We employ a sliding window approach for perplexity measurement as suggested by (Press et al., 2021). In the translation task, in addition to PPL, we also employed two common evaluation metrics used in machine translation, BLEU (Papineni et al., 2002) and COMET (Rei et al., 2020), for comprehensive evaluation.

**Setup:** We set the chunk size  $\Delta = 1024$  for all models. For the Llama2-7B-32K, we additionally set two wider chunk size  $\Delta = 2048$  and 4096 to investigate the effects of chunk size on generation quality and computational cost.

The input length  $L_{\mathcal{X}}$  for all models is set to be the difference between the window size  $\mathcal{W}$  and the chunk size:  $L_{\mathcal{X}} = \mathcal{W} - \Delta$ . For the Llama2-7B-32K, we noticed that its PPL decreases when the number of tokens within its context window approaches 32K. Therefore, we set its context window size  $\mathcal{W}$  to 24K, which, based on our preliminary experiments, is the optimal context window size we’ve identified. Note that at the beginning of a document, the number of context tokens may be smaller than  $L_{\mathcal{X}}$ . In such cases, we simply take all context as the input. In the generation process, once the token number in the context window reaches the context window size (4K and 24K for different models) we will rebuild and re-compute the KV states taking the  $L_{\mathcal{X}}$  recent tokens as input  $\mathcal{X}$ . When we update the Temp-Lora module, the length  $L_T$  of training input  $\mathcal{X}_T$  is set to 1024.

For the experiment on GuoFeng, each segment is treated as a chunk. In other words, after the model translates a complete segment, this segment is then used to update the Temp-Lora module. We decode the translations with beam search (beam width = 1, repetition penalty = 1.12).

All experiments are done with a single NVIDIA A800 GPU. We set the learning rate  $\alpha = 5 \times 10^{-5}$  and Num. of epochs  $n = 2$  for all models. We use a linear learning rate warmup for the first 2 chunks. We set Temp-Lora  $\alpha = 64$ ,  $rank = 64$ ,

<sup>1</sup>Codes and Data are available at: <https://github.com/TemporaryLoRA/Temp-LoRA/tree/main>

<sup>2</sup><https://huggingface.co/meta-llama/Llama-2-7b>

<sup>3</sup><https://huggingface.co/meta-llama/Llama-2-13b>

<sup>4</sup><https://huggingface.co/togethercomputer/LLaMA-2-7B-32K>

<sup>5</sup><https://huggingface.co/01-ai/Yi-6B-Chat>

Model	$\Delta$	0-100K	100-200K	200-300K	300-400K	400-500K	500K+	Avg.
<b>7B-4K</b>	-	10.47	10.21	10.03	9.24	8.96	4.61	10.19
<b>+TL</b>	1024	10.09(3.6%)	9.55(6.4%)	9.14(8.8%)	8.47 (8.3%)	8.31(7.2%)	<b>4.003(13.2%)</b>	9.58(5.9%)
<b>13B-4K</b>	-	9.49	9.27	9.11	8.49	8.05	4.24	9.25
<b>+TL</b>	1024	9.18(3.3%)	8.47(5.7%)	8.40(7.7%)	7.83(7.7%)	7.55(6.1%)	3.74(11.6%)	8.75(5.3%)
<b>7B-32K</b>	-	10.56	10.25	9.98	9.03	9.17	4.09	10.21
<b>+TL</b>	1024	10.25(2.9%)	9.69(5.4%)	9.21(7.7%)	8.37(7.2%)	8.42(8.2%)	3.73(8.9%)	9.69(5.0%)
<b>+TL + CR</b>	1024	10.26(2.8%)	9.71(5.3%)	9.22(7.5%)	8.39(7.1%)	8.43(8.0%)	3.72(9.0%)	9.71(4.9%)
<b>+TL</b>	2048	10.27(2.7%)	9.72(5.2%)	9.24(7.4%)	8.39(7.0%)	8.44(7.9%)	3.78(7.5%)	9.72(4.8%)
<b>+TL</b>	4096	10.33(2.2%)	9.77(4.6%)	9.31(6.7%)	8.42(6.7%)	8.60(6.2%)	3.91(4.5%)	9.77(4.3%)

Table 1: PPL on text with different context lengths. All tokens in PG19 are divided into six segments based on their position in the novel, i.e., the actual context length of the token. For instance, if a token is the 160K-th token in a novel, it would be categorized into the 100-200K segment. In this table, we report the PPL of various models under different settings in each segment. The percentages in () indicate the relative reduction of PPL of the model compared to the base model. TL and CR are short for Temp-Lora and cache reuse, respectively.

