Stateful Memory-Augmented Transformers for Efficient Dialogue Modeling

Qingyang Wu
Columbia University
qw2345@columbia.edu

Zhou Yu
Columbia University
zy2461@columbia.edu

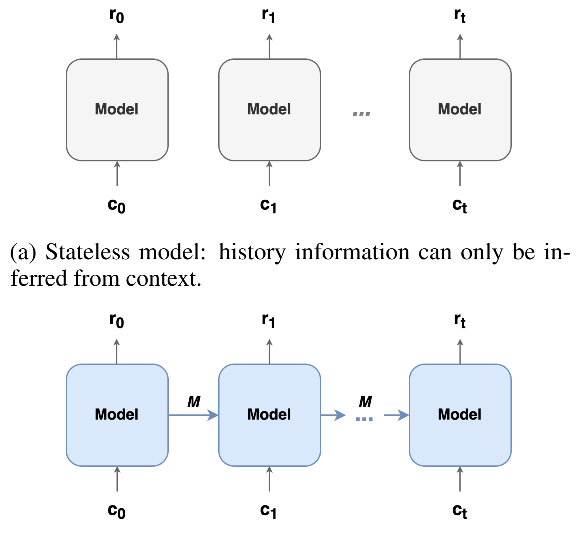
Abstract

Transformer models have achieved great performance in dialogue generation tasks. However, their inability to process long dialogue history often leads to truncation of the context. To address this problem, we propose a novel memory-augmented transformer that is compatible with existing pre-trained encoder-decoder models and enables efficient preservation of the dialogue history information. The new model incorporates a separate memory module alongside the pre-trained transformer, which can effectively interchange information between the memory states and the current input context. We evaluate the efficiency of our model on three dialogue datasets and two language modeling datasets. Experimental results show that our method has achieved superior efficiency and performance compared to other pre-trained Transformer baselines.

1 Introduction

Recently, Transformers (Vaswani et al., 2017) have achieved state-of-the-art results in many natural language processing tasks, particularly in language understanding and generation. In the field of open-domain dialogue modeling, DialoGPT (Zhang et al., 2020) has achieved great performance by extending the Transformer decoder model GPT2 (Radford et al., 2019) by pre-training it on a large corpus of open-domain dialogues. Subsequently, Meena (Adiwardana et al., 2020) and BlenderBot (Roller et al., 2021) further improved the performance of response generation with larger Transformer encoder-decoder models.

However, the attention mechanism in Transformer-based dialogue models, which has complexity scaling quadratically with the sequence length, makes them computationally expensive for long context inputs. As an example, BlenderBot (Roller et al., 2021) has to truncate the input length to 128 tokens for better efficiency, otherwise, the model's computational cost would become infeasible for real-time conversation tasks such as chatbot applications.



(a) Stateless model: history information can only be inferred from context.

(b) Stateful model: history information is carried by memory states M.

Figure 1: Illustration of Stateful vs. Stateless. "State" means a model's internal state representations. c_t and r_t represent the dialog context and response at timestep t. Stateful models can have smaller context size compared to stateless models because of memory.

Many studies have addressed the challenge of processing long sequences with Transformers (Katharopoulos et al., 2020; Qin et al., 2022; Hua et al., 2022; Dai et al., 2019; Rae et al., 2020). However, they focused on pure language modeling tasks and are primarily decoder-only models. Another limitation is that their models are not pre-trained with large corpora, which increases difficulty for performance comparison with existing pre-trained Transformers. More recently, Beltagy et al. (2020) addressed the problem by proposing Longformer Encoder-Decoder (LED) based on the pre-trained encoder-decoder model BART (Lewis et al., 2020) for sequence-to-sequence tasks. It uses a sparse attention window and achieves a linear time complexity. Nevertheless, LED is inefficient in dialogue modeling, because it is stateless and depends on the context to provide history information.

In this work, we utilize the idea of memory-augmented Transformers (Wu et al., 2020; Bulatov et al., 2022; Burtsev and Sapunov, 2020) and convert an existing pre-trained Transformer into a stateful model with internal memory representations. A stateful model can keep history information in its internal hidden states in contrast to a stateless model. As shown in Figure 1, most existing Transformer encoder-decoder models are stateless. They rely on the input context to provide history information, and therefore they typically require a larger context to avoid information loss. For a stateful model, it can store history information in its memory states. With a smaller context size, the stateful model can still retain most of the history information, which results in better efficiency than a stateless model.

Memformer (Wu et al., 2020) achieves statefulness by having internal memory states to store history information. The memory size is fixed so that the model will prioritize memorizing important information. To interact with the memory, it consists of a memory reader and a memory writer into a Transformer encoder-decoder model. Memformer has shown better efficiency on the language modeling dataset WikiText-103 (Merity et al., 2017) than the decoder-only models Transformer-XL (Dai et al., 2019) and Compressive Transformer (Rae et al., 2020). However, Memformer only focused on language modeling tasks and was not pre-trained on large corpora and cannot be directly used for downstream applications. Also, its structure does not fit the existing pre-trained Transformer encoder-decoder models.

To address these limitations in Memformer, we propose MemBART with new architecture modifications and training techniques that converts the existing pre-trained Transformer encoder-decoder model BART (Lewis et al., 2020) into a stateful memory-augmented Transformer encoder-decoder model. Specifically, we introduce a dual attention stream to enhance the memory module, which is accomplished by using a separate Transformer to update the memory states at each layer. We also implement a residual gated memory update mechanism to better retain important history information. At each timestep, the gating mechanism controls the extent of keeping or overwriting each memory slot's values for the next timestep. We further pre-train the memory module and enable the model to memorize important history information. As MemBART is a pre-trained model, it can be used for broader downstream applications.

Our contributions focus on introducing a novel stateful memory-augmented Transformer encoder-decoder model that is compatible with the existing pre-trained language model BART. We evaluate our model's performance on three dialogue datasets and two language modeling datasets. Experimental results demonstrate our model's superior efficiency in terms of latency and performance. We will release the checkpoints of our pre-trained MemBART models.

2 Related Work

2.1 Stateful Language Models

Recurrent neural networks (RNN) are naturally stateful models. Training RNNs on long time-series data often requires truncated back-propagation through time (Williams and Peng, 1990) and passing the internal states of the model to the next batch. Stateful RNNs are also widely used for recurrent reinforcement learning (Gold, 2003; Hausknecht and Stone, 2015), where the states of the agent need to be maintained. There have been variants of stateful RNNs (Weston et al., 2015; Sukhbaatar et al., 2015; Graves et al., 2016) studied to solve various tasks. However, due to parallel inefficiency, they are gradually succeeded by large Transformer models (Vaswani et al., 2017).

Decoder-only Transformers can be stateful by storing the previously computed keys and values. Transformer-XL (Dai et al., 2019) and Compressive Transformer (Rae et al., 2020) explore this direction, but their states have a theoretical maximum range of maintaining the information from previous tokens. Thus, they normally require a large memory size to be effective.

