


	<p>Journal of Artificial Intelligence General Science (JAIGS)</p> <p>ISSN: 3006-4023 (Online), Volume 5, Issue 1,2024 DOI: 10.60087</p> <p>Home page https://ojs.boulibrary.com/index.php/JAIGS</p>	
---	--	---

CoWPE: Adaptive Context Window Adjustment in LLMs for Complex Input Queries

Venkata Mohit Tamanampudi

Independent Researcher, USA

ABSTRACT

Recent work has shown that large language models, or LLMs, are capable of amazing processing context windows based on the nuance and complexity of respective input queries. By changing rotary position embedding (RoPE), a well-liked position encoding technique used by well-known LLMs like LLaMA and GPT-NeoX, recent studies have attempted to expand the context window of LLMs. In order to help LLMs efficiently adapt to a larger context window based on input query complexity and nuance, we identify in this work the inherent need for LLMs' attention entropy (i.e., the information entropy of attention scores) to maintain stability and introduce a novel extension to RoPE that combines adjusting RoPE's base frequency and scaling the attention logits. Our proposal, CoWPE, aims to accomplish this by building neighbor attention information and bi-level grouped attention in order to modify the context window of LLMs. While neighbor attention catches relationships between neighboring tokens within a given range, grouped attention collects interdependence among tokens that are far apart. During inference, the self-attention mechanism of the original model is utilized to calculate the two-level attentions. Our CoWPE requires no fine-tuning and can easily expand the context window of existing LLMs with a small amount of code adjustment. We carry out extensive tests on several benchmarks, and the outcomes demonstrate the CoWPE can successfully increase the context window duration of current LLMs.

Keywords: Large Language Models, LLMs, RoPE, Context Window, CoWPE, Llama, LLM training, Input Query, Complexity, Nuance, CoWPE

ARTICLE INFO: *Received:* 01.07.2024 *Accepted:* 07.08.2024 *Published:* 07.08.2024

© The Author(s) 2024. Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0>

1. Introduction

The quantity of text data that a language model can process at once while producing answers is referred to as a context window (Swimm Team, 2023). The term is used to refer to the maximum quantity of information which can be considered by the model at one time during the formulation of answers. For example, if the language model has 8000 tokens in its context window, then it implies that it can only analyze and generate a text base that only reaches 8000 tokens input at once. This is important since it is used to determine the extent at which a model can incorporate preceding text as a way of ensuring that its output is coherent. It contains every token found in the input text that the model examines in order to obtain context before responding. This window's size has a direct impact on the model's comprehension and response precision, therefore, if the window context size is large, the it will be able to contain a vast volume of text. Different language models process and interpret vast volumes of text differently because of differences in the size of the context window. A window that is too big could cause the model to analyze unnecessary information, whereas one that is too tiny could cause it to overlook important context. This implies that an appropriate size should be considered to have a precise analysis of the information.

The most data that an LLM may accept as query input is known as its context length. This includes both the input provided by the user and any prior text the model has generated within the same session. Context length essentially defines how much textual information the model can retain and utilize to produce coherent and contextually relevant answers. Put differently, a user can enter more information into a prompt to get a response if the context window is longer. Token length is really used by language models to measure content. A token is then defined as four letters (in English) or $\frac{3}{4}$ of a word; hence, 75 words are equal to 100 tokens. Tokens can be words, parts of words, or even individual characters, depending on the tokenization method used. The term "Word Tokenization" is also known as word-based tokenization, and this is a token that might correspond to a single word, such as "apple" or "computer." In light of this, the context lengths of a few of the most well-known LLMs are listed here.

