🎩 SirLLM: Streaming Infinite Retentive LLM

Anonymous ACL submission

Abstract

As Large Language Models (LLMs) become increasingly prevalent in various domains, their ability to process inputs of any length and maintain a degree of memory becomes essential. However, the one-off input of overly long texts is limited, as studies have shown that when input lengths exceed the LLMs' pre-trained text length, there is a dramatic decline in text generation capabilities. Moreover, simply extending the length of pre-training texts is impractical due to the difficulty in obtaining long text data and the substantial memory consumption costs this would entail for LLMs. Recent efforts have employed streaming inputs to alleviate the pressure of excessively long text inputs, but this approach can significantly impair the model's long-term memory capabilities.

Motivated by this challenge, we introduce Streaming Infinite Retentive LLM (SirLLM), which allows LLMs to maintain longer memory during infinite-length dialogues without the need for fine-tuning. SirLLM utilizes the Token Entropy metric and a memory decay mechanism to filter key phrases, endowing LLMs with both long-lasting and flexible memory. We designed three distinct tasks and constructed three datasets to measure the effectiveness of SirLLM from various angles: (1) DailyDialog; (2) Grocery Shopping; (3) Rock-Paper-Scissors. Our experimental results robustly demonstrate that SirLLM can achieve stable and significant improvements across different LLMs and tasks, compellingly proving its effectiveness. When having a coversation, "A sir could forget himself," but SirLLM never does!

1 Introduction

The proliferation of large language models (LLMs) (Touvron et al., 2023a; Achiam et al., 2023; Jiang et al., 2023) has spurred the development of various NLP applications, including widely-used tools like chatbots (Bill and Eriksson, 2023; Pandya and Holia, 2023), writing assistants (Bhat et al., 2023), and programming assistants (Kazemitabaar et al., 2024). These applications, aiming to enhance user interaction and conversational experience, often require infinite input length and a certain degree of memory capability. However, current LLMs are usually pre-trained on texts of limited length, and studies have shown that their text generation capabilities dramatically decline when input lengths exceed those of the pre-training texts (Xiao et al., 2023; Huang et al., 2023). Merely extending the length of pre-training texts is impractical, as acquiring infinitely long text data is exceedingly challenging, not to mention that it would result in substantial memory consumption for LLMs. Therefore, researching how to enable LLMs to handle infinite input lengths while maintaining memory capability is an urgent issue to be addressed.

With the emergence of this demand, researchers have gradually shifted their focus towards exploring ways to expand the input context length of LLMs. A line of these studies has particularly focused on optimizing the attention mechanism of LLMs. (Beltagy et al., 2020) first proposes the Sliding-window attention, as shown in Figure 1 (a). By restricting each token to only attend to a certain number of recent tokens, this method reduces computational complexity. In deployment scenarios, LLMs utilize a Key-Value (KV) cache to store the key and value tensors of past tokens at each generation step to effectively reduces the need to recompute past key and value tensors, thereby significantly lowering computational overhead. Consequently, Sliding-window attention ensures a stable decoding speed even when the KV cache is full, thereby allowing for longer texts during the pre-training phase. However, Xiao et al. (2023) discovered that this method does not truly achieve infinite input length, as the model's performance significantly deteriorates once the input length exceeds the size of the KV cache and intial tokens, however, receive a disproportionately higher amount of

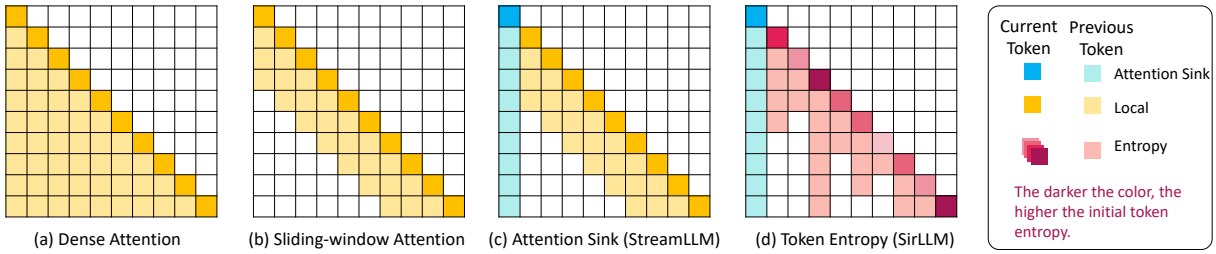


Figure 1: The visualization of SirLLM versus existing attention patterns.

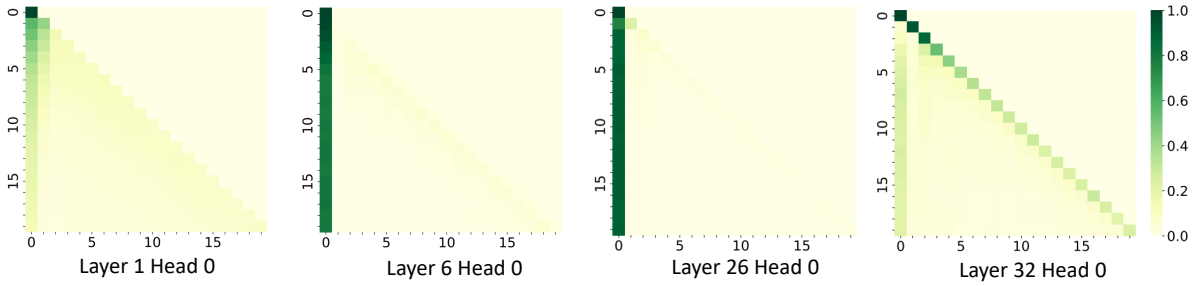


Figure 2: Attention sink phenomenon (Xiao et al., 2023). We visualize the average layer attention logits over 256 sentences, each with a length of 20, in Vicuna-7b-v1.3. We can see that in the shallow layers, a significant amount of the attention score is dedicated to the first tokens and in the final layer, the model focuses more on the recent tokens.

attention, a phenomenon termed as 'attention sink', as shown in Figure 2. Therefore, they proposed StreamLLM, as shown in Figure 1 (b). StreamLLM enhances the potential of window attention by preserving the KV cache of the initial tokens, thereby achieving infinite length input in streaming conversations without finetuning. However, while Sliding-window Attention and StreamLLM ensure an expanded input length, each generated token only attends to recent tokens (and initial attention sink tokens), resulting in a loss of memory for earlier parts of the conversation. This leads to a significant forgetting issue in long-distance dialogues. Furthermore, as observed in Figure 2, the range of recent tokens that the model focuses on is not very extensive. This observation leads us to contemplate **whether it's possible for the model to concentrate only on key terms during a conversation, filtering out less important tokens**. By remembering only the crucial information, the model might be able to maintain a longer memory span in the context of infinitely long conversations. More detailed related works please refer to Appendix A.

