CAMELoT: Towards Large Language Models with Training-Free Consolidated Associative Memory

Zexue He [1 2] **Leonid Karlinsky** [2] **Donghyun Kim** [3] **Julian McAuley** [1] **Dmitry Krotov** [2] **Rogerio Feris** [2]

Abstract

Large Language Models (LLMs) struggle with long input sequences due to high memory and runtime costs. Memory-augmented models offer a promising solution to this problem, but existing methods have limited memory capacity and require costly re-training to integrate with the LLM. In this work, we introduce CAMELoT, a **C**onsolidated **A**ssociative **M**emory **E**nhanced **Lo**ng **T**ransformer, which has an associative memory (AM) module integrated with any pre-trained attention-based LLM. The AM module in CAMELoT consolidates token representations into a non-parametric distribution model, balancing novelty and recency, therefore giving the LLM the capability to process the long input sequences without any re-training. By retrieving information from AM, CAMELoT achieves a significant perplexity reduction in long-context modeling benchmarks, e.g., 29.7% on Arxiv, even with a tiny context window of 128 tokens.

1. Introduction

Humans' memory systems can processed and consolidated events overtime, forming groups of related events that guide future actions by retaining essential information and discarding inessential details (Sara, 2000). Associative Memory (AM) is a key type of human-like memory system that links (associates) a query with stored representations (Willshaw et al., 1969; Hopfield, 1982). For any query, AM identifies the memory slot with the best matching representation. These representations summarize past experiences and guide future actions. Recently, there has been growing interest in designing modern associative memory networks (Krotov & Hopfield, 2016; Ramsauer et al., 2021). Significant literature exists on memory consolidation in neural networks (Dudai, 2004) and local learning rules, which are more computationally efficient than end-to-end backpropagation (Tyulmankov et al., 2021).

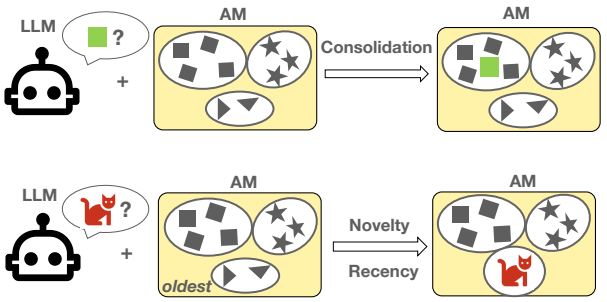

Figure 1. **C**onsolidated **A**ssociative **M**emory **E**nhanced **Lo**ng **T**ransformer (CAMELoT). Top: Consolidation of representations in the associative memory (AM) – related concepts are grouped together and averaged. Bottom: Recency-dependent incorporation of novel concepts – when a new concept is introduced with no close matches, the oldest slot (since its last update) is replaced.

Concurrently, large language models (LLMs) have become very important for many practical applications such as chatbots, text generation (Radford et al., 2019), and question answering (Chung et al., 2022), etc. A key parameter for LLMs is the input context length L that the models are trained with. Supporting longer context makes it possible to increase the performance by incorporating richer information (Press et al., 2022). However, extending the context length of state-of-the-art LLMs is challenging due to substantial resources requirements, e.g., the complexity of the conventional attention mechanism in LLMs scales quadratically (L^2) with the number of tokens.

These constraints raise a question: *can we develop a plug-and-play module for pre-trained (frozen) LLMs to handle (unlimited) long contexts beyond L?* Ideally, this module should be computationally efficient and *not* require retraining or fine-tuning of the LLM.

Inspired by AM, we propose a module that consolidates token representations into memory based on novelty and

[1]Computer Science and Engineering, University of California San Diego, La Jolla CA, USA [2]MIT-IBM Watson AI Lab, IBM Research, Cambridge MA, USA [3]Department of Artificial Intelligence, Korea University, Seoul, South Korea. Correspondence to: Zexue He <zehe@ucsd.edu>.

Proceedings of the 1st Workshop on Long-Context Foundation Models, Vienna, Austria. 2024. Copyright 2024 by the author(s).