Model	PPL↓	BLEU↑	COMET↑
Yi-6B-Chat	5.67	12.4	72.5
+TL	3.99(-29.6%)	19.0(53.2%)	78.6(8.4%)

Table 2: PPL, BLEU, and COMET of Yi-6B-Chat with and without Temp-Lora.

*dropout* = 0.05. Its training is in bfloat16 format, using Deepspeed ZeRO Stage 2, Flash Attention V2 (Dao, 2023).

### 3.1 Main Results

**PG19:** Table 1 presents the PPL comparisons across various models with and without the Temp-Lora module on PG19. We divide each document into segments ranging from 0-100K to 500K+ tokens, as shown in Table 1. Intuitively, In the initial segments such as 0-100K, where there is less context information, the improvement from Temp-Lora should be relatively modest. In contrast, as the segments grow longer and more information falls outside the model’s context window, Temp-Lora’s effects should become more pronounced.

The experimental results in Table 1 confirm our hypothesis. Firstly, the augmentation of Temp-Lora leads to a significant PPL reduction for all models, where we observe an average decrease of 5.9% on Llama2-7B-4K. An in-depth examination of Temp-Lora’s impact across different text segments reveals it is more notable on long text. For example, Temp-Lora augmented Llama-7B-4K achieves a direct PPL reduction of **13.2%** in the 500K+ segment. In contrast, its effect is relatively less pronounced in the 0-100K segment, where it only reduces the

PPL by 3.6%. We may simply conclude the results as: **with greater text comes greater necessity for Temp-Lora.**

Additionally, adjusting the chunk size from 1024 to 2048 and 4096 resulted in a slight increase in PPL. It is not surprising, as the Temp-Lora module was trained on the data from the previous chunks. An increase in chunk size means that the information stored in the module is more distant from the current token. Indeed, the choice of chunk size is a critical trade-off between generation quality and computational efficiency, which will receive detailed analysis in Section 3.2.

Finally, we also discovered that the cache reuse almost does not cause any loss in performance. This is very encouraging news, as this technique can significantly reduce the computational cost.

**GuoFeng:** Table 2 presents the remarkable impact of the Temp-Lora on the task of discourse-level literary translation. Compared to the base model, there are significant improvements across all metrics: a decrease in PPL by **-29.6%**, an increase in BLEU score by **+53.2%**, and a rise in COMET score by **+8.4%**. This experiment illustrates that the Temp-Lora module is not only effective in downstream tasks and chat-style models, but it is even more effective than in pre-training tasks.

One might wonder why the effects are more pronounced in translation. This is attributed to the differences between tasks: open-domain novel completion is a task without standard answers, making it challenging for even an active human reader of a specific novel to predict the content of the next chapter. In such cases, even if Temp-Lora



$\Delta$	Metric	Base	24K	16K	8K	4K	3K	2K
1024	PPL	10.21	9.69(-5.0%)	9.70(-4.9%)	9.75(-4.5%)	9.82(-3.8%)	9.91(-2.9%)	10.03(-1.7%)
	FLOPs(T)	324.7	380.1(+17.0%)	271.9(-16.2%)	163.6(-49.5%)	109.5(-66.2%)	96.0(-70.4%)	82.4(-74.5%)
	Lat.(s)	52.9	63.1(+19.3%)	46.3(-12.4%)	30.1(-43.0%)	24.9(-53.0%)	25.2(-52.2%)	24.2(-54.1%)
2048	PPL	10.21	9.72(-4.8%)	9.73(-4.7%)	9.79(-4.1%)	9.88(-3.8%)	<b>9.99(-2.1%)</b>	-
	FLOPs(T)	324.7	366.3(+12.8%)	258.0(-20.5%)	149.8(-53.8%)	95.6(-70.5%)	<b>82.1(-74.7%)</b>	-
	Lat.(s)	52.9	63.2(+19.4%)	46.4(-12.2%)	30.1(-42.9%)	25.6(-51.5%)	<b>25.1(-52.4%)</b>	-
4096	PPL	10.21	9.77(-4.3%)	9.79(-4.1%)	9.86(-3.4%)	-	-	-
	FLOPs(T)	324.7	359.3(+10.6%)	251.1(-22.6%)	142.8(-56.0%)	-	-	-
	Lat.(s)	52.9	62.9(+18.8%)	46.2(-12.6%)	30.0(-43.0%)	-	-	-