Linear attention Transformers can act as RNNs with states. They use a linearized kernel to approximate softmax operation. Different variants of linear Transformers (Katharopoulos et al., 2020; Hua et al., 2022; Qin et al., 2022) have been proposed and achieved great performance in language modeling tasks. However, there are no pre-trained large linear Transformers yet. Similar models such as Memorizing Transformer (Wu et al., 2022), Block-Recurrent Transformer (Hutchins et al., 2022), Recurrent Memory Transformer (Bulatov et al., 2022) focus on language modeling tasks or synthetic tasks and are not applicable for broader NLP tasks.

2.2 Stateless Language Models

For long documents processing, sparse Transformers are another direction. The main idea is to apply a sparse attention matrix to skip computations of tokens that are far away. Many works (Child et al., 2019; Zaheer et al., 2020; Beltagy et al., 2020) have explored different sparse attention patterns with linear complexity. Especially, Longformer extended the pre-trained BART (Lewis et al., 2020) with sparse attention and introduced Longformer-Encoder-Decoder (LED) for sequence-to-sequence tasks. However, these models are stateless, which are inefficient for dialogue modeling. They require the context to be long enough to cover enough history information. The context also needs to be re-computed at every timestep due to bidirectional attention. Besides, sparse Transformers need full attention for the local window, which makes them less competitive against non-sparse models when the context is short. In contrast, our stateful memory-augmented method can have a shorter context input while still memorizing the history information.

3 Methods

In this section, we first describe the background of memory-augmented Transformers. Then we introduce an novel memory module that is compatible with existing Transformer encoder-decoder models. We further pre-train the memory module with the sequence denoising objective to initialize the memorization capability. In the end, we analyze the theoretical complexity of our proposed model for dialogues.

3.1 Memory-Augmented Transformer

Memformer (Wu et al., 2020) modifies a Transformer encoder to interact with a fixed-size dynamic memory, so that it can store and retrieve history information. It comprises a memory reader and a memory writer. The memory reader utilizes cross attention to retrieve history information from the memory M_t:

$$\begin{aligned} Q_{H^l}, K_{M^l}, V_{M^l} &= H^l W_Q, M_t W_K, M_t W_V \\ A^l &= \text{MHAttn}(Q_{H^l}, K_M) \\ H^{l+1} &= \text{Softmax}(A^l)\, V_M \end{aligned}$$

where H^l is the input's hidden states at layer l.

For the memory writer, each memory slot $m_t^i \in M_t$ is projected into a query to attend to itself and the final layer's input hidden states H^L:

$$\begin{aligned} Q_{m_t^i}, K_{m_t^i} &= m_t^i W_Q, m_t^i W_K \\ K_{H^L}, V_{H^L} &= H^L W_K, H^L W_V \\ A_{m_t^i} &= \text{MHAttn}(Q_{m_t^i}, [K_{m_t^i}; K_{H^L}]) \\ m_{t+1}^i &= \text{Softmax}(A^{m_t^i})[m_t^i; V_{H^L}] \end{aligned}$$

Memory states are reset with the reset signal r.

$$\begin{aligned} r &= \begin{cases} 1, & \text{if } t = 0 \\ 0 & \text{otherwise} \end{cases} \\ M_t' &= \text{LayerNorm}((1-r) \odot M_t + v_b) \end{aligned}$$

Also, we normalize the memory states at every timestep with a bias term v_b as the forgetting mechanism. v_b determines the initial memory M_0 which is LayerNorm(v_b).

3.2 Dual Attention Stream

Memformer adds cross-attention layers between self-attention and feed-forward layers to achieve memory functionality. However, directly injecting layers inside a pre-trained Transformer will interfere the distribution of learnt knowledge and lead to worse performance. Therefore, we aim to integrate the memory module with a minimal influence of the original pre-trained Transformers.

We propose a dual attention stream so that the memory path has minimal interference with the input sequence's data path. Inside every layer l, we separately project the input sequence H^l and the memory states M^l to queries Q, keys K, and values V:

$$\begin{aligned} Q_{H^l}, K_{H^l}, V_{H^l} &= W_{H^l} H^l \\ Q_{M^l}, K_{M^l}, V_{M^l} &= W_{M^l} M^l \end{aligned}$$

Then, there are two attention streams to realize memory reading and memory writing simultaneously at each layer:

$$\begin{aligned} A_{H^l} &= \text{Attention}(Q_{H^l}, [K_{M^l}; K_{H^l}]) \\ H^{l+1} &= \text{Softmax}(A_{H^l})[V_{M^l}; V_{H^l}] \\ A_{M^l} &= \text{Attention}(Q_{M^l}, [K_{M^l}; K_{H^l}]) \\ M^{l+1} &= \text{Softmax}(A_{M^l})[V_{M^l}; V_{H^l}] \end{aligned}$$

Specifically, the attention stream A_{H^l} serves as memory reading, where the input sequence's hidden states H^l gathers the information from the

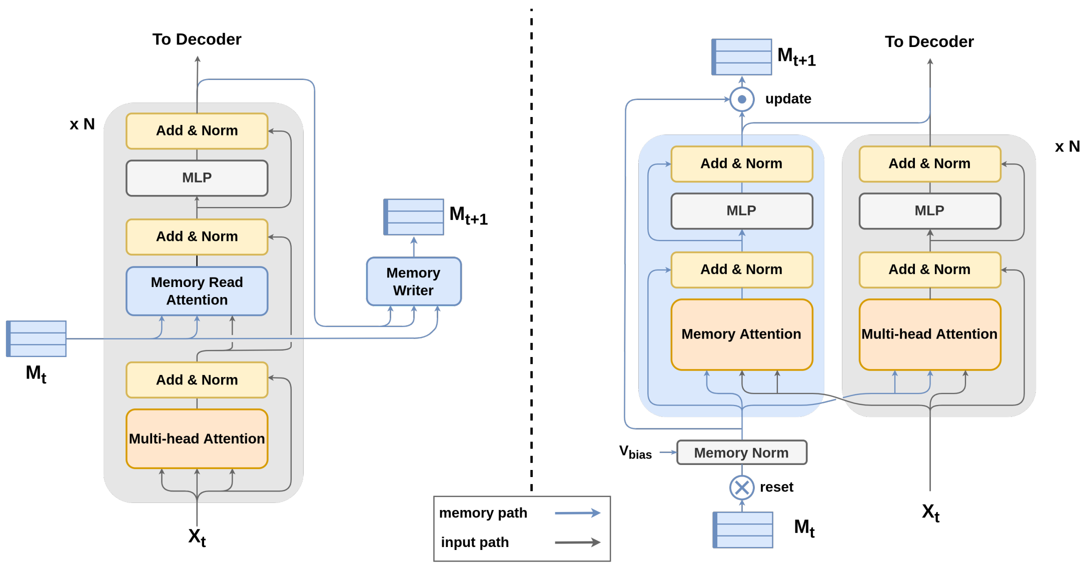


Figure 2: **Left**: Memformer with cross attention to read from memory and a separate memory writer to update information in memory slots. **Right**: MemBART with the dual attention stream to handle memory reading and writing simultaneously. This design reduces the interference with the pre-trained model's distribution.

memory states M_t to get the next layer's representation H^{l+1}. The other attention stream A_{M^l} serves as memory writing. Note that we update memory states at every layer. Each memory slot $m^l \in M^l$ attend to itself and the input's hidden states to obtain the next layer's memory slots M^{l+1}. Each memory slot does not interfere with other memory slots when updating.