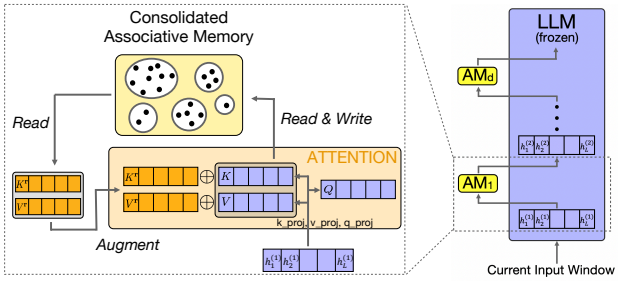

Figure 2. The general pipeline of CAMELoT. Every layer of the backbone LLM is augmented with an AM module (we draw AM in the first attention layer here, just as an example). Keys and values are calculated for every token, keys are used to search for relevant memorized tokens in the memory bank and return them (Read). The retrieved memory keys and values are prepended to the original token keys and values as prefixes. Finally, the attention operation is applied on the concatenation of the retrieved and native keys and values (Augment). After retrieval, the memory state is modified according to the Write operation.

recency of input *concepts*. As shown in Figure 1, when modeling a input sequence, similar information is consolidated together, using a computationally cheap local writing rule, whereas the outdated one is discarded. As shown in Figure 2, the consolidated context is modeled as non-parametric distributions, one per key-space of each LLM layer. These distributions are dynamically updated as the context window moves, with new modes created for novel information and outdated ones replaced. Long-context attention is approximated by retrieving modes closest to the current context hidden states and adding them as a key-value cache. This module can be integrated with any pre-trained attention-based LLM, extending its context window beyond L by approximating a full-context attention over all the past.

Our method requires no retraining, fine-tuning, or adaptors between the LLM and the AM module. We conduct comprehensive experiments over the long-context language modeling tasks, which demonstrates significant improvements compared to the baselines. For example, when coupled with a pre-trained LLaMA model, our memory-enhanced network achieves up to 29.7% perplexity reduction in long-context modeling on Arxiv compared to the base LLM.

2. Associative Memory Enabled LLM

For long document modeling, efficiently using past context information is crucial. Our model is built on three desiderata: (1) **consolidation**: redundant past information should be compressed into a single memory slot; (2) **novelty**: new concepts should be detected and stored in a new memory slot upon first encounter; (3) **recency**: outdated memory slots should be discarded when the topic shifts to accommodate new concepts. To achieve these desiderata, we equiped the memory module in CAMELoT with a **Read** and **Write** operations, supporting information retrieval from the memory bank and the update to the memory bank. With the retrieved information, the current context window of LLM is memory-enhanced via the **Augment** operation. Our method is agnostic to the specific choice of many popular transformer architectures, in the sense that any attention-based LLM can be enhanced with the AM in CAMELoT.

2.1. Read Operation

When a context window of length L is processed through the LLM, keys and values from every layer (more generally can be an arbitrary subset of layers) are passed to the corresponding AM module (one per memory-augmented layer). AM in each layer consists of M memory slots, enumerated by the index $\mu = 1, ..., M$. Each slot contains two vector variables: memory keys $K_\mu^{\mathbf{mem}}$ and memory values $V_\mu^{\mathbf{mem}}$, and two integer scalar variables: counts c_μ (number of consolidated instances), and age τ_μ (how old the current slot is since its last update).

When a set of keys K_i and values V_i (index $i = 1, ..., L$ enumerates individual tokens from the current context window) is passed to the AM module to retrieve relevant information, a *search function* identifies the memory slots with the strongest association (highest similarity) between the input token key K_i and AM's memory slot keys $\{K_\mu^{\mathbf{mem}}\}$:

$$\hat{\mu}(i) = \underset{\mu}{argmax}\big[sim(K_\mu^{\mathbf{mem}}, K_i)\big] \tag{1}$$

The keys and their corresponding values of these L strongest-associated memories ($K^{\mathbf{r}}$ and $V^{\mathbf{r}}$) are returned for the current L native tokens and passed back to the LLM in the form of the key-value cache.