In response to the aforementioned challenges, we propose the Streaming Infinite Retentive LLM (SirLLM) in this paper, as illustrated in Figure 1 (d). Initially, we employ an LLM to calculate the token entropy metric for each input token, thereby assessing their significance. Subsequently, tokens with higher token entropy values, deemed as key tokens, are preserved within the KV cache. This method enhances the model's memory capabilities in the context of infinitely long streaming dialogues. To validate the effectiveness of SirLLM, we conducted experiments across three distinct tasks: (1) DailyDialog: We created a multi-turn daily dialogue dataset based on the DailyDialog dataset (Li et al., 2017a). (2) Grocery Shopping: We developed a grocery shopping dataset. Users first inform the LLM about the groceries they need to purchase. Following this, users engage in multi-turn dialogues with the LLM, culminating in the users asking the LLM to recall the required groceries. (3) Rock-Paper-Scissors: We constructed a rock-paper-scissors dataset featuring three types of players, each with a preference for one of the three moves (rock, paper, scissors). Players engage in multiple rounds of rock-paper-scissors with the LLM, which is tasked with analyzing the user's historical preferences to maximize its winning rate. The results of these experiments effectively demonstrate the enhanced memory capabilities of SirLLM in infinite conversation.

2 Method

2.1 Preliminaries

Xiao et al. (2023) proposed StreamLLM. They dis-

covered that the model disproportionately focuses on initial tokens and break when removing initial tokens' KV cache. Therefore, based on the Sliding-window attention, instead of throwing away all of the previous KV cache except the recent token's KV cache, they keep the first initial tokens KV cache as shown in Figure 1 (c). Figure 1 (c) illustrates the StreamLLM process, which can be formulated as follows. We define the indices of the attention sink tokens and the recent tokens as Id_{sink} and Id_{recent}, respectively:

$$\mathrm{Id}_{sink} = \{0, 1, ..., n_{sink}\}$$

$$\mathrm{Id}_{recent} = \{L - n_{recent} + 1, ..., l - 1, l\}$$

where, n_{sink} and n_{recent} denotes the KV cache size of the attention sink tokens and recent tokens respectively. l denotes the total length of the past key-value states.

Then the StreamLLM only keeps the selected tokens' past key and value states:

$$\mathrm{KV}_{cache} = \mathrm{K}_{cache}[\mathrm{Id}_{sink}, \mathrm{Id}_{recent}]$$

where $\mathrm{X}[\mathrm{Id}]$ indicates the selection of vectors from X using indices in using indices in Id.

However, StreamLLM primarily focuses on recent tokens and the initial attention sink tokens. This raises an intriguing question: Could we conserve cache space occupied by recent tokens by only preserving the past key-value states of critical tokens? Such an approach would allow the model to access information from tokens over a longer time span, potentially enhancing its long-term memory and reducing the problem of forgetting. To address this issue, our first step is to define a metric that can accurately measure the importance of each token.

2.2 Token Entropy

Recent work (Li et al., 2023) has focused on context compression. This involves utilizing LLMs to calculate the information contained in each token, thereby compressing the input context to enhance the model's inference efficiency. Inspired by this, we have designed a token entropy metric to assess the significance of tokens. Given a input sequence $X = \{x_1, x_2, ..., x_n\}$, where x_i denotes i-th token and n denotes the total token number. We define the token entropy of the i-th token as:

$$e_i = -\mathrm{logP}(x_i|x_0, x_1, ..., x_{i-1})$$

A token with higher token entropy indicates that it contains more information and is therefore more critical. In our experiments, we utilize the LLM to calculate the generation probability of each token. This approach allows us to obtain the entropy of each token concurrently with its generation, without necessitating additional computational effort.

Does higher token entropy equate to increased LLM focus? To investigate whether tokens with higher entropy indeed carry more information and consequently garner more attention from LLMs, we extracted 256 sentences from the Wikitext corpus (Merity et al., 2017), focusing on the first 40 tokens of each sentence. To mitigate the attention sink effect, we omitted the first token, starting our analysis from the second token, thus providing a clearer view of the model's attention distribution across other tokens. The 40 tokens were divided into four segments based on token entropy, with segment 1 having the lowest entropy and segment 4 the highest. We calculated the average attention logits for each segment at every layer and plotted these values in a scatter plot, as shown in Figure 3. For a more tangible understanding, we also computed the average attention logits across all layers for each segment. The results show that tokens in segments with higher entropy have higher attention logits. This pattern reinforces the hypothesis that higher entropy tokens, which are presumably less predictable and therefore more informative, are given priority by the LLM's attention mechanism. This finding supports the validity of the token entropy metric as an indicator of a token's significance.