This dual attention stream allows the information to exchange effectively between the memory slots and the input sequence, while minimally affects the original pre-trained Transformer's knowledge.

3.3 Residual Gated Memory Update

The dual attention stream achieves memory reading and writing simultaneously at each layer. However, as the number of layers increases, the final layer's memory representation may be hard to retain the previous timestep's information.

As a workaround, we implement a residual gating mechanism. We let the encoder predict a score $z_t \in (0, 1)$ with sigmoid to control the update of each memory slot separately.

$$\begin{aligned} H_{M_{t+1}} &= \text{Encoder}(x_t, M_t) \\ M'_{t+1} &= \text{MLP}(H_{M_{t+1}}) \\ z_t &= \sigma_z(W_z H_{M_{t+1}} + b_z) \\ M_{t+1} &= z_t \odot M'_{t+1} + (1 - z_t) \odot M_t \end{aligned}$$

x_t is the input sequence length. $H_{M_{t+1}}$ is the final layer's memory hidden states. M'_{t+1} is the next timestep's memory slots candidate.

3.4 Learning to Memorize Important Information

As the memory size is fixed, the model needs to learn what information to keep and what to forget, but the memory module initially has no knowledge of that. Therefore, it requires further pre-training for the memory module to learn to memorize important information.

We use the sequence denoising objective as the memory module's pre-training objective. We split a document into segments, add random masks to these segments, and feed them into the model sequentially. This objective can teach the model to memorize important information. If important words such as named entities appear in previous timesteps but are masked in the current input context, the model can predict them back with the help of memory. For less important words that can be inferred from the context or grammar, the model can choose not to store them in the dynamic memory.

3.5 Complexity Analysis

Our method is efficient in processing long sequences compared to traditional Transformers, especially in modeling dialogues. For example, consider a dialogue with T turns, and N tokens at each turn. The overall complexity for a Transformer to process all the turns would be $\mathcal{O}(N^2 + 2N^2 + \ldots + TN^2)$, or simply $\mathcal{O}(T^2N^2)$. If we keep all the history tokens, a traditional encoder-decoder model would require to re-compute all the history tokens because of the bidirectional attention, which increases the complexity. In practice, due to the limitation of the maximum number of positional embeddings and the GPU memory constraint, we often truncate the dialog history to a fixed length.

In contrast, our stateful model can store the history information in the fixed-size memory. The implementation has a complexity of $\mathcal{O}(TN^2)$, and it does not require re-computation for the history tokens. For efficient Transformer models such as Longformer, the complexity can be reduced from $\mathcal{O}(T^2N^2)$ to $\mathcal{O}(T^2N)$. However, when the context length N is small, the number of turns T is the leading factor for efficiency, where our method shows better efficiency in theory.

4 Memory Module Pre-training

As mentioned above, the memory module needs to be pre-trained to learn to memorize important information. However, to compare the effectiveness of our proposed approach with the previous models, it would be expensive to pre-train all model variants. Therefore, we use a simple text recall task to evaluate different models before pre-training on large corpora.

For all model variants, we choose BART (Lewis et al., 2020) as the backbone as it has demonstrated great performance on conversational datasets. We also initialize the memory module's self attention and feed-forward parameters with the pre-trained weights for better adaptation.

4.1 Model Selection with Text Recall Task

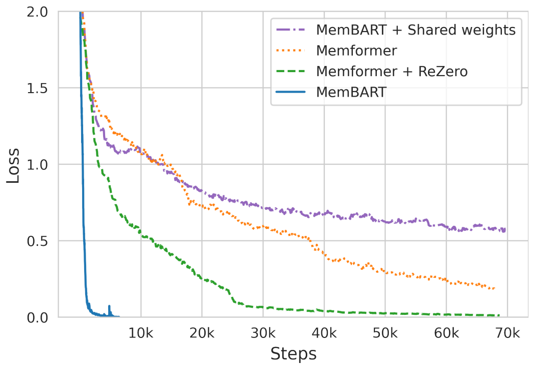


Figure 3: Loss curves for different models for the text recall task.

The text recall task lets the model recover the previous timestep's input text, where the history information can only flow through the memory bottleneck.

We evaluate different model variants with the text recall task to select the best model before pre-training. The first is directly adding the memory cross-attention layers into BART (Memformer), which the model's architecture is similar to Memformer (Wu et al., 2020). The second model uses ReZero (Bachlechner et al., 2021) that it applies a zero-initialized trainable weight when adding the memory cross-attention layer, so that the model's output distribution is not changed initially (Memformer + ReZero). The third model is our proposed MemBART where the memory module shares the weights with BART (MemBART + Shared weights). The last one is our final model MemBART without sharing weights between the memory module and the pre-trained Transformer (MemBART).

The training details are in Appendix A. In Figure 3, we can observe that the original Memformer (orange) did not converge to zero loss. MemBART with shared weights (purple) also did not converge and performed worse, suggesting that the memory states should have different distribution space from the word embeddings. Memformer with ReZero (green) converged slowly in the end. In comparison, MemBART (blue) only used one quarter of the time to reach nearly zero loss. The result shows that our proposed memory module architecture is compatible with the pre-trained BART and can be efficiently trained for memorization tasks.

4.2 Sequence Denoising Pre-training

We have shown that the proposed MemBART has outperformed Memformer and other model variants. Now, we pre-train MemBART with the sequence denoising objective for the memory module to memorize important information. We have two sizes of models: MemBART base (183M) and MemBART large (558M). We use a similar pre-training corpus to BART to avoid data leaking, which includes a subset of BooksCorpus (Zhu et al., 2015), CommonCrawl (Raffel et al., 2020), OpenWebText (Gokaslan and Cohen, 2019). We filter out documents that are less than 512 tokens for better memory learning. We split the document into segments with a window size of 512 and an overlap of 128 tokens. At each timestep, we randomly mask 30% of input sequence tokens. We pre-train the model for 100k steps, which takes about 0.125% of the original pre-training cost of BART. Other pre-training details are in Appendix B.