2.2. Augment Operation

The list of retrieved key-value caches ($K^{\mathbf{r}}$ and $V^{\mathbf{r}}$) are passed back to the base LLM and used as the prefix context in each respective memory-augmented layer. They are prepended to the LLM keys and values of current input tokens. Then causal attention is performed on the concatenated list, which after the augmentation contains $2L$ keys and values (the length of current native context + the length of retrieved memories) and L queries (current context only), resulting in the augmented transformer attention output $[a_1, \cdots, a_L]$. The attention output results in augmented hidden states $[h_1, \cdots, h_L]$ which are the input to the next

layer, as shown in the following equations:

$$[a_1, \cdots, a_L] = Attn(Q, K', V') \quad (2)$$

$$Q = [Q_1, Q_2, \cdots, Q_L] \quad (3)$$

$$K' = K^{\mathbf{r}} \oplus [K_1, \cdot, K_L], \quad (4)$$

$$V' = V^{\mathbf{r}} \oplus [V_1, \cdots, V_L] \quad (5)$$

2.3. Write Operation

The state of AM is updated by the current context window according to the **Write** operation which has two parts.

Consolidation. If the similarity between the current context token key and the strongest-associated memorized key is large ($> R$, R is a hyper-parameter), the concept described by that token is declared familiar and, for this reason, its key and value are consolidated with the key and value stored in that memory slot. Specifically, memory slots are updated according to:

$$K^{\mathbf{mem}}_{\hat{\mu}(i)} \leftarrow \frac{K_i + c_{\hat{\mu}(i)} K^{\mathbf{mem}}_{\hat{\mu}(i)}}{c_{\hat{\mu}(i)} + 1} \quad (6)$$

$$V^{\mathbf{mem}}_{\hat{\mu}(i)} \leftarrow \frac{V_i + c_{\hat{\mu}(i)} V^{\mathbf{mem}}_{\hat{\mu}(i)}}{c_{\hat{\mu}(i)} + 1} \quad (7)$$

$$c_{\hat{\mu}(i)} \leftarrow c_{\hat{\mu}(i)} + 1 \quad (8)$$

where c_μ tracks the number of instances consolidated in slot μ. Thus, the consolidated representations stored in each slot μ are always arithmetic averages of individual instances that went into that slot.

Novelty and Recency. If the similarity with the closest memorized key is weak ($< R$), the concept is declared novel. In this case, the oldest unused memory slot (the one with maximal age τ_μ) is replaced with K_i, V_i, and its age is set to 0. After each slot $\hat{\mu}(i)$ update its age $\tau_{\hat{\mu}(i)}$ is set to 0, the ages of all slots that had no matching current context hidden state are incremented by 1.

3. AM-augmented Long Language Modeling

Datasets We evaluate the long-context language modeling capabilities of CAMELoT using three standard datasets. The test perplexity is reported on each of the datasets: **Wiki-103** (Merity et al., 2016)[1], which comprises articles from Wikipedia covering various topics with good language quality; **Arxiv** (Gao et al., 2020)[2], a collection of academic papers in Mathematics, Computer Science, and Physics; and **PG-19** (Rae et al., 2019)[3], which includes full-length books (Wu et al., 2022; Wang et al., 2023; Tworkowski et al., 2023).

[1]https://blog.salesforceairesearch.com/the-wikitext-long-term-dependency-language-modeling-dataset/

[2]Taken from the Pile: https://pile.eleuther.ai/

[3]https://github.com/google-deepmind/pg19

Baselines. We compare CAMELoT against two notable memory-augmented transformers in long language modeling tasks: **Transformer-XL** (Dai et al., 2019), a finetuning-based approach which stores a fixed length of previous input in a cache to enhance the current input without any similarity-based retrieval; **Memorizing Transformer** (Wu et al., 2022) a finetuning-based model saving past caches in a circular manner, where older caches are replaced by newer ones as the memory bank fills up (no consolidation occurs).

For a fair comparison, in CAMELoT and the baselines experiments, we used the same LLaMa2-7B backbone (original baselines used weaker backbones, such as GPT2), and did not use fine-tuning. More implementation details are shown in Appendix A.1. We also provide ablation study results in Appendix A.2.

3.1. Results

Table 1 compares CAMELoT with the baseline models. While memory-augmented methods generally improve upon the base model on test perplexity, our analysis uncovers the following observations. Transformer-XL shows the least improvement, hindered by the lack of relevance assessment during memory augmentation. The Memorizing Transformer, with its capability to selectively retrieve relevant information from the past, outperforms Transformer-XL. However, it lacks memory consolidation, meaning it can only hold a finite cache before older memories are overwritten, limiting its long-term utility. By not only selecting relevant past information but also employing a novel memory consolidation process, CAMELoT significantly enhances model performance (16.6% on PG-19, and 29.7% on Arxiv, and 6.36% on wikitext-103, relative to the base model on average), surpassing other memory-augmented methods.