LLM Name	Context Window Length
Llama	2K
Llama 2	4K
GPT-3.5-turbo	4K
GPT-4	8K
Mistral 7B	8K
Palm-2	8K
Claude	9K
Google Gemini	32K
Claude 2	100,00K

Table 1: LLMs context window lengths

For instance, In the context of the LLaMA (Large Language Model Meta AI) model, a "2K token" refers to the model's ability to process up to 2,000 tokens of text at once, a 4K context window from Llama 2 is equal to six

pages, and a 32K context length is equal to 49 pages. Since most contemporary LLMs are trained with a fixed duration of training sequences, their context window length is constrained (Zhao et al., 2023; Yang et al., 2023). The context window length at the pre-training phase determines it. When the pretraining context window is exceeded by the length of the input texts during inference, LLM behavior becomes unpredictable and performance degrades significantly. Long input sequences will cause the model's perplexity (PPL) to expand (Xiao et al., 2023; Peng et al., 2023; Han et al., 2023; Chen et al., 2023b). To increase the context window of pretrained LLMs, numerous techniques for doing so have been devised recently. These methods aim to extend the amount of text a model can consider at once, enhancing its ability to handle longer or more complex inputs effectively. Sparse Attention Mechanisms can be computationally expensive with large context windows. Sparse attention mechanisms, such as those used in models like Longformer, focus on a subset of attention heads or specific regions of the text, reducing the overall computational load. Hierarchical attention divides the text into smaller chunks and processes each chunk separately before aggregating the results. This approach breaks down the problem of handling long sequences into more manageable pieces (Chen et al., 2023b). Memory-Augmented Models use external memory mechanisms to store and retrieve information from previous contexts. Techniques such as memory networks or differentiable neural computers allow models to access information beyond the immediate context window. Chunking and Sliding Window involves breaking the input text into overlapping chunks or windows (Chen et al., 2023b). Each chunk is processed independently, and information from overlapping regions helps maintain continuity. Recurrent architectures, such as those used in transformers with recurrence or recursion, process sequences in a step-by-step manner, allowing for longer-term dependencies (Chen et al., 2023b). Newer variations of transformers, such as Linformer or Reformer, use different mechanisms to reduce the complexity of self-attention calculations. Attention-augmented models, such as those incorporating global and local attention mechanisms, combine both detailed and broad context processing to manage longer texts. A simple method would be to adjust these models on sufficient long texts. In addition, several techniques aim to increase context window length through more effective fine-tuning techniques. The terms "PI" (Chen et al., 2023b), "CLEX" (Chen et al., 2023a), "Yarn" (Peng et al., 2023), "PoSE" (Zhu et al., 2023), "LongLora" (Chen et al., 2023c), and "ABF" (Xiong et al., 2023) are a few of the prominent strategies among these modern approaches. These techniques seek to increase the content window on the presumption that pretrained LLMs are incapable of handling lengthy text. However, because of the quadratic complexity of Transformers, these methods usually need to be fine-tuned in order to achieve extension, which can be time- and resource-intensive. Furthermore, the lack of high-quality long text data makes these fine-tuning techniques difficult. Real-world data is typically brief, while lengthier texts frequently lack significant long-range dependencies. It is not because LLMs are unable of grasping extended contexts that they perform poorly when faced with lengthy texts. According to our investigation, the main obstacle preventing LLMs from managing lengthier contexts effectively is the positional O.O.D issue, also known as the Out-of-Distribution (O.O.D) issues associated with positional encoding. This issue occurs when LLMs are exposed to new relative distances that were not present during their pretraining phase during text sequences encountered during inference that are longer than the pretraining context window. It is well known that when dealing with O.O.D. inputs, neural networks (NNs) can exhibit unpredictable behaviors (Liu et al., 2021; Shen et al., 2021; Bai et al., 2021; Zhang et al., 2023).

1

Remapping the unseen relative positions to those encountered during pretraining would be a logical and workable solution to this problem, expanding the LLMs' natural capacity to accommodate lengthier contexts. CoWPE maps unseen big relative positions to those encountered during pretraining using a straightforward floor division technique. Natural language demonstrates the property that meaning is largely constant throughout small ranges, such as paragraphs. Consequently, the relative ordering of crucial information is captured effectively when near or even identical position encodings are used. This methodical technique perfectly complements the

functionality of the floor operation. Furthermore, this same idea is also shared by T5 (Raffel et al., 2020) and iRPE (Wu et al., 2021).