Models \ Context	**64**		**128**		**256**		**512***	
	PPL ↓	**F1 ↑**	**PPL↓**	**F1 ↑**	**PPL↓**	**F1 ↑**	**PPL↓**	**F1 ↑**
BART base	10.91	25.01	9.39	25.44	8.64	26.31	8.76	26.22
Memformer base (512)	9.14	25.37	8.95	25.81	8.64	27.23	-	-
MemBART base (64)	8.68	27.34	8.58	27.37	8.46	27.05	-	-
w/o history	10.52	25.54	9.44	26.52	8.57	26.23	-	-
w/o pre-training	10.67	25.26	9.37	26.12	8.60	26.45	-	-
MemBART base (128)	**8.59**	27.45	8.57	27.52	8.39	**27.52**	-	-
MemBART base (256)	8.60	**27.65**	**8.49**	**27.68**	**8.38**	27.41	-	-
GPT2-12	10.93	25.18	9.86	26.03	9.06	26.55	9.04	26.52
GPT2-24	9.51	25.46	8.56	26.52	7.82	27.19	7.81	27.20
BART large	9.12	25.50	8.01	26.84	7.33	28.67	7.31	28.64
MemBART large (128)	**7.47**	**28.06**	**7.33**	**28.57**	**7.15**	**29.16**	-	-

Table 1: PersonaChat results. MemBART with 64 context length outperforms the baselines with 512 context length. MemBART (64) means the memory size is 64. "w/o pre-training" means without pre-training the memory module. * denotes that the context window can cover most dialogues.

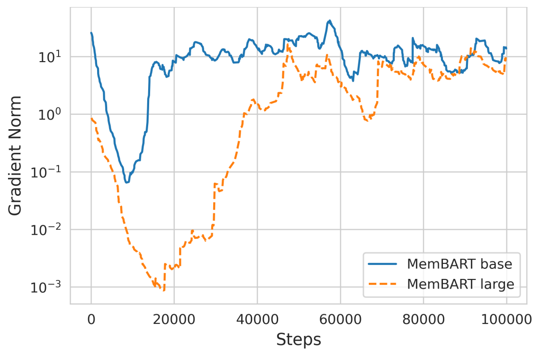


Figure 4: Memory's gradient norm during pre-training. When the gradient is near the minimum, the model performs terribly in downstream tasks.

Datasets	**#Turns**	**Avg. Len**	**Max Len**
PersonaChat	14.66	244	715
Persuasion	20.58	456	1,437
Multi-Session Chat	60.52	1,823	2,705
Arxiv	-	13,409	156,605
PG19	-	105,830	1,181,156

Table 2: Dialogue and long document datasets statistics.

In Figure 4, we show the magnitude of the gradients flowing through memory states during pre-training. At the early stage of the pre-training (less than 20,000 steps), we observe that the MemBART base model does not perform well in the downstream tasks. We suspect that when the gradient norm is small, it means that model is not actively using the memory states. Therefore, the gradient norm serves as an indicator of when the memory module is learnt. For MemBART large, the downstream tasks' performance improves after 50,000 steps when the gradient norm reaches the maximum. This pattern suggests that it needs a certain number of pre-training steps for the memory module to learn to memorize important information, and the large model needs more update steps to learn memorization.

5 Downstream Tasks

In this section, we introduce the downstream tasks and datasets for evaluation. Then, we show the results on the dialogue and language modeling tasks.

5.1 Experiment Setup

Datasets: We experimented on three different dialogue datasets: PersonaChat (Zhang et al., 2018), PersuasionForGood (Wang et al., 2019), and Multi-Session Chat (MSC) (Xu et al., 2022). Especially, Multi-Session Chat addresses the problem of lacking long-context dialogue datasets in the current community. It is the largest human-human dataset for long conversations with five sessions and average 60 turns of utterances. To further test the model's capability, we also evaluated our model on two language modeling tasks: Arxiv and PG19 (Rae et al., 2020). Due to computational constraints, we selected the $2,809$ CS AI Arxiv papers, and a subset of 200 books from PG19 for evaluation. We split 10% of the data for testing. The statistics of all the datasets are shown in Table 2.

Baselines: We compared MemBART with

Base Models	**Context**	**Latency (ms)** ↓	**Total** ↓	**Session 1** ↓	**Session 2** ↓	**Session 3** ↓	**Session 4** ↓	**Session 5** ↓
BART base	128	16.41	13.05	10.99	12.52	13.18	13.65	14.02
BART base	256	22.12	12.83	10.94	12.29	12.97	13.37	13.78
BART base	512	36.80	12.68	10.92	12.14	12.77	13.19	13.61
BART base	1,024	64.65	12.53	10.81	11.93	12.50	13.10	13.55
LED base	2,048	227.75	12.52	10.76	12.13	12.59	12.93	13.42
Memformer base (512)	128	24.37	12.77	10.99	12.50	13.09	13.46	13.81
MemBART base (128)	128	20.42	12.41	10.72	11.95	12.52	12.88	13.23
MemBART base (128)	256	32.09	12.25	**10.62**	11.76	12.37	12.71	13.06
MemBART base (128)	512	66.70	**12.15**	10.63	**11.67**	**12.23**	**12.57**	**12.97**
Large Models	**Context**	**Latency (ms)**	**Total**	**Session 1**	**Session 2**	**Session 3**	**Session 4**	**Session 5**
GPT2-12	512	65.77	13.99	12.81	13.45	14.03	14.33	14.78
GPT2-12	1,024	149.05	13.56	12.82	13.48	13.84	13.53	13.82
GPT2-24	512	172.43	11.65	11.07	11.14	11.66	11.86	12.20
GPT2-24	1,024	395.84	11.56	11.03	11.12	11.52	11.75	12.11
BART large	128	45.37	10.61	9.50	10.13	10.68	10.94	11.29
BART large	256	63.79	10.37	9.38	9.86	10.44	10.67	11.02
BART large	512	103.20	10.23	9.44	9.71	10.26	10.52	10.85
BART large	1,024	190.79	10.10	9.41	9.64	10.06	10.36	10.68
LED large	2,048	655.19	10.05	9.43	9.60	10.04	10.27	10.60
MemBART large (128)	128	59.51	10.17	9.22	9.61	10.24	10.47	10.85
MemBART large (128)	256	102.42	10.09	**9.20**	9.65	10.09	10.38	10.72
MemBART large (128)	512	197.79	**9.99**	9.22	**9.51**	**10.03**	**10.23**	**10.58**

Table 3: MSC test set perplexity results. Compared to LED 2048 context length, MemBART base is 11.15x faster (227.75 vs. 20.42) and MemBART large is 6.40x faster (655.19 vs. 102.42). More details are in Appendix C.

GPT2, BART, and Longformer (LED) under different context windows. We also evaluated Memformer+ReZero with memory length 512 (denoted as "Memformer base (512)") to show the effectiveness of the new architecture. Note that Memformer+ReZero is pre-trained under the same setting of MemBART-base. We used beam search with a beam size of 4 when generation is needed. For evaluation metrics, we reported perplexity for all the datasets and word overlap F1 for PersonaChat. We also measured the latency as an important metric for efficiency, where the results for all the models are in Table 3.