3.2. Discussion

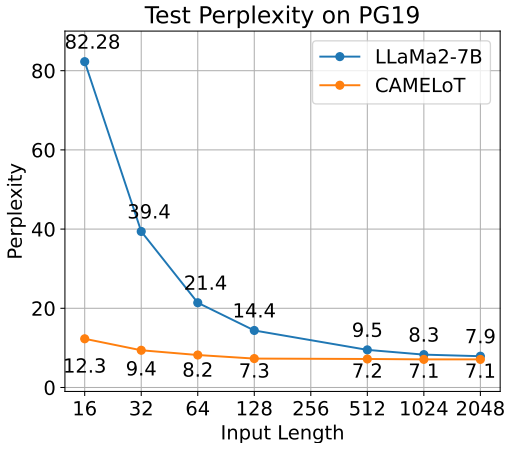

Figure 3. Test perplexity on PG19 with different input lengths.

	PG-19 and Arxiv				wikitext-103		
	Input Length	Retrieved Mem.	PG-19	Arxiv	Input Length	Retrieved Mem.	Wikitext-103
LLaMa2-7B	512	None	9.54	5.99	512	None	16.0
	1024	None	8.33	4.98	1024	None	14.80
	2048	None	7.88	4.35	2048	None	14.46
	Avg	-	8.58	5.12	Avg	-	15.09
Transformer-XL	512	512	8.44	4.15	256	256	15.02
	1024	1024	8.27	3.81	512	512	14.21
	2048	2048	7.86	3.65	1024	1024	14.2
	Avg	-	8.19	3.87	Avg	-	14.48
Memorizing Transformers	512	512	8.12	3.82	256	256	14.18
	1024	1024	7.4	3.63	512	512	14.07
	2048	2048	7.34	3.62	1024	1024	14.39
	Avg	-	7.62	3.69	Avg	-	14.21
CAMELoT	512	512	7.24	3.61	256	256	14.06
	1024	1024	7.14	**3.60**	512	512	**14.00**
	2048	2048	**7.10**	**3.60**	1024	1024	14.34
	Avg	-	7.16	3.60	Avg	-	14.13

Table 1. Language Modeling Perplexity on wikitext-103, Arxiv, and Pg-19. For wikitext-103, we notice the maximum length of its documents is smaller than 2k. Therefore, we report results of models whose effective input length $\leq$ 2048 (i.e., input length $\leq$ 2048 for non-augmented model; and input length $\leq$ 1024 for memory-augmented models). **Bold**: Best perplexity on each dataset. Avg: Average.

Shorter Inputs, Better Performance Figure 3 shows CAMELoT's performance with different input lengths on PG-19 test set, with 10k memory slots. Unlike models without memory augmentation, CAMELoT demonstrates a relatively consistent performance across different input lengths. This stability can be attributed to the integration of additional knowledge in the AM saved from previous inputs. As CAMELoT accumulates past information, its visible context range extends beyond the current input, allowing an effective modeling of long-range dependencies irrespective of the length of the current input. In contrast, the model lacking memory augmentation relies solely on the local context of the current input, leading to performance fluctuations based on input length.

CAMELoT maintains its effectiveness even with tiny input lengths (e.g., 128), reducing the demand on hardware resources such as large GPUs. This enables transformers to operate attention with shorter inputs but without compromising the quality of language modeling. Such an advantage lowers the barriers for deploying large language models in environments where computational budget is limited.

4. Related Works

Long-range self-attention is a line of work that tackles long-context modeling. It includes low-rank factorization (Wang et al., 2020), dilated attention (Ding et al., 2023), sparsity (Beltagy et al., 2020; Zaheer et al., 2020; Kitaev et al., 2020), and FlashAttention (Dao et al., 2022; Dao, 2023). These methods struggle to retrieve information in the middle of the input (Liu et al., 2023). At the same time, they can be used in tandem with our proposed approach for long context modeling.