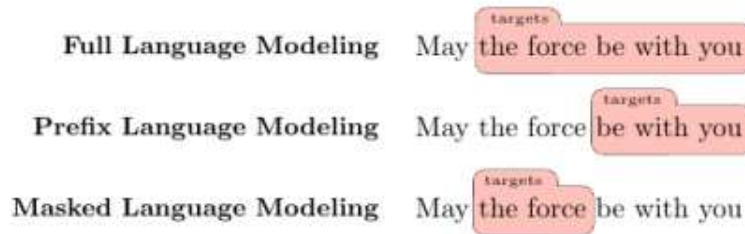


Figure 1: LLM's context window mapping

Our suggestions can be summarized as below:

- By mapping the unobserved large relative locations at inference to known positions at training, we propose CoWPE, which aims to enlarge the context window of LLMs without any fine-tuning. LLMs are able to preserve coherence in complicated context windows as a result.
- Even if they haven't seen exceptionally long texts throughout training, LLMs with RoPE are naturally adept at handling lengthy materials.

2. Related Work

The increasing demand for machines to handle complicated context windows, such as translation, summarization, information retrieval, conversational exchanges, etc., is the reason for the necessity for generalized models. Significant advances in language models have been observed recently, mainly due to the availability of large-scale training data, enhanced computational capabilities, and transformers (Vaswani et al., 2017). By enabling the development of LLMs that can mimic human-level performance on a variety of tasks, these advancements have brought about a fundamental transformation (Adiwanrdan et al., 2020). Modern artificial intelligence systems known as large language models (LLMs) are able to comprehend and produce text with cohesive communication, as well as generalize to multiple tasks (Radford et al., 2019). In the past, statistical language modeling gave way to neural language modeling, which in turn led to pre-trained language models (PLMs) and finally to LLMs in natural language processing (NLP). PLMs are trained in a self-supervised context on a large corpus of text (Chang et al., 2018) with the goal of learning a generic representation that is shareable among diverse NLP tasks, in contrast to conventional language modeling (LM), which trains task-specific models in supervised settings. PLMs outperform classical language modeling (LM) in terms of performance increases after being adjusted for downstream workloads. Because larger PLMs yield greater performance advantages, PLMs are being replaced with LLMs by drastically raising model parameters.

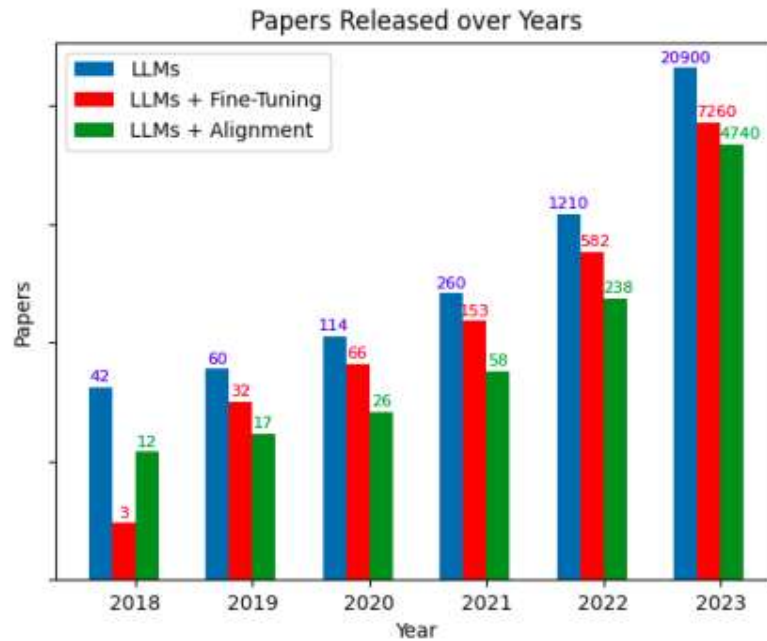


Figure 2: The trend of papers released over years containing keywords "Large Language Model", "Large Language Model + Context window", and "Large Language Model + Input Query complexity and nuance"

When given work descriptions and examples, LLMs respond to task requests with accuracy. Pre-trained LLMs, however, perform worse in zero-shot situations than in few-shot scenarios and are unable to understand human intent. The improvement of generalization to unseen tasks is achieved by fine-tuning them with task instructions data (Chung et al., 2022) and aligning with human preferences (Touvron et al., 2023). This leads to a considerable improvement in zero-shot performance and a decrease in misaligned behavior. Numerous LLMs have been put out in the literature in response to this development (Zhang et al., 2022). Figures 1 and 2 illustrate an increasing trend in the quantity of LLMs that have been released and the names of a few notable LLMs that have been suggested over time. Early LLM research used models like T5 and mT5.