2.3 Streaming Infinite Retentive LLM

Upon obtaining the entropy values for each token, we enhance the model's memory capability by selectively preserving the key-value states of only the key tokens and propose Streaming Infinite Retentive LLM (SirLLM) as shown in Figure 4. To elaborate further, we maintain both a key-value (KV) cache KV_{cache} and a token entropy cache E in parallel. The token entropy cache stores the entropy values of tokens present in the KV cache. When the number of tokens stored in C exceeds the pre-training length L, we utilize E to select the tokens with the higher token entropy, thereby conserving space in the KV cache:

$$E = \{e_1, e_2, ..., e_l\}; \quad \mathrm{Id}_{entropy} = \mathrm{Top}_k(E)$$

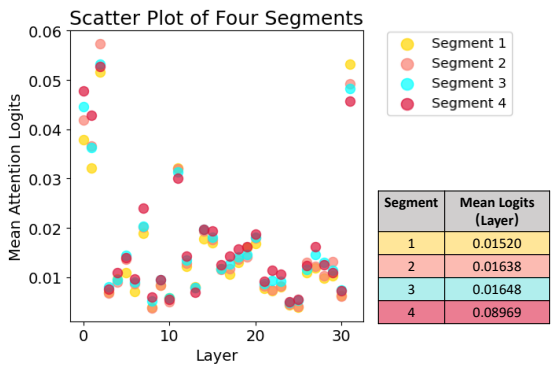


Figure 3: Scatter Plot of the average attention logits over 256 sentences at every layer. We divide the tokens into four segments based on token entropy, with segment 1 having the lowest entropy and segment 4 the highest. The figure indicates that as token entropy increases, so does the attention that the LLM allocates to that token.

$$\mathrm{KV}_{cache} = \mathrm{KV}_{cache}[\mathrm{Id}_{sink}, \mathrm{Id}_{entropy}]$$

$$E = E[\mathrm{Id}_{sink}, \mathrm{Id}_{entropy}]$$

where Top_k denotes the selection of the top k tokens with the highest token entropy. Higher token entropy implies a lower probability of the model generating the word, indicating such words carry more information and are likely to be key tokens.

Following StreamLLM, SirLLM concentrates on the token positions within the cache rather than their original positions in the text when determining relative distances and injecting positional information. For instance, if the current cache holds tokens [0, 1, 2, 3, 5, 7, 11, 12] and the model is in the process of decoding the 13th token, it assigns positions as [0, 1, 2, 3, 4, 5, 6, 7] instead of using the original text positions.

However, simply preserving tokens with the highest token entropy, as previously described, can lead to a limitation in the KV cache. After lengthy multi-turn dialogues with users, the cache may become restricted to a few tokens with very high entropy, making it difficult for the cache to adapt. This could result in a 'rigid memory' within the model, lacking flexibility. An effective dialogue system should, like human memory, have a more flexible mechanism for long and short-term memory: the more distant the memory, the easier it is for the model to forget it. This approach ensures the freshness of the LLM's memory, thereby enhancing the user's conversational experience. To address this, we propose using a decay ratio η_{decay} less than 1. After each round of dialogue, the stored entropy cache E is multiplied by this decay ratio $E = E \times \eta_{decay}$, allowing the model to naturally forget older key information and focus more on recent critical information. The overall process of SirLLM can be referred to in Algorithm 1.

Algorithm 1 Streaming Infinite Retentive LLM

Input: i-th turn's user input $I_i = \{x_1, x_2, ..., x_n\}$
Output: i-th turn's system response $R_i = \{r_1, r_2, ..., r_m\}$
for turn t in range(i) **do**
  if KV cache size >L **then**
    $\mathrm{Id}_{\mathrm{entropy}} = \mathrm{Top}_{\mathrm{k}}(\mathrm{E})$
    $\mathrm{KV}_{cache} \leftarrow \mathrm{KV}_{cache}[\mathrm{Id}_{\mathrm{sink}}, \mathrm{Id}_{\mathrm{entropy}}]$
    $E \leftarrow E[\mathrm{Id}_{\mathrm{sink}}, \mathrm{Id}_{\mathrm{entropy}}]$
  end if
  $R_t, \mathrm{KV}_{cache} = LLM(\mathrm{KV}_{cache}, I_t)$
  $E_t = Entropy([I_t, R_t]) = \{e_1, e_2, ..., e_{n+m}\}$
  $E \leftarrow E + E_t$
  $E = E \times \eta_{decay}$
end for

3 Experiments

3.1 Experimental Setup

We tested SirLLM on two different categories of large models: Vicuna-13b-v1.3, Vicuna-7b-v1.3 (Zheng et al., 2023), Yi-34B-Chat, Yi-6B-Chat [1]. Following the evaluation methodologies used in (Brown et al., 2020; Touvron et al., 2023b; Gao et al., 2023), we evaluate the performance of SirLLM on various datasets by appending different option letters to the answers. We then calculate the logits for each option and select the option with the highest logits as the final answer. All experiments were conducted on an NVIDIA A800 GPU.

3.1.1 Baslines

To comprehensively evaluate the effectiveness of SirLLM, we utilized three baseline models:

StreamLLM: StreamLLM (Xiao et al., 2023) preserves the key-value states of only the attention sink tokens and recent tokens.

RandomLLM: RandomLLM maintains the key-value states of the attention sink tokens as well as a random selection of historical tokens.

IntervalLLM: Taking inspiration from (Child et al., 2019), we developed IntervalLLM. This

[1] https://github.com/01-ai/Yi

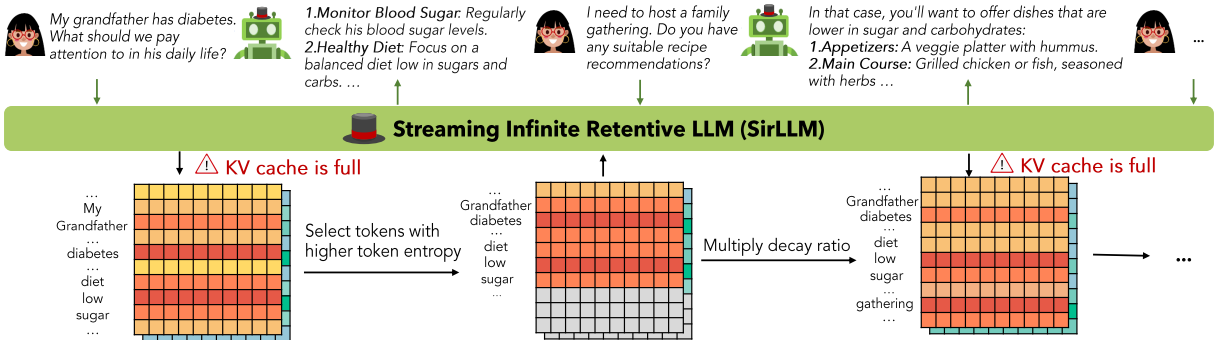


Figure 4: Framework overview of SirLLM. When the number of tokens stored in KV cache exceeds the pre-training length L, SirLLM calculates the entropy of each token and selects the tokens with the higher token entropy, thereby conserving space in the KV cache

model, in addition to preserving attention sink tokens, uniformly samples tokens from the historical token sequence at fixed intervals. These intervals are adaptively determined, The size of these intervals is adaptively determined, calculated as interval $= \lfloor \frac{\text{history token length}}{\text{cache size}} \rfloor$. This approach continues until the cache size is fully utilized