Table 3: PPL, FLOPs (per 1K tokens), and latency (per 1K tokens) across various context window sizes  $\mathcal{W}$  for Llama2-7B-32K with Temp-Lora and cache reuse. The base model’s context window size is 24K. Tokens are decoded with greedy search. Please note that  $\mathcal{W}$  should be larger or equal to the sum of the chunk size and training input length,  $\Delta + L_T$ . If not, the context window will not be sufficient to include all tokens in Temp-Lora updating. The percentages in () indicate the relative PPL change of the model compared to the base model’s best.

module stores substantial context information, the model can only slightly reduce the PPL of ground-truth text. In contrast, in discourse-level translation, most word translations are unique for consistency. By effectively storing these translation mappings in the Temp-Lora module, substantial improvements can be readily achieved.

### 3.2 Further Analysis

In Section 3.1, we demonstrated the Temp-Lora framework’s ability to enhance long text generation. A key observation is that certain context information is simultaneously stored within both the model’s parameters (Temp-Lora) and its KV cache, leading to an overlap. For the further understanding of the framework’s efficiency, we gradually shorten the context window size  $\mathcal{W}$  during inference to eliminate this overlap. After shortening  $\mathcal{W}$ , we can reduce the computational cost for both models. However, this reduction results in less context information being stored in the KV cache, compelling the model to rely primarily on the Temp-Lora module for accessing contextual information. We measure Llama-7B-32K’s PPL, Floating Point Operations (FLOPs), and latency across these varied context window sizes  $\mathcal{W}$ . This experiment aims to explore the balance between generation quality and computational efficiency in different scenarios.

We observe that models augmented with Temp-Lora consistently surpass the base model’s performance, regardless of the context window size. Notably, even when we reduce the window size to **1/16** of its maximum (2048), a reduction in PPL from 10.21 (base model) to 10.03 is also observed.

It is important to note that as the context window diminishes, there is still a gradual increase in the model’s PPL. As a whole, this trend highlights that Temp-Lora and the KV cache are orthogonal, jointly enhancing the overall performance of the model when used together.

Concerns may arise regarding the additional computational cost associated with updating Temp-Lora. We were surprised to find that in the most “economical” Temp-Lora configuration,  $\Delta = 2K$  and  $\mathcal{W} = 4K$ , the model **not only reduces PPL by 3.8% but also saves 70.5% of FLOPs and 51.5% of latency**. Conversely, if we disregard computational costs entirely, then in the most “luxurious” configuration,  $\Delta = 1K$  and  $\mathcal{W} = 24K$ , we can achieve a 5.0% reduction in PPL with an additional 17% of FLOPs and 19.6% of extra latency.

After summarizing the experimental results, we got some suggestions for applying Temp-Lora in real-world scenarios:

- For applications demanding the highest level of long text generation, integrating Temp-Lora into existing models — without altering any parameters — can significantly enhance performance at a relatively modest cost.
- For applications where minimal latency or memory usage is paramount, computational costs can be significantly lowered by reducing input lengths and storing context information within Temp-Lora. Under this setup, we can process texts of almost infinite length (500K+ in our experiments) using a fixed short window size, such as 2K or 4K.

- Note that in scenarios without extensive text, for example, less than the model’s window size in pretraining, Temp-Lora is useless.

## 4 Conclusion

We proposed Temp-Lora, an alternative way for efficient long text generation. The essence of the Temp-Lora framework lies in training during the inference process using the generated output. It enables the storage of nearly infinite context information directly within the model’s parameters, marking a distinct difference from existing attention weights-based techniques.

Our experimental results across various applications, including language modeling and discourse-level literary translation, demonstrated the profound impact of Temp-Lora. We showed that Temp-Lora not only greatly enhances the quality of long text generation but also significantly reduces computational costs. In particular, Temp-Lora led to remarkable improvements in perplexity (a reduction of 13.1% on a subset of PG19 and 29.6% on a subset of GuoFeng), as well as increases in BLEU (by 53.2%) and COMET scores (by 8.4%). The effectiveness of Temp-Lora becomes increasingly apparent as text length grows. When considering the implementation of Temp-Lora in your applications, bear in mind this guiding principle: **With Greater Text Comes Greater Necessity for Temp-Lora** – a rule that becomes increasingly relevant in the context of extensive texts.