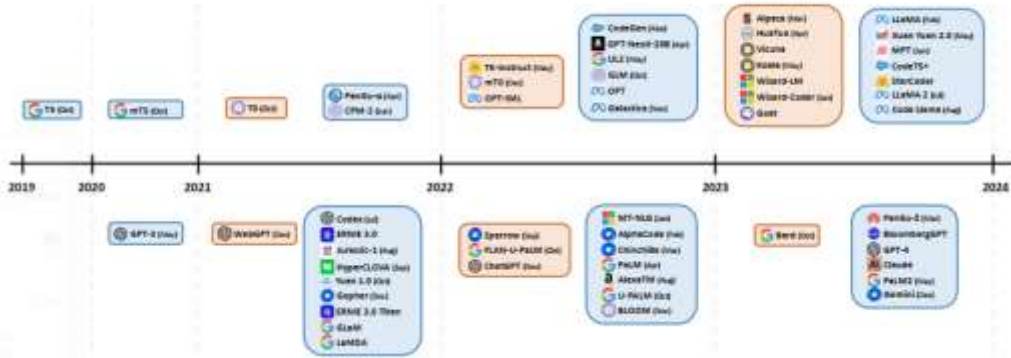


Figure 3: LLM releases are shown chronologically, with orange cards corresponding to "instruction-tuned" models and blue cards representing "pre-trained" models. Models with an upper half indicate open-source availability, whereas models with a lower half indicate closed-source availability.

3. Preliminary

3.0. Context Window Position Encoding (CoWPE).

Context window encoding enriches the LLM context window with additional data that adjusts the word's inference based on the input query complexity in the model's prompt sentence. As a result, the pattern can be identified based on the word order even in cases where two sentences include the same term. The "position encoding model" is used to determine the context window's position hence creating a distinct vector for every position. Rectangles are used to symbolize these. The word embedding and the encoding vector space have the same size, therefore they go together. The resultant vectors are passed into the Transformer and are now rich in positional data. After then, the data is processed by the Transformer using a multi-layered feed network to generate the output. Large language models (LLMs) based on transformers.

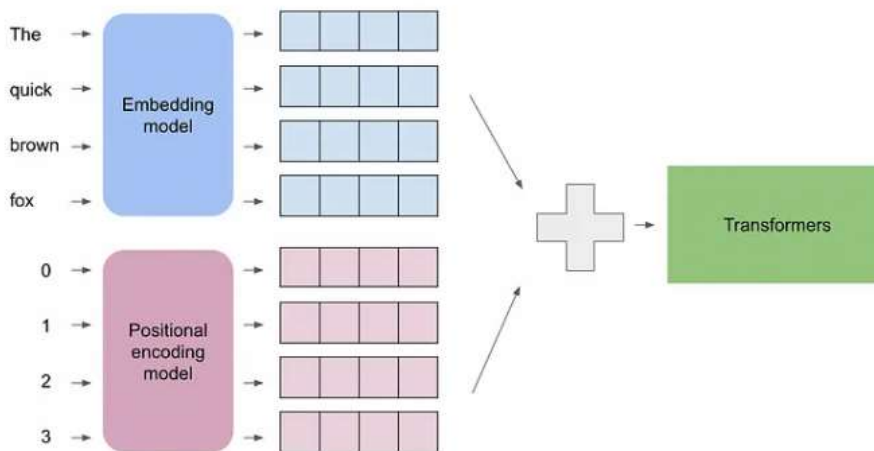


Figure 4: Context Window Position Encoding Adjustment

In most LLM models, context windows can be adjusted by first deciding which tokens should be included when measuring distance using their context vectors. To do that, a context window adjustment value is computed for every complex and nuance input query q_i and nuance k_j ;

$$g_{ij} = \sigma(\mathbf{q}_i^T \mathbf{k}_j) \tag{1}$$

j is less than i and σ is our sigmoid function. An adjustment value of 1 could imply that the context window will be adjusted in the measured position while a value of 0 means it will be ignored. The adjustment is premised on the complexity and nuance of the input query.