Memory-augmented LLMs is another stream of work aiming at modelling the extended context window (Dai et al., 2019; Wu et al., 2022; Tworkowski et al., 2023; Weston et al., 2014). For example, Wu et al. (2022) save static past (key, value) pairs of input into cache bank and use KNN retrieval to improve language modeling. Tworkowski et al. (2023) further improve this approach with contrastive learning. Wang et al. (2023) tackle the memory staleness of these models by training a side network. Unlike these methods, our approach uses dynamical consolidations of past tokens, compressing redundant information from the past input into a memory of fixed size.

5. Conclusion

We have introduced CAMELoT, **C**onsolidated **A**ssociative **M**emory **E**nhanced **Lo**ng **T**ransformer, for long dependency modeling without the need for training. CAMELoT has a model-agnostic design, allowing seamless integration into different language models. Experimental results prove its effectiveness, with the long-context language modeling perplexity significantly reduced (by up to 29.7%), and superior performance is consistently obtained even with a tiny input

window of 128 tokens. Future research directions connecting AM and LLMs involve improving the AM design (e.g., automatically learning a Write function) or tackling other long context modeling tasks (e.g., long document summarization or advanced reasoning).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, which are common in studies of LLMs, none which we feel must be specifically highlighted here. Not contradictory, we recommend the language models to be audited properly when deployed in real-world applications such as chatbots where real user information can be the input of the language models.

References

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. Scaling instruction-finetuned language models, 2022.

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, T. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

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.

Ding, J., Ma, S., Dong, L., Zhang, X., Huang, S., Wang, W., Zheng, N., and Wei, F. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*, 2023.

Dudai, Y. The neurobiology of consolidations, or, how stable is the engram? *Annu. Rev. Psychol.*, 55:51–86, 2004.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Krotov, D. and Hopfield, J. J. Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29, 2016.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models, 2016.

Press, O., Smith, N. A., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation, 2022.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rae, J. W., Potapenko, A., Jayakumar, S. M., Hillier, C., and Lillicrap, T. P. Compressive transformers for long-range sequence modelling. *arXiv preprint*, 2019. URL https://arxiv.org/abs/1911.05507.

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Gruber, L., Holzleitner, M., Adler, T., Kreil, D., Kopp, M. K., et al. Hopfield networks is all you need. In *International Conference on Learning Representations*, 2021.

Sara, S. J. Retrieval and reconsolidation: toward a neurobiology of remembering. *Learning & memory*, 7(2):73–84, 2000.

Tworkowski, S., Staniszewski, K., Pacek, M., Wu, Y., Michalewski, H., and Miłoś, P. Focused transformer: Contrastive training for context scaling. *arXiv preprint arXiv:2307.03170*, 2023.

Tyulmankov, D., Fang, C., Vadaparty, A., and Yang, G. R. Biological learning in key-value memory networks. *Advances in Neural Information Processing Systems*, 34: 22247–22258, 2021.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Wang, W., Dong, L., Cheng, H., Liu, X., Yan, X., Gao, J., and Wei, F. Augmenting language models with long-term memory. *arXiv preprint arXiv:2306.07174*, 2023.

Weston, J., Chopra, S., and Bordes, A. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

Willshaw, D. J., Buneman, O. P., and Longuet-Higgins, H. C. Non-holographic associative memory. *Nature*, 222(5197): 960–962, 1969.

Wu, Y., Rabe, M. N., Hutchins, D., and Szegedy, C. Memorizing transformers. *arXiv preprint arXiv:2203.08913*, 2022.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

A. Appendix

A.1. Experiment Details

Environments All transformers-based languague models are implemented based on the HuggingFace[4] libraries (version 4.34.0) or the officially released Github Repos. All codes are implemented with Python 3.10.12 and PyTorch 2.2.0 with CUDA 12.1.0. We run experiments with 2 NVIDIA A100 GPUs, one for language model inference and one for hosting the memory banks. Each has memory of 80GB.

Hyper-parameters In long-context language modeling tasks, we set batch size to be 4. For the similarity hyper-parameter R, we conduct a hyper-parameter study on wikitext-103 and use $R = 0.93$ for all experiments.

We show the study of R in the following: We conduct hyper-parameter study for similarity threshold R on a subset of wikitext-103 validation set, in which the examples are randomly sampled. We take LLaMa2 models with input length to be 128. The results are shown in Table 2.

From the results, we notice the best R is within $[0.9, 0.95)$. Therefore we use 0.93 in our experiments.