Models	**Context Length**			
	128	**256**	**512**	**1024***
BART base	10.93	10.90	10.80	10.78
MemBART base (64)	10.69	10.66	10.66	-
w/o history	10.86	10.79	10.75	-
MemBART base (128)	10.65	10.57	10.56	-
MemBART base (256)	**10.59**	**10.56**	**10.54**	-
GPT2-12	10.51	10.38	10.33	10.31
GPT2-24	9.37	9.20	9.14	9.11
BART large	9.54	9.40	9.24	9.27
MemBART large (128)	**9.34**	**9.18**	**9.12**	-

Table 4: Perplexity ↓ results for Persuasion dataset. MemBART (64) means the memory size is 64. * denotes that the context length can cover most dialogs.

5.2 Dialogue Datasets Results

Table 1,4,3 show the results for PersonaChat, PersuasionForGood, and MSC, respectively. We list several main observations below.

The memory module memorizes the history information, and the pre-training is necessary. In Table 1, we show that by resetting the memory states (w/o history), MemBART performs similarly to BART base. Also, without pre-training, it does not initially learn to memorize the history.

MemBART can be much faster with a small input context size while having better performance. In PersonaChat, MemBART with 64 memory size and 64 context length can be on par with the performance of BART with 512 context length. The same pattern holds for PersuasionForGood (Persuasion) and Multi-Session Chat(MSC) dataset. Especially in MSC, MemBART base can achieve similar perplexity (12.41) compared to LED base with context length 2,048, but ***11.15 times faster***. MemBART large achieves similar perplexity (10.09) compared to LED large with context length 2,048, while ***6.40 times faster***.

Encoder-decoder models utilize history information better than decoder-only models. For Per-

Models	Context	Arxiv	PG19
BART base	512	15.40	33.70
BART base	1,024	15.09	31.20
LED base	2,048	**13.97**	30.08
MemBART base (128)	512	14.34	**29.81**
GPT2-12	512	17.53	32.20
GPT2-12	1,024	15.35	28.31
GPT2-24	512	15.34	22.33
GPT2-24	1,024	13.84	**20.86**
BART large	512	12.92	24.08
BART large	1,024	12.31	23.07
LED large	2,048	**11.82**	23.04
MemBART large (128)	512	12.24	22.26

Table 5: Language Modeling perplexity scores on Arxiv and PG19 datasets. Lower is better.

sonaChat and MSC, BART base and MemBART large outperforms GPT2-12 and GPT2-24 respectively. The exception is in Persuasion, where the conversations contain more single-turn utterances. This observation suggests that encoder-decoder models utilize history information better, and it is probably because of the bidirectional context.

MemBART's performance improves as the context size increases. BART and GPT2's performance improves when context size increases. The results show that increasing the context size for MemBART can also improve its performance, although only by a small margin. We suspect that using a larger context size can help the model to enhance the memorization of history information and alleviate situations where some information is not kept in the memory.

Increasing memory size improves MemBART performance. For MemBART models, the history information is stored inside memory. Thus, we want to study how the performance scales with the memory size. We evaluated memory size 64, 128, and 256. We observe that when increasing the size of memory from 64 to 128, there is a large improvement, but from 128 to 256, the improvement is marginal.

5.3 Language Modeling Datasets Results

We have also evaluated on two language modeling tasks Arxiv and PG19 to better understand the model's effectiveness. Due to the computational constraint, we use subsets of the two datasets for evaluation. We show the results in Table 5.

MemBART performs slightly worse than LED large with 2048 context on Arxiv, but better on PG19. We suspect that it is because Arxiv papers are very structured and use terminologies across the paper, but PG19 books have less long-term dependency. The similar performance pattern can also be observed between BART and GPT, which suggests that encoder models are better at using long-term information, and decoder models are better at short-term information.

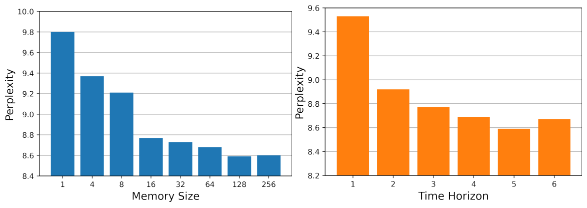


Figure 5: Effects of changing memory size (left) and time horizon (right).

5.4 Ablation Studies

We also evaluate the effect of varying memory sizes and back-propagation time horizons on PersonaChat dataset with a context length of 64. When varying the memory size, we set the time horizon to 5. In Figure 5, increasing the memory size has a significant improvement for perplexity until it reaches 128. When varying the time horizon, memory size is set to 128. In the right figure, the time horizon being 1 (gradients cannot flow through memory) achieved performance better than BART, suggesting that the memory after pre-training can capture history information. Increasing the time horizon to 2 can significantly improve the performance.

6 Conclusion

In conclusion, we introduce a new stateful memory-augmented Transformer encoder-decoder model that can preserve long dialogue history while being compatible with pre-trained encoder-decoder models. By incorporating a separate memory module with dual attention stream and residual gating mechanism, our model effectively interchanges information between the memory states and the pre-trained transformer. The experimental results have demonstrated the superiority of our method in terms of efficiency and performance, when comparing with other pre-trained models such as BART, GPT, and Longformer. For future work, we will enhance other existing language models with the stateful memory, expanding the range and capabilities of our memory-augmented transformer models.

Limitations

In our approach, we introduce additional pre-training as we need to initialize the memory module's weights. This is necessary as the additional pre-training enables the model to effectively preserve long dialogue history while building on top of pre-trained models such as BART. Note that the additional pre-training cost is only 0.125% compared to pre-training BART from scratch. After pre-training, our model is several times more efficient compared to the baselines.

Our work focuses on improving the efficiency of the encoder-decoder models. Many recent works (Tay et al., 2022; Soltan et al., 2022) show that encoder-decoder models may have competitive performance compared to GPT-3 and are much more efficient, which adds the value of our work. Also, casual decoder models can be easily transformed into non-causal decoder models, which make it possible to apply our method to the decoder-only models.

Another important thing to note is the difference in our work compared to retrieval-augmented models like the recent Unlimiformer (Bertsch et al., 2023) and LongMem (Wang et al., 2023). In general, there is no free lunch for memorization. Retrieval-augmented models normally require to store the historical encodings into memory and retrieve them later when needed. However, the storing process results in an increasing memory cost when there is more history. In contrast, our method has a constant memory cost which by default can process inputs of infinite length.

Ethical Considerations

In this work, we focused on the efficiency of the modeling. We pre-trained our model on a large corpus similar to BART. We used the existing filtered data to guarantee safety. However, there is still chance that offensive and toxic data are used during pre-training. Also, as dialogue models are becoming more efficient and powerful, they may be misused for scam, harassment, propaganda... We will address these problem in the future with existing techniques (Xu et al., 2020) to build safer dialogue models.

References

Daniel Adiwardana, Minh-Thang Luong, David R. So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, and Quoc V. Le. 2020. Towards a human-like open-domain chatbot. *CoRR*, abs/2001.09977.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Gary Cottrell, and Julian J. McAuley. 2021. Rezero is all you need: fast convergence at large depth. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence, UAI 2021, Virtual Event, 27-30 July 2021*, volume 161 of *Proceedings of Machine Learning Research*, pages 1352–1361. AUAI Press.