## 5 Applications

For the application of Temp-Lora in different scenarios, we got some experiences to share:

- 1) We arrive at a conclusion similar to the one from [Ovadia et al. \(2023\)](#): Temp-Lora cannot replace RAG ([Li et al., 2022](#)), as training for a limited number of steps is insufficient for the model to clearly remember accurate factual information.
- 2) For role-play, especially when playing specific roles based on user-uploaded documents like Harry Potter ([Chen et al., 2023b](#)), Temp-Lora works wonders: we only need to train a Temp-Lora module on the user-uploaded documents, then the model can fully understand the character’s background and personality.
- 3) Personal assistants like ChatGPT ([OpenAI, 2023](#)): once a dialogue session exceeds the model’s context window size, a Temp-Lora module can be created to store the context knowledge. In most

cases, maintaining the context window size from the pre-training stage is sufficient, eliminating the need for additional context window extension.

4) Game NPCs & Generative Agents ([Park et al., 2023](#)): One significant challenge in such applications is that the model’s context window is not sufficient to record all past events in an NPC’s field of view. With Temp-Lora, we can record such events in the Temp-Lora module through training, instead of a complex memory-summarization-reflection pipeline.

5) Human-Machine Interactive Translation ([Huang et al., 2021](#)): Translations for the same company or project often require a high degree of consistency in translation mappings (e.g., the translations of named entities). We can continuously update the Temp-Lora module with already translated results to help the model remember those translation mappings.

## References

- Guanzheng Chen, Xin Li, Zaiqiao Meng, Shangsong Liang, and Lidong Bing. 2023a. [Clex: Continuous length extrapolation for large language models](#).
- Nuo Chen, Yan Wang, Haiyun Jiang, Deng Cai, Yuhao Li, Ziyang Chen, Longyue Wang, and Jia Li. 2023b. [Large language models meet harry potter: A dataset for aligning dialogue agents with characters](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8506–8520, Singapore. Association for Computational Linguistics.
- Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023c. [Extending context window of large language models via positional interpolation](#).
- Tri Dao. 2023. [Flashattention-2: Faster attention with better parallelism and work partitioning](#).
- Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. [Flashattention: Fast and memory-efficient exact attention with io-awareness](#).
- Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2023. [Lm-infinite: Simple on-the-fly length generalization for large language models](#).
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. [Lora: Low-rank adaptation of large language models](#).
- Guoping Huang, Lemao Liu, Xing Wang, Longyue Wang, Huayang Li, Zhaopeng Tu, Chengyan Huang, and Shuming Shi. 2021. [Transmart: A practical interactive machine translation system](#).

- Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. 2022. [A survey on retrieval-augmented text generation](#).
- OpenAI. 2023. [Gpt-4 technical report](#).
- Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2023. [Fine-tuning or retrieval? comparing knowledge injection in llms](#).
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. [Bleu: a method for automatic evaluation of machine translation](#). In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.
- Joon Sung Park, Joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. [Generative agents: Interactive simulacra of human behavior](#).
- Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. [Yarn: Efficient context window extension of large language models](#).
- Ofir Press, Noah A. Smith, and Mike Lewis. 2021. [Train short, test long: Attention with linear biases enables input length extrapolation](#). *CoRR*, abs/2108.12409.
- Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. 2019. [Compressive transformers for long-range sequence modelling](#). *arXiv preprint*.
- Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. [COMET: A neural framework for MT evaluation](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.
- Shuming Shi, Enbo Zhao, Duyu Tang, Yan Wang, Piji Li, Wei Bi, Haiyun Jiang, Guoping Huang, Leyang Cui, Xinting Huang, Cong Zhou, Yong Dai, and Dongyang Ma. 2022. [Effidit: Your ai writing assistant](#).
- Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. [Roformer: Enhanced transformer with rotary position embedding](#).
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. [Llama: Open and efficient foundation language models](#).
- Longyue Wang, Zefeng Du, DongHuai Liu, Deng Cai, Dian Yu, Haiyun Jiang, Yan Wang, Shuming Shi, and Zhaopeng Tu. 2023a. [Guofeng: A discourse-aware evaluation benchmark for language understanding, translation and generation](#).
- Longyue Wang, Zhaopeng Tu, Yan Gu, Siyou Liu, Dian Yu, Qingsong Ma, Chenyang Lyu, Liting Zhou, Chao-Hong Liu, Yufeng Ma, Weiyu Chen, Yvette Graham, Bonnie Webber, Philipp Koehn, Andy Way, Yulin Yuan, and Shuming Shi. 2023b. [Findings of the WMT 2023 shared task on discourse-level literary translation: A fresh orb in the cosmos of LLMs](#). In *Proceedings of the Eighth Conference on Machine Translation*, pages 55–67, Singapore. Association for Computational Linguistics.