To ensure a fair comparison, in line with StreamLLM, all models preserve the KV cache states of attention sink tokens with a uniform size of 4 and we report the average accuracy for RandomLLM

3.2 Results

To thoroughly validate the effectiveness of the SirLLM framework, we designed three distinct tasks, each assessing SirLLM from a different perspective: (1) DailyDialog: This task evaluates SirLLM's conversational coherence and memory capabilities in everyday multi-turn dialogue scenarios. (2) Grocery Shopping: In this task, we focus on assessing SirLLM's memory capabilities. Initially, the LLM is informed about the groceries to be purchased. Subsequent rounds of commonsense QA with the LLM are conducted, culminating in a query to ascertain if SirLLM remembers the required groceries. (3) Rock-Paper-Scissors: In this task, by engaging in multiple rounds of rock-paper-scissors with users having distinct throwing preferences, we test whether SirLLM can utilize its enhanced memory ability to analyze historical information, discern users' throwing preferences, and thereby maximize its winning probability.

3.2.1 DailyDialog

Dataset Construction To assess the performance of SirLLM in everyday dialogue scenarios, we evaluate SirLLM using the DailyDialog dataset (Li et al., 2017b). DailyDialog is a high-quality, multi-turn, open-domain English dialogue dataset. To measure SirLLM's effectiveness more intuitively, we have reformatted DailyDialog into a multiple-choice question format, where SirLLM is tasked with selecting the most appropriate response. We have selected a sample from the constructed DailyDialog dataset, as illustrated in Figure 7 in Appendix D. For more detailed statistics and construction process about the modified dataset, please refer to Table 5 in Appendix B. From the Table 5, we observe that the average number of tokens per turn in the modified DailyDialog dataset is approximately 461.55. Therefore, we have set the cache size to 512. It was found that 199 dialogs in the dataset have token counts exceeding 512. In such longer dialog scenarios, SirLLM can be highly effective. By enabling the LLM to remember only key tokens, SirLLM is endowed with a longer memory span. This capability allows it to engage more effectively in extended dialogues.

Results In the table 1, to ensure a fair comparison, each model is configured with a unified KV cache size of 512. Table 1 displays the performance of various models on the DailyDialog dataset. It is evident that SirLLM demonstrates a clear advantage over three baseline models across four different LLMs. It is noteworthy that SirLLM's performance remains consistently stable, whereas RandomLLM and IntervalLLM sometimes even lead to performance degradation. When employing Yi-34b, SirLLM achieved the highest accuracy of 90.35% on the modified DailyDialog dataset, marking an impressive 5.00% increase in accuracy compared to StreamLLM. These results robustly demonstrate SirLLM's capability to enhance the memory ability of LLMs, providing them with a longer retention span and thereby offering users a

	# Entropy	# Recent	η_{decay}	ACC(%)	Δ
Yi-6b *Attention Sink Size: 4*					
Stream	0	508	1	76.90	
Random	508	0	1	71.10	-5.80
Interval	508	0	1	65.20	-11.70
SirLLM	508	0	0.7	**83.85**	**6.95**
Yi-34b *Attention Sink Size: 4*					
Stream	0	508	1	85.35	
Random	508	0	1	82.17	-3.18
Interval	508	0	1	70.70	-14.65
SirLLM	508	0	0.7	**90.35**	**5.00**
Vicuna-7b *Attention Sink Size: 4*					
Stream	0	508	1	57.55	
Random	508	0	1	57.48	-0.13
Interval	508	0	1	54.45	-3.10
SirLLM	508	0	0.5	**59.15**	**1.60**
Vicuna-13b *Attention Sink Size: 4*					
Stream	0	508	1	71.10	
Random	508	0	1	69.27	-1.83
Interval	508	0	1	62.05	-9.05
SirLLM	508	0	0.6	**71.40**	**0.30**

Table 1: Results for the DailyDialog dataset are presented as follows: # Entropy and # Recent indicate the cache sizes allocated for tokens with the highest entropy and for recent tokens, respectively. ACC (%) represents the accuracy. Δ signifies the improvement of the model relative to the baseline StreamLLM.

smoother conversational experience.

3.2.2 Grocery Shopping

Dataset Construction To more vividly demonstrate SirLLM's superior memory capabilities, we designed the second task, Grocery Shopping, based on the CommonsenseQA (CSQA) (Talmor et al., 2019) dataset to create the Grocery Shopping dataset. Specifically, in the first interaction, the user informs the LLM of the groceries they wish to purchase. This is followed by 20 rounds of commonsense QA sessions with the LLM, where the questions are sourced from the train and development splits of the CSQA dataset and formatted as multiple-choice questions. After these 20 rounds, the user then asks the LLM to recall and select the required groceries from four options. This task is designed to test the LLM's long-term memory through the grocery-related questions and its ability to maintain excellent short-term memory and reasoning skills through the commonsense QA. The detailed dataset statistics can be found in Table 6 in Appendix B and dataset samples can be found in Figure 8 in Appendix D. From the table, we can see that the average token length per dialog is 1223.81 and all the 548 dialogs' total token number exceeds 1024. Therefore, we set the cache size for Grocery Shopping as 1024.

Result In the Grocery Shopping task, to enable the model to maintain a longer memory, we uniformly set the decay ratio to 1. The overall results can be found in Table 2.