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

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R. Gormley. 2023. Unlimiformer: Long-range transformers with unlimited length input. *CoRR*, abs/2305.01625.

Aydar Bulatov, Yuri Kuratov, and Mikhail Burtsev. 2022. Recurrent memory transformer. In *Advances in Neural Information Processing Systems*.

Mikhail S. Burtsev and Grigory V. Sapunov. 2020. Memory transformer. *CoRR*, abs/2006.11527.

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

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2978–2988. Association for Computational Linguistics.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

Carl Gold. 2003. FX trading via recurrent reinforcement learning. In *2003 IEEE International Conference on Computational Intelligence for Financial Engineering, CIFEr 2003, Hong Kong, March 20-23, 2003*, pages 363–370. IEEE.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwinska, Sergio Gomez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John P. Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. 2016. Hybrid computing using a neural network with dynamic external memory. *Nat.*, 538(7626):471–476.

Matthew J. Hausknecht and Peter Stone. 2015. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposia, Arlington, Virginia, USA, November 12-14, 2015*, pages 29–37. AAAI Press.

Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc V. Le. 2022. Transformer quality in linear time. *CoRR*, abs/2202.10447.

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. 2022. Block-recurrent transformers. *CoRR*, abs/2203.07852.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. 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.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. 2022. cosformer: Rethinking softmax in attention. *CoRR*, abs/2202.08791.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. 2021. Recipes for building an open-domain chatbot. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 300–325. Association for Computational Linguistics.

Saleh Soltan, Shankar Ananthakrishnan, Jack FitzGerald, Rahul Gupta, Wael Hamza, Haidar Khan, Charith Peris, Stephen Rawls, Andy Rosenbaum, Anna Rumshisky, Chandana Satya Prakash, Mukund Sridhar, Fabian Triefenbach, Apurv Verma, Gokhan Tur, and Prem Natarajan. 2022. Alexatm 20b: Few-shot learning using a large-scale multilingual seq2seq model. *arXiv*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2440–2448.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Neil Houlsby, and Donald Metzler. 2022. Unifying language learning paradigms. *CoRR*, abs/2205.05131.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023. Augmenting language models with long-term memory. *CoRR*, abs/2306.07174.

Xuewei Wang, Weiyan Shi, Richard Kim, Yoojung Oh, Sijia Yang, Jingwen Zhang, and Zhou Yu. 2019. Persuasion for good: Towards a personalized persuasive dialogue system for social good. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5635–5649, Florence, Italy. Association for Computational Linguistics.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ronald J. Williams and Jing Peng. 1990. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Comput.*, 2(4):490–501.

Qingyang Wu, Zhenzhong Lan, Jing Gu, and Zhou Yu. 2020. Memformer: The memory-augmented transformer. *CoRR*, abs/2010.06891.

Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. 2022. Memorizing transformers. In *International Conference on Learning Representations*.

Jing Xu, Da Ju, Margaret Li, Y-Lan Boureau, Jason Weston, and Emily Dinan. 2020. Recipes for safety in open-domain chatbots. *CoRR*, abs/2010.07079.

Jing Xu, Arthur Szlam, and Jason Weston. 2022. Beyond goldfish memory: Long-term open-domain conversation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5180–5197, Dublin, Ireland. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2204–2213. Association for Computational Linguistics.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020. DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL 2020, Online, July 5-10, 2020*, pages 270–278. Association for Computational Linguistics.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 19–27. IEEE Computer Society.

A Different Model Variants

We evaluate different model variants to select the model with best memory effectiveness. We choose the text recall task for evaluation. The task is constructed as recalling previous text segment. Suppose we have an a document split into text segments $x_0, x_1, \ldots, x_t$. The encoder receives an input x_t at timestep t. The decoder needs to predict x_{t-1}. In this way, memory has to compress the previous information into the memory.

Memformer The first model is directly applying Memformer by adding the memory cross-attention layers to BART. The cross-attention layer is between the attention layer and the MLP layer. Below is the simplified formulation without showing the normalization:

$$\begin{aligned} H^l &= H^l + \text{Attn}(H^l) \\ H^l &= H^l + \text{CrossAttn}(H^l, M_t) \\ H^l &= H^l + \text{MLP}(H^l) \end{aligned}$$

Memformer + ReZero uses ReZero (Bachlechner et al., 2021) by adding a zero-initialized trainable weight α when adding the memory cross-attention layer, and therefore the model's output distribution will get updated smoothly.

$$\begin{aligned} H^l &= H^l + \text{Attn}(H^l) \\ H^l &= H^l + \alpha\,\text{CrossAttn}(H^l, M_t) \\ H^l &= H^l + \text{MLP}(H^l) \end{aligned}$$

MemBART + Shared weights A direct variant of our approach is sharing the weights between the memory module and the pre-trained Transformer. This is similar to append trainable prompting embeddings to the input sequence.

MemBART is our proposed approach. The main difference from Memformer is the memory module, where the memory reading and writing are handled with a separate Transformer. The information flow between the memory module and the pre-trained Transformer is achieved by the dual attention flow to minimally influence the original model distribution.

The detailed training hyper-parameters are shown in the Table 6. The back-propagation time horizon is set to 2 because it is sufficient for this task. The training takes approximately less than 12 hours to finish on one A6000 GPU.

Hyperparams	**All models**
Encoder Layers	6
Decoder Layers	6
Hidden size	768
Attention heads	12
Memory size	32
Context length	512
Batch size	8
Warm-up steps	1k
Learning rate	3e-5
Time horizon	2
Dropout	0.0
Weight decay	0.01
Maximum Update steps	100k

Table 6: Hyper-parameters for the text recall task.

B Sequence Denoising Pre-training Details

As mentioned, we use the same training objective as BART. Also, the pre-training corpus is selected to similar to BART. Since our model is highly based on BART, we use the same tokenization as BART. We filter out documents that are shorter than 512 tokens. Each document is split into segments with a window size of 512 and an overlap of 128 tokens.

Hyperparams	**MemBART-base**	**MemBART-large**
Encoder Layers	6	12
Decoder Layers	6	12
Hidden size	768	1024
Attention heads	12	16
Context length	512	512
Stride	128	128
mask ratio	0.3	0.3
permutation ratio	0.0	0.0
replace length	1	1
Batch size	32	32
Warm-up steps	5k	5k
Learning rate	3e-5	1e-5
Time horizon	6	6
Dropout	0.0	0.0
Weight decay	0.01	0.01
Update steps	100k	100k

Table 7: Hyper-parameters for training MemBART-base and MemBART-large.

We pre-train our models with the hyper-parameters shown in Table 7. Note that training 100k steps only takes about 0.125% of the original pre-training cost of BART. The pre-training for MemBART-base takes about 4 day on four A6000 GPUs. The pre-training for MemBART-large takes about 8 days on four A6000 GPUs. We also train a Memformer+ReZero model for comparison using the same setting as MemBART base.