$$P_{ij} = \sum_{k=j}^i g_{ik}$$

Next, the context window position adjustments are calculated by adding the adjustment value between the current input query position and the targeted prompt token. Suppose the adjustments are always 1, then $(p_{ij} = i - j + 1)$ and we recover tokens based on the input query's positions. As a result, CoWPE can be thought of as relative PE expanded. But generally speaking, p_{ij} can represent the number of characters, the count of certain context window character kinds like nouns or numerals, or other ideas the Transformer thinks helpful during training.

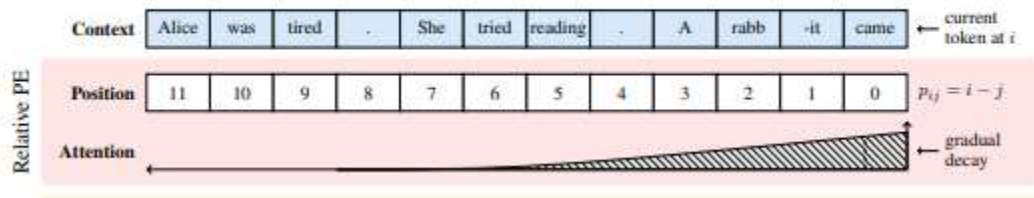


Figure 5: CoWPE

3.1. Adjustment Window Visualization.

The sine and cosine functions are combined with the encoding function to produce different signals. Due to the selection of sine and cosine powers, the model can generate long sequences that are not encountered during training, providing a way to encode location information that is easy for the model to understand. This can be translated to a corresponding mathematical formula

$$f(x) = \exp(x * c) \tag{3}$$

whereby x indicates the adjustment location along the context window dimension of $d_{model}/2$ and c refers to a negative constant $-\text{math.log}(10000.0) / d_{model}$. The corresponding plot can be seen as indicated in the figure below. The various patterns observed in the plot above ensures that every location in the context window has a distinct encoding during adjustment, which aids the model in differentiating between positions.

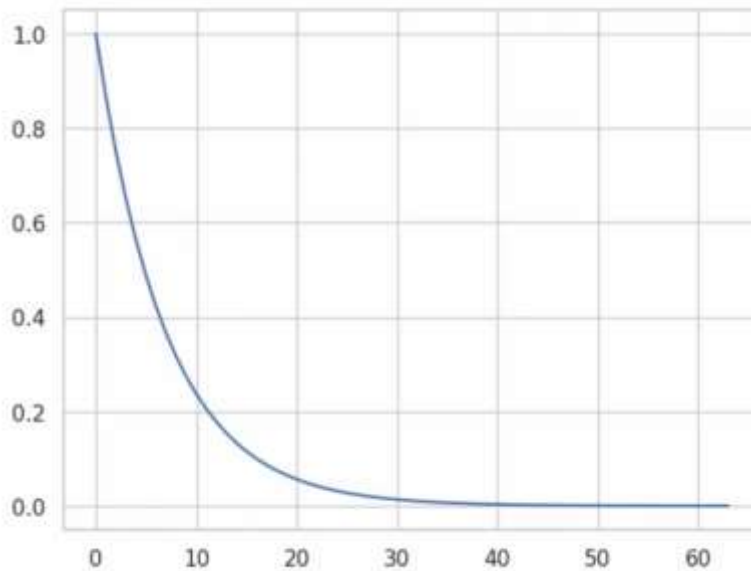


Figure 6: Adjustment context window visualization

4. Flip-Flop LLMs adjustment model

With reference to the LLMs context window flip flop adjustments, we look into the LLMs' innate capacity to adjust its context windows based on the complexity of their input queries. Next, we the model's ability to efficiently adapts the context window of current LLMs to the intricacy and subtlety of input questions without requiring any fine-tuning.