	# Entropy	# Recent	ACC_c	ACC_g	Δ_c	Δ_g
Yi-6b Attention Sink Size: 4; $\eta_{decay} = 1$						
Stream	0	1020	71.33	25.73		
Random	1020	0	70.33	77.55	-1.00	51.82
Interval	1020	0	63.98	21.72	-7.20	-4.01
SirLLM	1020	0	**72.44**	**99.27**	**1.11**	**73.54**
Vicuna-7b Attention Sink Size: 4; $\eta_{decay} = 1$						
Stream	0	1020	50.84	28.65		
Random	1020	0	50.97	85.04	0.13	56.39
Interval	1020	0	47.21	23.72	-3.63	-4.93
SirLLM	1020	0	**51.04**	**96.17**	**0.20**	**67.52**
Vicuna-13b Attention Sink Size: 4; $\eta_{decay} = 1$						
Stream	0	1020	60.10	24.45		
SirLLM	1020	0	**60.23**	**97.08**	**0.13**	**72.63**
Yi-34b Attention Sink Size: 4; $\eta_{decay} = 1$						
Stream	0	1020	81.35	26.29		
SirLLM	1020	0	**81.44**	**89.60**	**0.09**	**63.31**

Table 2: Results for the Grocery Shopping dataset: # Entropy and # Recent indicate the cache sizes allocated for tokens with the highest entropy and for recent tokens, respectively. ACC_c and ACC_g represents the accuracy for commonsense QA and Grocery Shopping, respectively. Δ_c and Δ_g signify the improvement of the model relative to the baseline StreamLLM.

Table 2 clearly indicates that SirLLM consistently demonstrates an improvement in accuracy across different models. Specifically, SirLLM not only maintains its commonsense question-answering abilities that require short-term memory but also shows a substantial enhancement in memory capabilities for the Grocery Shopping task. This outcome is attributed to SirLLM's effective utilization of larger cache space allocated for key information, allowing it to maintain more contextual information in extended dialogues. This underscores SirLLM's efficacy not only in specific tasks but also in maintaining its memory advantage across different types of tasks, which is crucial for building a more adaptable and multifunctional dialogue system.

3.2.3 Rock-Paper-Scissors

Dataset Construction To better observe the performance of SirLLM in scenarios with infinitely long streaming dialogue inputs, we constructed a Rock-Paper-Scissors dataset. In this dataset, we created three players with preferences for throw-

	# Entropy	# Recent	η_{decay}	Paper			Rock			Scissors			Average
				win	tie	lose	win	tie	lose	win	tie	lose	win
Yi-6b													
Stream	0	1020	1	47.30	29.25	23.45	28.15	27.15	44.70	24.20	49.20	26.60	33.22
Random	1020	0	1	20.00	49.45	30.55	49.73	31.02	19.25	27.18	21.93	50.88	32.31
Interval	1020	0	1	19.45	50.15	30.40	50.00	30.90	19.10	27.35	20.8	51.85	32.27
SirLLM	1020	0	0.8	49.60	30.25	20.15	30.20	20.90	48.90	26.45	48.10	25.45	**35.42**
Yi-34b													
Stream	0	1020	1	44.10	34.00	21.90	30.35	26.55	43.10	32.15	40.20	27.65	35.53
Random	1020	0	1	30.57	19.68	49.62	19.08	50.12	30.97	51.70	27.35	20.95	33.78
Interval	1020	0	1	45.40	26.05	28.55	35.45	24.30	40.25	26.20	46.15	27.65	35.68
SirLLM	1020	0	0.8	48.30	30.65	21.05	30.30	29.40	40.30	29.50	39.00	31.50	**36.03**
Vicuna-7b													
Stream	0	1020	1	26.60	34.45	38.95	32.85	41.05	26.10	30.45	22.30	47.25	29.97
Random	1020	0	1	19.57	49.62	30.82	49.82	31.03	19.15	27.68	20.90	51.42	32.36
Interval	1020	0	1	29.15	37.95	32.90	24.40	45.80	29.80	28.25	35.05	36.70	27.27
SirLLM	1020	0	0.8	29.45	22.00	48.55	20.75	49.10	30.15	51.95	27.20	20.85	**34.05**
Vicuna-13b													
Stream	0	1020	1	29.75	22.20	48.05	29.90	44.05	26.05	49.85	26.35	23.80	36.50
Random	1020	0	1	44.02	26.63	29.35	30.43	21.70	47.97	28.32	46.43	25.25	34.26
Interval	1020	0	1	29.80	22.65	47.55	21.80	45.65	32.50	50.25	27.80	21.90	33.95
SirLLM	1020	0	0.9	30.35	19.60	50.05	45.10	34.65	20.25	51.00	26.55	22.45	**42.15**

Table 3: Results for the Rock-Paper-Scissors dataset. # Entropy and # Recent denote the allocated cache sizes for tokens with the highest entropy and for the most recent tokens, respectively. 'Rock,' 'Paper,' and 'Scissors' correspond to players with a preference for each respective move. 'Win,' 'Tie,' and 'Lose' represent the win rate (%), tie rate (%), and loss rate (%), respectively.

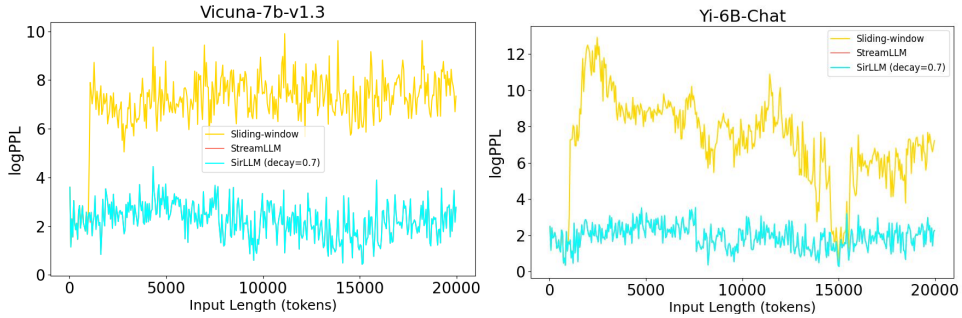
ing rock, paper, or scissors, respectively. In each round, we inform the LLM of the previous round's user move and the outcome, and then we ask the LLM to analyze the user's throwing preferences to maximize its own winning rate for the next round. Detailed information about the dataset and the probabilities of each player's moves can be found in Table 7 in Appendix B. A sample of the data is illustrated in Figure 9 in Appendix D. Unlike the DailyDialog and Grocery Shopping datasets, where the KV cache is reset to zero after each round, the Rock-Paper-Scissors task allows the LLM to engage in 2000 rounds of play without resetting the KV cache, achieving a truly infinite number of dialogue turns. This aims to observe whether SirLLM can remember key information and more user historical preferences to better maximize its win rate.