R	Perplexity
R=0.1	PPL=18.72
R=0.2	PPL=18.71
R=0.3	PPL=18.71
R=0.4	PPL=18.69
R=0.5	PPL=18.67
R=0.6	PPL=18.46
R=0.7	PPL=17.35
R=0.8	PPL=15.30
R=0.9	PPL=**14.38**
R=0.95	PPL=14.90

Table 2. Hyper-parameter study for R on the validation subset of wikitext-103

A.2. More Ablation Studies

A.2.1. Ablation: Effectiveness of Each Component in CAMELoT

To assess the impact of each component within CAMELoT, we define the following ablation variants:

CAMELoT w/o Retrieval: Instead of retrieving the closest matching memory concept for each token in the current input, a random memory concept is returned.
CAMELoT w/o Recency: If a token's mode has no close match in memory, it randomly replaces a memory slot rather than the outdated one, ignoring recency.
CAMELoT w/o Novelty. Tokens are consolidated into their closet slot, regardless of if they are from novel modes. R=-1 in cosine similarity retrieval.
CAMELoT w/o Consolidating. Memory gets updated by token representations based on temporal recency, without consolidating, setting R=+1.

We evaluate on PG-19 Sampled dataset, a subset of PG-19 comprising 20% of the books in test set. We report test perplexity for each variant with a context length of 2048.

Results shown in Table 3 reveal that CAMELoT w/o Read performs significantly worse compared with full model, emphasizing the crucial role of Read function in ensuring semantic relevance. When a random cache is returned in this variant, it might provide limited or even harmful information for current modeling. CAMELoT w/o Recency also shows a notable performance dip over the full CAMELoT model, confirming the essential role of maintaining the proper recency in the memory. Variations in token consolidation and replacement also impact performance, resulting in different performance

[4] https://huggingface.co/models

Models	PPL
LLaMa2-7B	7.30
CAMELoT	**6.85**
CAMELoT w/o Read	> 20
CAMELoT w/o Recency	9.25
CAMELoT w/o Novelty	7.23
CAMELoT w/o Consolidation	7.00

Table 3. Ablation Study on PG-19-sampled. We report the relative performance lost in perplexity over the full CAMELoT.

	Perplexity
CAMELoT + Cosine Similarity	16.96
CAMELoT + Euclidean Similarity	**17.45**

Table 4. Analysis of similarity function on a subset of wikitext-103 validation set.

drops compared to the full approach. A larger decrement can be expected if the memory size gets smaller or the modeling corpus gets longer. These findings suggest CAMELoT's optimal performance relies on the combination of relevance, recency, novelty, and effective consolidation.

A.2.2. Ablation: Different Choices of Similarity Function in Read Operation

We conduct an ablation study on the similarity function in Read operation. Similarly, we randomly sampled a subset data from the validation set of wikitext-103. We conduct evaluation experiments with cosine similarity and euclidean similarity. We use input window with 128 tokens. Note in this experiment we use LLaMa2-7B. The results are shown in Table 4. We notice cosine similarity gives the best performance and we use cosine similarity in our other experiments.

A.2.3. Ablation: Different Memory Sizes

Model	Input Context	Memory Size	Perplexity
LLaMa2-7B	512	None	9.84
	2048	None	7.88
CAMELoT	512	4096	7.42
	2048	4096	7.22
	512	10k	7.24
	2048	10k	**7.10**

Table 5. Language Modeling performance on PG19 with different sizes of memory banks and different input lengths.

In this section, we analyze how the size of the memory affects CAMELoT. We compare its performance on the PG-19 dataset using two configurations: one with 4,096 memory slots and another with 10k slots. The findings are presented in Table 5.

With each memory slot designed to hold a unique mode of information, increasing the number of slots allows CAMELoT to capture a wider range of knowledge. As a result, the version with 10k slots outperforms, showing a notable improvement in test perplexity – 26.4% for inputs of 512 length and 9.9% for 2048 length relative to the base model.

However, the 4,096-slot configuration also performs strongly, with only slightly lower improvements (24.6% and 8.4%, for the same input lengths) than CAMELoT with 10k slots. This good performance demonstrates that the effectiveness of CAMELoT does not solely rely on the quantity of data modes it can hold in its memory, but also on how it manages and utilizes this data through mechanisms like consolidation and novelty. This balance ensures CAMELoT remains effective across various memory sizes and input lengths, maintaining stability and efficiency.