4.0 Preliminary Analysis

The Flip-Flop LLMs context window adjustment is cited to highlight the inability of Transformer models to capture characters in large context windows (Liu et al., 2024). The input query are made up of a series of alternating instructions (ignore, write, and read; *iwr*), each instruction is followed by a single bit of data, 0 or 1, which the model must memorize if it follows *w*, or recall the last memory if it follows *r*. Every input query string will undoubtedly begin with *w* and end with *r*. For instance, given the string the anticipated output is 1 because the previous write operation is followed by 1. The model must be able to focus intently on the most recent instance of the *w* in order to do this task. symbol, whose location changes between sequences as a result of disregarding instructions. In-distribution and out-of-distribution (OOD) test sets are defined by the job, with the latter increasing the distance to the final write with an increase in the quantity of ignore command.

For models with dimension 256, 4 heads, and 4 layers, we duplicate the setup outlined in Liu et al. (2024) and report test error after 10K training steps. Table 2 (left) presents the findings. They demonstrate how CoWPE performs better than current techniques, enabling the model to learn not only the in-distribution job but also to generalize to OOD sequences, which is a capability that current PE techniques are unable to offer. This is made feasible by CoWPE, which enables the model to use the keys of particular tokens to adjust their counts into the positional embedding, thereby turning on the adjustment function for those tokens and attending to their last-seen positions. For instance, position 1 will correspond if the adjustments are 1 solely on write tokens.

Table 2: Context Window adjustment based on complexity

PE method	Flip-Flop Context Window Adjustment		
	in-domain (wpass=50)	OOD longer context Window (wpass=100)	OOD shorter context Window (wpass=20)
Relative PE	2.1	10.3	35
CoWPE	0.2	0	8

5. LLMs Adjustment Modelling

Our dataset, the Wikitext-103 dataset (Merity et al., 2017), which comprises 100M tokens taken from Wikipedia, to evaluate our approach on a language modeling job. We train a Transformer model with 15 layers and a hidden size of about 800, matching the design of GPT-2 (Radford et al., 2019). Using a batch size of 65, we train using the negative log-likelihood loss for 100 epochs. Although the maximum position value in CoWPE is set to a far smaller value of p , the model has a context size of 1024. In Table 5, we compare various context window position encoding techniques. The worst performing is Absolute PE. CoWPE performs better than relative PE and gets much better when added to relative PE. This demonstrates that, even in language modeling in general Context Window position encoding helps a lot. Refer to the table below:

Wikitext-103 and Code		
PE Method	Params (M)	Test PPL
RoPE	18.9	5.1
CoWPE	26	4.3
CoWPE + RoPE	19.3	7

Table 3: Context Window encoding comparisons

CoWPE can direct attention to abstract items within the context window based on the analysis. We demonstrate how location alone on Wikitext-103 can draw attention. Because CoWPE is contextualized, context window characters can be adjusted according to their complexity. As seen in the plots in Figure 7, the segments are discovered to be split on the left by newline tokens (shown by black plus signs), and separated on the right by section names such as " = Description = " (marked in a similar manner).