Result The results showcased in Table 3 for the Rock-Paper-Scissors dataset reveal that SirLLM consistently surpasses the baseline StreamLLM for players with varied throwing preferences. Upon closer examination of the data, it becomes apparent that SirLLM delivers a steady enhancement in win rates against players of different preferences, maintaining this enhanced performance uniformly across all the models evaluated. Furthermore, the decay mechanism integrated within SirLLM plays a crucial role in sustaining a balanced performance over numerous rounds, as reflected by its uniformly elevated win rates. This characteristic of SirLLM proves especially advantageous in scenarios involving extended interactions, such as long-duration Rock-Paper-Scissors games, where the model's capacity to adapt and recall previous moves is imperative for success.

4 Further Exploration

4.1 Few-shot

Brown et al. (2020) demonstrates that few-shot learning can significantly aid models in reasoning and answering questions. SirLLM, by eliminating redundant KV cache, achieves enhanced memory capabilities, which translates into improved performance on the CSQA dataset. This improvement could also be interpreted as SirLLM's ability to incorporate more few-shot exemplars with less cache, thereby attaining higher accuracy. On this premise, we compared SirLLM with 1-shot, 2-shot, and 3-shot learning approaches, with results as presented in Table 4. In n-shot experiments, we prepend the preceding n questions as few-shot exemplars before each question, aiming to simulate an input format similar to that of StreamLLM. As shown in the table, SirLLM not only improves upon the baseline



Average logPPL

	Vicuna-7b	Yi-6b
Sliding-window	7.07	7.13
StreamLLM	2.25	1.91
SirLLM	2.25	1.92

Figure 5: The perplexity of language modeling on 20K token text. The Sliding-window's PPL escalates dramatically once the token length exceeds the pre-trained length. In contrast, both SirLLM and StreamLLM, which incorporate attention sink tokens, show stable performance. SirLLM and StreamLLM's performances are almost identical, effectively demonstrating that SirLLM's memory mechanism does not impair the model's answering performance and can indeed reinforce the model's memory capabilities.

	ACC_c	ACC_g	Δ_c	Δ_g
Yi-6b				
Stream	71.33	25.73		
1-shot	58.66	25.00	-12.67	-0.73
2-shot	63.95	25.36	-7.38	-0.37
3-shot	65.42	23.72	-5.91	-2.01
SirLLM	72.44	99.27	1.11	**73.54**
Yi-34b				
Stream	81.35	26.29		
1-shot	75.14	23.91	-6.21	-2.38
2-shot	78.50	24.64	-2.85	-1.65
3-shot	79.20	25.18	-2.15	-1.11
SirLLM	81.44	89.60	0.09	**63.31**
Vicuna-7b				
Stream	50.84	28.65		
1-shot	48.54	27.01	-2.30	-1.64
2-shot	49.11	27.19	-1.73	-1.46
3-shot	49.81	27.55	-1.03	-1.10
SirLLM	51.04	96.17	0.20	**67.52**
Vicuna-13b				
Stream	60.10	24.45		
1-shot	55.34	22.26	-4.76	-2.19
2-shot	58.94	26.46	-1.16	2.01
3-shot	60.44	27.01	0.34	2.56
SirLLM	60.23	97.08	0.13	**72.63**

Table 4: Few-shot results for Grocery Shopping dataset

StreamLLM in both the CSQA and Grocery Shopping datasets, but it also maintains this enhanced performance despite the increment in the number of shots. This consistency underscores the model's ability to leverage the rich information contained within the few-shot examples without becoming overwhelmed by the increased data.

4.2 PPL for long text

Following the approach of StreamLLM, we plotted the log Perplexity (logPPL) of SirLLM, StreamLLM, and Sliding-window on texts spanning 20,000 tokens across various LLMs, as depicted in the Figure 5. The Figure reveals that while the Sliding-window model exhibits volatility in PPL, particularly beyond the length it was trained on, SirLLM maintains a consistent and stable PPL, suggesting a robustness to input length. The average logPPL values in the accompanying table further corroborate this, with SirLLM matching StreamLLM's performance closely across both Vicuna-7b and Yi-6b models. This indicates that SirLLM and StreamLLM have comparable short-term memory capabilities, with SirLLM not adversely affecting the model's ability to retain information over shorter durations. This alignment of PPL between SirLLM and StreamLLM, despite SirLLM's enhanced memory function, underscores the efficacy of SirLLM's design in managing longer context without compromising the language model's fluency or coherence.

5 Conclusion

Addressing the critical challenges of managing infinite input lengths and maintaining memory capability, SirLLM harmonizes long dialogue retention without the necessity of model fine-tuning by selectively fortifies the model's focus on pivotal information. Through experiments across three tailor-made tasks: DailyDialog, Grocery Shopping, and Rock-Paper-Scissors, SirLLM has demonstrated a consistent and stable improvement over existing models, irrespective of the complexity and length of the dialogues. The experimental outcomes validate the robustness and versatility of SirLLM, making it an invaluable asset for future explorations and applications in natural language processing.

Limitation

The limitations of SirLLM include: (1) Adaptation to Various Scenarios: Currently, users may need to manually adjust the decay ratio to achieve desired outcomes in different application scenarios. Developing an adaptive mechanism that automatically tunes the decay ratio based on specific contexts presents a viable direction for future work. (2) Significance Discrepancy: What users consider important information may not always align with the model's criteria, leading to potential omissions in memory retention. Therefore, a more accurate mechanism for cache retrieval and storage warrants detailed exploration in future research endeavors. This could ensure that the model better aligns with user priorities and improves overall recall accuracy.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Joshua Ainslie, Santiago Ontañón, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 268–284. Association for Computational Linguistics.

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

Moshe Berchansky, Peter Izsak, Avi Caciularu, Ido Dagan, and Moshe Wasserblat. 2023. Optimizing retrieval-augmented reader models via token elimination. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 1506–1524. Association for Computational Linguistics.