B.1 Batch Processing and Dispatch

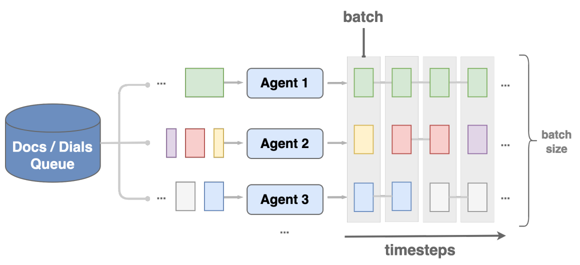


Figure 6: The illustration of how documents or dialogues are processed and batched.

As batches are temporal-dependent in our paradigm, we implement a batch dispatcher to efficiently process the documents and dialogues as shown in Figure 6. In this paradigm, a number of the agents whose size is equal to the batch size share the same data queue to fetch documents. When finished processing a document, the agent pops a new document from the shared queue, and it splits the document into text segments or utterances to output one context input at each timestep. The agent also handles the reset signal and token padding when documents have varied lengths. All the agents are synchronized, and the batch is collected at each timestep. This paradigm simplifies the preservation of the temporal order in batches and the alignment between varied-length documents or dialogues. We use this batch dispatcher across all our experiments.

C Multi-Session Chat Full Experiments

We have shown the full experiments on multi-session chat under different settings. Latency is measured with dummy inputs based on the context length during training. The label's length is fixed to 128, and the batch size is 4. We report the average of 10 runs and the corresponding variance. We select the best models based on the validation set and then evaluate them on the test set. The validation results are shown in Table 9. The test results are shown in Table 10.

One observation is that Longformer would pad the sequence to the multiples of $1,024$ due to the sparse attention mechanism. This behavior results in very slow performance when the context size is small.

Another observation is that for later sessions, especially Session 4 and 5, history information matters. For Session 5, BART base gets 4.5% performance loss when the context size is truncated to 128. BART large gets 6.5% performance loss due to truncation. In contrast, as MemBART has memory, the performance difference is smaller when using different context sizes.

D The Number of Parameters

Models	#Parameters
BART base	139M
MemBART base	183M
BART large	406M
MemBART large	558M

Table 8: The number of parameters for BART and MemBART.

We show the number of parameters of BART and MemBART in Table 8. Since MemBART incorporates additional memory module. It is slightly larger than its counterpart BART model. But as a trade-off, MemBART is much faster than BART.

E GPU Memory Efficient Training

Memformer proposed a variant of gradient checkpointing to efficiently train this type of stateful models. The GPU memory consumption scales linearly with the back-propagation time horizon because it requires unrolling the computation graph as equal to the number of timesteps.

We applied this efficient training algorithm for the MemBART large model model with time horizon 6. Without efficient back-propagation method, it would consume a large amount of GPU memory, which makes the training infeasible. MRBP traverses the critical path in the computational graph during the forward pass and recomputes the partial computational graph for the local timestep during the backward pass. The algorithm takes an input with a rollout $x_t, x_{t+1}, \dots, x_T$ and the previous memories $M_t, M_{t+1}, \dots, M_T$ if already being computed. It then obtains each timestep's memory and stores those memories in the replay buffer. The following is the algorithm details:

Algorithm 1: BP through Memory Replay

```
Input: rollout=[$x_t, x_{t+1}, \dots, x_T$]: a list
       containing previous inputs
       memories=[$M_t, M_{t+1}, \dots, M_T$]:
       memory from the previous
  ▷ Initialize a list for
    back-propagation
1  replay = list([$M_t$])
  ▷ Forward pass & no gradient
2  for $t = t, t+1, \dots, T-1$ do
3      $M_{t+1}$, _ = Model($x_t$, $M_t$)
4      replay.append($M_{t+1}$)
5  end
  ▷ Backward pass with gradient
6  $\nabla M_{t+1} = 0$
7  for $t = T, T-1, \dots, t+1, t$ do
       ▷ Recompute
8      $M_{t+1}$, $O_t$ = Model($x_t$, $M_t$, $r_t$)
9      $loss$ = $L(O_t)$
10     $loss$.backward()
11     $M_{t+1}$.backward($\nabla M_{t+1}$)
12     $\nabla M_{t+1}$ = $\nabla M_t$
13 end
  ▷ Update the memories
14 memories = Buffer
15 memories.pop()
```

Base Models	**Context**	**Latency (ms)**	**Total**	**Session 1**	**Session 2**	**Session 3**	**Session 4**	**Session 5**
BART base	128	$16.41_{\pm 0.73}$	12.72	10.84	13.19	13.15	13.17	12.77
BART base	256	$22.12_{\pm 0.89}$	12.50	10.77	12.85	12.89	12.96	12.58
BART base	512	$36.80_{\pm 1.17}$	12.33	10.71	12.61	12.67	12.81	12.43
BART base	1,024	$64.65_{\pm 0.72}$	12.22	10.69	12.46	12.38	12.77	12.38
Longformer base	256	$110.07_{\pm 0.28}$	12.55	10.78	12.92	12.93	13.07	12.57
Longformer base	512	$113.73_{\pm 3.16}$	12.35	10.73	12.64	12.66	12.87	12.40
Longformer base	1,024	$115.96_{\pm 0.25}$	12.20	10.67	12.55	12.46	12.65	12.26
Longformer base	2,048	$227.75_{\pm 0.13}$	12.16	10.69	12.54	12.46	12.58	12.15
MemBART base (64)	128	$17.23_{\pm 1.19}$	12.17	10.6	12.60	12.54	12.55	12.14
MemBART base (64)	256	$29.39_{\pm 0.73}$	12.06	10.59	12.40	12.36	12.47	12.09
MemBART base (64)	512	$59.73_{\pm 0.66}$	11.95	10.57	12.28	12.22	12.33	11.98
MemBART base (128)	128	$20.42_{\pm 1.47}$	12.12	10.6	12.50	12.45	12.51	12.14
MemBART base (128)	256	$32.09_{\pm 0.18}$	11.96	10.49	12.29	12.28	12.37	11.97
MemBART base (128)	512	$66.70_{\pm 1.83}$	11.86	10.50	12.15	12.14	12.27	11.89
MemBART base (256)	128	$26.56_{\pm 0.57}$	12.11	10.58	12.51	12.43	12.47	12.13
MemBART base (256)	256	$40.92_{\pm 0.63}$	12.00	10.50	12.35	12.34	12.40	12.01
MemBART base (256)	512	$75.54_{\pm 0.14}$	11.83	10.47	12.11	12.10	12.24	11.86
Large Models	**Context**	**Latency (ms)**	**Total**	**Session 1**	**Session 2**	**Session 3**	**Session 4**	**Session 5**
GPT2-12	128	$16.24_{\pm 1.13}$	14.17	12.87	14.57	14.5	14.51	14.03
GPT2-12	256	$30.80_{\pm 0.48}$	13.91	12.70	14.20	14.23	14.25	13.81
GPT2-12	512	$65.77_{\pm 0.74}$	13.76	12.68	14.03	14.02	14.11	13.67
GPT2-12	1,024	$149.05_{\pm 0.38}$	13.33	12.66	14.04	13.82	13.26	12.71
GPT2-24	128	$42.39_{\pm 2.50}$	11.91	11.15	12.17	12.10	12.10	11.83
GPT2-24	256	$81.80_{\pm 0.18}$	11.66	10.98	11.83	11.83	11.86	11.62
GPT2-24	512	$172.43_{\pm 0.12}$	11.52	10.99	11.63	11.64	11.72	11.48
GPT2-24	1,024	$395.84_{\pm 0.64}$	11.43	10.96	11.59	11.48	11.62	11.37
BART large	128	$45.37_{\pm 1.31}$	10.42	9.31	10.75	10.61	10.68	10.44
BART large	256	$63.79_{\pm 0.40}$	10.15	9.17	10.35	10.34	10.40	10.20
BART large	512	$103.20_{\pm 2.40}$	10.00	9.22	10.12	10.12	10.28	10.03
BART large	1,024	$190.79_{\pm 0.29}$	9.87	9.20	10.03	9.91	10.09	9.90
Longformer large	256	$316.42_{\pm 2.37}$	10.25	9.28	10.43	10.41	10.55	10.30
Longformer large	512	$322.68_{\pm 1.74}$	10.06	9.24	10.18	10.15	10.38	10.13
Longformer large	1,024	$334.87_{\pm 5.54}$	9.90	9.20	10.06	9.95	10.15	9.92
Longformer large	2,048	$655.19_{\pm 5.25}$	9.87	9.23	10.09	9.90	10.04	9.89
MemBART large (128)	128	$59.51_{\pm 0.91}$	9.99	9.17	10.19	10.14	10.22	10.02
MemBART large (128)	256	$102.42_{\pm 2.07}$	9.92	9.08	10.10	10.06	10.15	9.95
MemBART large (128)	512	$197.79_{\pm 4.85}$	9.79	9.08	9.90	9.88	10.03	9.84