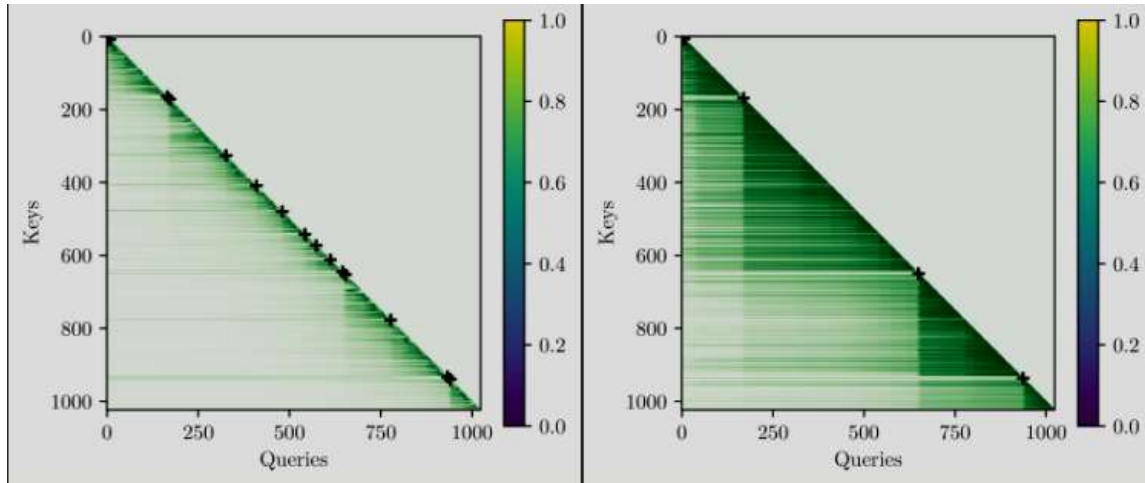


Figure 7: Adjustments based on Input query complexity

CoWPE can be used to adjust context window positions to a significantly greater number of tokens since it bases its assignment of positions on context window. The number of adjustments will differ for every sample, even though the number of tokens was set during training. As a result, tokens that are outside of T 's training span may nevertheless receive position values that fall inside of p_{max} , the maximum value. On the other hand, each token position is associated with an embedding in relative PE. As a result, those tokens won't have any position embedding added to them when there are $T' - T$ unobserved positions throughout test time. We evaluate a variation of relative PE where unseen positions leverage the embedding problem. Longer context window adjustments result in poorer adjustments in relative PE. In comparison, the relative-capped variant performs far better. Still, CoWPE performs better than it, and the difference grows when the context window adjustment is substantially longer than the training environment as seen in the graphical representations in Figure 8 below:

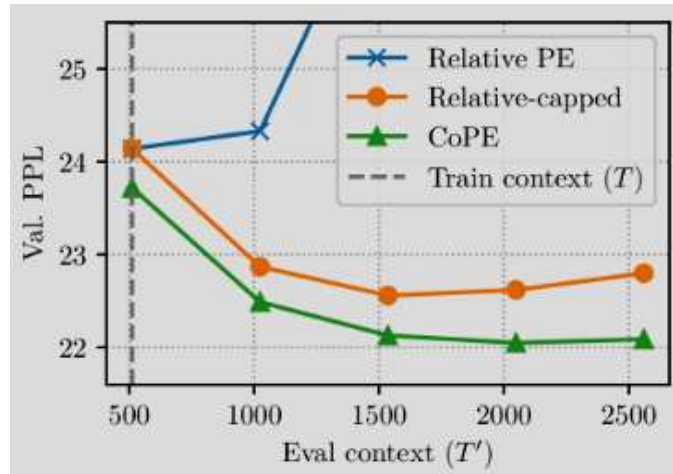


Figure 8: Context window adjustment plot

6. Conclusion

LLMs with big context windows usually lack the accuracy and reliability of smaller sizes, even while they offer more possibilities, boost efficiency, and can be applied to a higher range of use situations. There's going to be a lot of effort. Thankfully, LLM development is still in its early stages, and a large number of academics and producers of AI solutions are dedicated to discovering workable solutions, and they are moving in that direction at a pace that is encouraging.

In this research, we contend that LLMs possess an innate capacity to manage lengthy sequences, and we suggest utilizing CoWPE as an LLM context window adjustment technique that departs from the token-based position paradigm by adjusting context window position based on input query nuance and complexity. This method works well on many activities and gives you more flexibility when addressing people based on their position. Although the scope of this research was limited to the text and code domains, CoWPE could be extended to other domains, such as speech and video, where token position appears even less appropriate intuitively. Using CoWPE to train larger models and track their performance on downstream tasks is an additional approach worth investigating.