Avinash Bhat, Disha Shrivastava, and Jin LC Guo. 2023. Approach intelligent writing assistants usability with seven stages of action. *arXiv preprint arXiv:2304.02822*.

Desirée Bill and Theodor Eriksson. 2023. Fine-tuning a llm using reinforcement learning from human feedback for a therapy chatbot application.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023. Longlora: Efficient fine-tuning of long-context large language models. *CoRR*, abs/2309.12307.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2023. Model tells you what to discard: Adaptive KV cache compression for llms. *CoRR*, abs/2310.01801.

Mandy Guo, Joshua Ainslie, David C. Uthus, Santiago Ontañón, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2022. Longt5: Efficient text-to-text transformer for long sequences. In *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 724–736. Association for Computational Linguistics.

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. *CoRR*, abs/2308.16137.

Yunpeng Huang, Jingwei Xu, Zixu Jiang, Junyu Lai, Zenan Li, Yuan Yao, Taolue Chen, Lijuan Yang, Zhou Xin, and Xiaoxing Ma. 2023. Advancing transformer architecture in long-context large language models: A comprehensive survey. *arXiv preprint arXiv:2311.12351*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Majeed Kazemitabaar, Runlong Ye, Xiaoning Wang, Austin Z Henley, Paul Denny, Michelle Craig, and

Tovi Grossman. 2024. Codeaid: Evaluating a classroom deployment of an llm-based programming assistant that balances student and educator needs. *arXiv preprint arXiv:2401.11314*.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019a. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5244–5254.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019b. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017a. DailyDialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017b. Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*, pages 986–995. Asian Federation of Natural Language Processing.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6342–6353, Singapore. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Keivalya Pandya and Mehfuza Holia. 2023. Automating customer service using langchain: Building custom open-source gpt chatbot for organizations. *arXiv preprint arXiv:2310.05421*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4149–4158. Association for Computational Linguistics.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *CoRR*, abs/2306.05685.

A Additional Related Work

Many works (Li et al., 2019a; Guo et al., 2022; Han et al., 2023; Ainslie et al., 2020; Chen et al., 2023) focused on expanding the input context length of LLMs by optimizing the attention mechanism. Beltagy et al. (2020) first proposes the sliding window attention, which let each token to only attend to a certain number of recent tokens. When the KV cache is full sliding window attention would discard the earliest token to preserve a stable decoding speed and performance. Child et al. (2019) proposed the fixed Sparse Transformer. Formally, this method initially preserves the key and value states of recent tokens as local context information. Subsequently, it employs a column attention mechanism with a specified stride. This mechanism summarizes information from previous locations and propagates it to all future tokens, functioning as a form of global attention. Li et al. (2019b) proposed a LogSparse self-attention where each element can only to attend to itself and its previous cells with an exponential step size. Xiao et al. (2023) introduced the attention sink phenomenon and proposed StreamLLM, a model specifically designed to achieve true infinite input length. StreamLLM, during its attention calculation, maintains the focus on both the initial tokens and the recent tokens. This approach ensures stable performance in the context of infinite streaming conversations.

However, the aforementioned approaches either save tokens with given stride, randomly select, or do not preserve the key-value (KV) cache of history tokens, leading to significant forgetting issues in the model. SirLLM addresses this by utilizing the LLM itself to calculate token entropy, selectively preserving the KV cache of tokens with the highest entropy. This method effectively conserves memory space, ensuring that only the most crucial information is retained.

Another related work is by Ge et al. (2023), which introduced FastGen, an adaptive KV cache compression method for Large Language Models. FastGen implements four distinct compression strategies: Special Tokens, Punctuation, Locality, and Frequency. It begins by analyzing the behavior of various attention heads to select the most effective compression strategy for each. FastGen optimizes KV cache management when generating new tokens by applying the chosen compression strategy to each token, instead of merely appending new KV vectors. This method enhances memory efficiency without sacrificing model performance. However, this work focuses more on KV cache compression rather than infinite-length dialogue system and enhancing the memory capabilities of LLMs.

Another category of work related to our research is context compression. (Li et al., 2023) compress the input context by selecting the lexical units (tokens, phrases, sentences) with higher self-information computed by a base language model. (Berchansky et al., 2023) proposed a token filtering method for optimizing retrieved long documents to speed up the decoding process. This method involves using mean cross-attention scores computed at a specific layer across all attention heads to eliminate less critical tokens. Then, only the top k% of input tokens with the highest scores are retained and used in predicting subsequent tokens. Although retrieval-based methods can identify more accurate contexts based on input, they typically require greater computational and time resources. In contrast, SirLLM does not necessitate maintaining an additional vector database and does not disrupt the model's end-to-end computational process. SirLLM can significantly enhance the model's memory capabilities efficiently without modifying the model's architecture or requiring fine-tuning.

B Dataset Statistics

B.1 DailyDialog

DailyDialog	Statistics
#dialogs	518
#average turn	3.85
#average token (dialog)	461.55
#average word (dialog)	309.92
# dialogs ($\geq$ 512)	199

Table 5: Detailed statistics of DailyDialog(modified)

We modified the test split of the DailyDialog dataset to create a set of four-option multiple-choice questions. This set includes one correct option and three dummy choices, which are selected from the validation split. Table 5 presents the detailed statistics of the modified DailyDialog dataset. In this table, #dialogs indicates the total number of dialogs; #average turn refers to the average number of turns per dialog; #average token (dialog) represents the average number of tokens

per dialog, calculated using the Vicuna-7b-v1.3 tokenizer; #average word (dialog) signifies the average number of words per dialog; and #dialogs ($\geq$ 512) shows the count of dialogs where the total number of tokens exceeds 512.

B.2 Grocery Shopping

Grocery Shopping	Statistics
#dialogs	548
#groceries	53
#average turn	22
#average token (dialog)	1223.81
#average word (dialog)	631.60
# dialogs ($\geq$ 1024)	548

Table 6: Detailed statistics of Grocery Shopping

Table 6 presents the detailed statistics of the Grocery Shopping dataset. In this table, #dialogs indicates the total number of dialogs; #groceries represents the number of different types of groceries; #average turn refers to the average number of turns per dialog; #average token (dialog) represents the average number of tokens per dialog, calculated using the Vicuna-7b-v1.3 tokenizer; #average word (dialog) signifies the average number of words per dialog; and #dialogs ($\geq$ 1024) shows the count of dialogs where the total number of tokens exceeds 1024.