Table 9: Complete Multi-Session Chat results on the validation set. Latency is measured with the average of 10 runs.

Base Models	**Context**	**Latency (ms)**	**Total**	**Session 1**	**Session 2**	**Session 3**	**Session 4**	**Session 5**
BART base	128	$16.41_{\pm 0.73}$	13.05	10.99	12.52	13.18	13.65	14.02
BART base	256	$22.12_{\pm 0.89}$	12.83	10.94	12.29	12.97	13.37	13.78
BART base	512	$36.80_{\pm 1.17}$	12.68	10.92	12.14	12.77	13.19	13.61
BART base	1,024	$64.65_{\pm 0.72}$	12.53	10.81	11.93	12.50	13.10	13.55
Longformer base	256	$110.07_{\pm 0.28}$	12.87	10.78	12.36	13.02	13.45	13.88
Longformer base	512	$113.73_{\pm 3.16}$	12.69	10.77	12.19	12.79	13.22	13.67
Longformer base	1,024	$115.96_{\pm 0.25}$	12.55	10.74	12.12	12.59	13.02	13.48
Longformer base	2,048	$227.75_{\pm 0.13}$	12.52	10.76	12.13	12.59	12.93	13.42
MemBART base (64)	128	$17.23_{\pm 1.19}$	12.42	10.72	11.95	12.52	12.93	13.23
MemBART base (64)	256	$29.39_{\pm 0.73}$	12.34	10.66	11.86	12.46	12.84	13.16
MemBART base (64)	512	$59.73_{\pm 0.66}$	12.23	10.66	11.78	12.32	12.66	13.02
MemBART base (128)	128	$20.42_{\pm 1.47}$	12.41	10.72	11.95	12.52	12.88	13.23
MemBART base (128)	256	$32.09_{\pm 0.18}$	12.25	10.62	11.76	12.37	12.71	13.06
MemBART base (128)	512	$66.70_{\pm 1.83}$	12.15	10.63	11.67	12.23	12.57	12.97
MemBART base (256)	128	$26.56_{\pm 0.57}$	12.38	10.67	11.90	12.51	12.86	13.20
MemBART base (256)	256	$40.92_{\pm 0.63}$	12.25	10.59	11.76	12.38	12.74	13.07
MemBART base (256)	512	$75.54_{\pm 0.14}$	12.09	10.57	11.62	12.18	12.53	12.90
Large Models	**Context**	**Latency (ms)**	**Total**	**Session 1**	**Session 2**	**Session 3**	**Session 4**	**Session 5**
GPT2-12	128	$16.24_{\pm 1.13}$	14.36	12.91	13.80	14.43	14.79	15.22
GPT2-12	256	$30.80_{\pm 0.48}$	14.13	12.80	13.57	14.21	14.53	14.93
GPT2-12	512	$65.77_{\pm 0.74}$	13.99	12.81	13.45	14.03	14.33	14.78
GPT2-12	1,024	$149.05_{\pm 0.38}$	13.56	12.82	13.48	13.84	13.53	13.82
GPT2-24	128	$42.39_{\pm 2.50}$	12.03	11.17	11.52	12.07	12.30	12.62
GPT2-24	256	$81.80_{\pm 0.18}$	11.78	11.02	11.28	11.82	12.04	12.36
GPT2-24	512	$172.43_{\pm 0.12}$	11.65	11.07	11.14	11.66	11.86	12.20
GPT2-24	1,024	$395.84_{\pm 0.64}$	11.56	11.03	11.12	11.52	11.75	12.11
BART large	128	$45.37_{\pm 1.31}$	10.61	9.50	10.13	10.68	10.94	11.29
BART large	256	$63.79_{\pm 0.40}$	10.37	9.38	9.86	10.44	10.67	11.02
BART large	512	$103.20_{\pm 2.40}$	10.23	9.44	9.71	10.26	10.52	10.85
BART large	1,024	$190.79_{\pm 0.29}$	10.10	9.41	9.64	10.06	10.36	10.68
Longformer large	256	$316.42_{\pm 2.37}$	10.43	9.34	9.95	10.52	10.75	11.11
Longformer large	512	$322.68_{\pm 1.74}$	10.28	9.37	9.77	10.32	10.57	10.92
Longformer large	1,024	$334.87_{\pm 5.54}$	10.13	9.42	9.66	10.11	10.38	10.72
Longformer large	2,048	$655.19_{\pm 5.25}$	10.05	9.43	9.60	10.04	10.27	10.60
MemBART large (128)	128	$59.51_{\pm 0.91}$	10.17	9.22	9.61	10.24	10.47	10.85
MemBART large (128)	256	$102.42_{\pm 2.07}$	10.09	9.20	9.65	10.09	10.38	10.72
MemBART large (128)	512	$197.79_{\pm 4.85}$	9.99	9.22	9.51	10.03	10.23	10.58

Table 10: Complete Multi-Session Chat results on the test set. Latency is measured with the average of 10 runs.