7. References

- Adiwardana, D. M.-T. (2020) Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, et al., Towards a humanlike open-domain chatbot. *arXiv preprint arXiv:2001.09977* (2020).
- Bai et al. (2021) Bai, T., Luo, J., Zhao, J., Wen, B., and Wang, Q. Recent advances in adversarial training for adversarial robustness. *arXiv preprint arXiv:2102.01356*, 2021.
- Brown et al. (2020) Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- Chen et al. (2023c) Chen, Y., Qian, S., Tang, H., Lai, X., Liu, Z., Han, S., and Jia, J. Longlora: Efficient fine-tuning of long-context large language models *arXiv preprint arXiv:2309.12307*, 2023c.

- Chen et al. (2023b) Chen, S., Wong, S., Chen, L., and Tian, Y. Extending context window of large language models via positional interpolation *arXiv preprint arXiv:2306.15595*, 2023b.
- Chuang et al. (2024) Chuang, Y.-N., Xing, T., Chang, C.-Y., Liu, Z., Chen, X., and Hu, X. Learning to compress prompt in natural language formats. *arXiv preprint arXiv:2402.18700*, 2024.
- Chung, H. W. L. (2022) Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, et al., Scaling instruction-finetuned language models, *arXiv preprint arXiv:2210.11416* (2022)
- Dai et al. (2019) Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.
- Dao et al. (2022) Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.
- Han et al. (2023) Han, C., Wang, Q., Xiong, W., Chen, Y., Ji, H., and Wang, S. Lm-infinite: Simple on-the-fly length generalization for large language models, *arXiv preprint arXiv:2308.16137*, 2023.
- Jiang et al. (2023b) Jiang, H., Wu, Q., , Luo, X., Li, D., Lin, C.-Y., Yang, Y., and Qiu, L. LongLLMLingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *ArXiv preprint*, abs/2310.06839, 2023b. URL <https://arxiv.org/abs/2310.06839>. *arXiv:2109.07958*, 2021.
- Liu et al. (2021) Liu, J., Shen, Z., He, Y., Zhang, X., Xu, R., Yu, H., and Cui, P. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021.
- Liu et al. (2024) Liu, Z., Yuan, J., Jin, H., Zhong, S., Xu, Z., Braverman, V., Chen, B., and Hu, X. Kivi: A tuning-free asymmetric 2bit quantization for. kv cache. *arXiv preprint arXiv:2402.02750*, 2024.
- Press et al. (2021) Press, O., Smith, N. A., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.
- Rae et al. (2019) Rae, J. W., Potapenko, A., Jayakumar, S. M., and Lillicrap, T. P. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.
- Raffel et al. (2020) Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.
- Radford, A. J. (2019) Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, *OpenAI blog* 1 (8) (2019) 9.
- Shen et al. (2021) Shen, Z., Liu, J., He, Y., Zhang, X., Xu, R., Yu, H., and Cui, P. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021.
- Swimm Team. (2023). *Swimm.io*. [swimm.io | https://swimm.io/learn/large-language-models/llm-context-windows-basics-examples-and-prompting-best-practices](https://swimm.io/learn/large-language-models/llm-context-windows-basics-examples-and-prompting-best-practices)
- Touvron et al. (2023) Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- Vaswani et al. (2017) Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

- Wu et al. (2021) Wu, K., Peng, H., Chen, M., Fu, J., and Chao, H. Rethinking and improving relative position encoding for vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10033–10041, 2021.
- Xiao et al. (2023) Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.
- Xiong et al. (2023) Xiong, W., Liu, J., Molybog, I., Zhang, H., Bhargava, P., Hou, R., Martin, L., Rungta, R., Sankararaman, K. A., Oguz, B., et al. Effective long-context scaling of foundation models. *arXiv preprint arXiv:2309.16039*, 2023.
- Xue et al. (2020) Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., and Raffel, C. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.
- Yang et al. (2023) Yang, J., Jin, H., Tang, R., Han, X., Feng, Q., Jiang, H., Yin, B., and Hu, X. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *arXiv preprint arXiv:2304.13712*, 2023.
- Zhang et al. (2022) Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.
- Zhao et al. (2023) Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.
- Zhu et al. (2023) Zhu, D., Yang, N., Wang, L., Song, Y., Wu, W., Wei, F., and Li, S. Pose: Efficient context window extension of llms via positional skip-wise training. *arXiv preprint arXiv:2309.10400*, 2023.