B.3 Rock-Paper-Scissors dataset

Rock-Paper-Scissors		Statistics
#rounds		2000
#average token (rounds)		54
#average word (rounds)		35
Player 1 (Rock)	rock	0.5
	paper	0.3
	scissors	0.2
Player 2 (Paper)	rock	0.2
	paper	0.5
	scissors	0.3
Player 3 (Scissors)	rock	0.3
	paper	0.2
	scissors	0.5

Table 7: Detailed statistics of Grocery Shopping

Table 7 presents the detailed statistics of the Rock-Paper-Scissors dataset. In this table, #rounds indicates the total number of Rock-Paper-Scissors rounds; #average token (rounds) represents the average number of tokens per rounds, calculated using the Vicuna-7b-v1.3 tokenizer; #average word (rounds) signifies the average number of words per round. In the table 7, the preferences for each player's moves and their corresponding probabilities of throwing rock, paper, or scissors are also listed.

C The Impact of Decay Ratio on Memory Retention

To more vividly illustrate the impact of the decay ratio on the memory capabilities of LLMs, we conducted experiments using various decay ratios in the Grocery Shopping task. The results of these experiments are presented in Figure 6. From the Figure 6, we can observe that when the decay ratio is set below one, the model completely forgets the groceries desired by the user after 20 rounds of commonsense question and answer sessions. However, adjusting the decay ratio does not significantly impact the model's performance on tasks requiring short-term memory, such as commonsense question answering. By fine-tuning the decay ratio, we can flexibly adapt the memory capabilities of the LLM to suit different scenarios. This effectively demonstrates the stability and efficacy of SirLLM's memory mechanism.

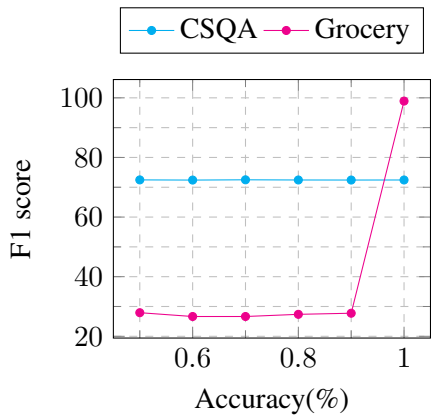


Figure 6: Performance of different decay ratio in Grocery Shopping dataset.

D Dataset Samples

DailyDialog

USER: Choose the best response for the input: What are your personal weaknesses ? Choices: (A) The Olympic Park has a small forest . It is fit for picnic . (B) Then how about this room , with just shower ? (C) I'm afraid I'm a poor talker . I,m not comfortable talking with the people whom I have just met for the first time . That is not very good for business , so I have been studying public speaking . (D) We have Sichuan food , Hangzhou food and Beijing Food . Which do you like best ?

ASSISTANT: C

USER: Choose the best response for the input: Are you more of a leader or a follower ? Choices: (A) I'm living at Room 222 , Building 18 , N . P . U . (B) I'm not sure , but she probably won't remember if she has ! She's very forgetful ! (C) I don't try to lead people . I'd rather cooperate with everybody , and get the job done by working together . (D) It's okay . It's probably my husband's fault.Is there a pool in this hotel ?
ASSISTANT: C

USER: Choose the best response for the input: Do you think you can make yourself easily understood in English ? Choices: (A) Yes , in most circumstances . (B) Sure , I'll try my best . Here we are . (C) Fine , I'll bring both our office furniture and equipment catalogs on Thursday . (D) No , I want the fish .
ASSISTANT: A

Figure 7: A sample from the DailyDialog dataset

Grocery Shopping

USER: I want you to buy the GROCERY: [chocolate]

ASSISTANT: OK

USER: A revolving door is convenient for two direction travel, but it also serves as a security measure at a what? Choices: (A) bank (B) library (C) department store (D) mall (E) new york

ASSISTANT: A

USER: What do people aim to do at work? Choices: (A) complete job (B) learn from each other (C) kill animals (D) wear hats (E) talk to each other

ASSISTANT: A

...

USER: Which one is the GROCERY that I want you to buy earlier? Choices: (A) vanilla extract (B) tea (C) chocolate (D) eggs

ASSISTANT: C

Grocery List

"milk", "bread", "eggs", "apples", "bananas", "chicken", "rice", "tomatoes", "lettuce", "cheese", "orange juice", "yogurt", "carrots", "bell peppers", "onions", "garlic", "beef", "pasta", "cereal", "olive oil", "butter", "spinach", "cucumber", "potatoes", "chocolate", "coffee", "tea", "flour", "sugar", "baking soda", "oats", "almonds", "peanut butter", "jelly", "canned beans", "canned tomatoes", "frozen peas", "frozen corn", "tofu", "salmon", "shrimp", "maple syrup", "honey", "mustard", "ketchup", "soy sauce", "vinegar", "baking powder", "vanilla extract", "cinnamon", "paprika", "black pepper", "salt"

Figure 8: A sample from the Grocery Shopping dataset

Rock-Paper-Scissors

USER: Let's play a game of Rock-Paper-Scissors. Choose one option from: (A) rock; (B) paper; (C) scissors. You choose:

ASSISTANT: C

USER: The opponent chose [paper] in the last round. You [win]. Analyze the opponent's history preferences to maximize your win rate. Choose one option from: (A) rock; (B) paper; (C) scissors. You choose:

ASSISTANT: A

USER: The opponent chose [rock] in the last round. You [tie]. Analyze the opponent's history preferences to maximize your win rate. Choose one option from: (A) rock; (B) paper; (C) scissors. You choose:

ASSISTANT: B

...

USER: The opponent chose [rock] in the last round. You [lose]. Analyze the opponent's history preferences to maximize your win rate. Choose one option from: (A) rock; (B) paper; (C) scissors. You choose:

ASSISTANT: B

Figure 9: A sample from the Rock-Paper-Scissors dataset