MemReasoner: A Memory-augmented LLM Architecture for Multi-hop Reasoning

Ching-Yun Ko[1*], **Sihui Dai**[1,2*†], **Payel Das**[1*],
Georgios Kollias[1], **Subhajit Chaudhury**[1], **Aurelie Lozano**[1]
[1]IBM AI Research, [2]Princeton University
daspa@us.ibm.com

Abstract

Recent benchmarks suggest that there remains significant room to improve large language models' ability to robustly reason across facts distributed in extremely long documents. In this work, we propose MemReasoner, a new memory-augmented LLM architecture that is trained to perform temporal reasoning, along with multiple computational steps, over the context stored in the memory. Experiments show that MemReasoner trained on the core reasoning facts generalizes better, when compared to off-the-shelf large language models and existing recurrent models, on a test distribution where the required facts are scattered across long natural text up to 128k tokens. Further, MemReasoner demonstrates robust reasoning performance relative to the baselines, when the answer distribution in test samples differs from that in the training set.

1 Introduction

Transformer-based large language models (LLMs) have recently shown impressive performance in many natural language processing (NLP) tasks, including machine translation, question answering, and reading comprehension, demonstrating signature of general reasoning abilities. However, when restricted to individual NLP reasoning benchmarks, particularly those that require logical reasoning, current LLMs typically perform poorly despite efforts to improve accuracy through prompt engineering (Wei et al., 2022; Min et al., 2022). As such, more evidence seems to support the hypothesis that powerful LLMs often learn statistical features and correlations to simulate reasoning rather than performing true reasoning (Ruder, 2021).

The recently introduced BABILong benchmark further establishes this point, as it is designed to test LLM's ability to reason across facts distributed in extremely long documents (Kuratov et al., 2024). BABILong is developed based on the bAbi benchmark Weston et al. (2015), which is composed of 20 reasoning tasks. These include fact chaining, simple induction, deduction, counting, and handling lists/sets (Weston et al., 2015). This set of tasks was designed as prerequisites for any system that aims to having a conversation with a human. BABILong further introduces irrelevant natural text from the PG19 book corpus Rae et al. (2019) into the original context to make it artificially longer and include distracting text, while the underlying reasoning task remains the same. For examples of the task samples in BABILong, see Figure 1. Experiments with popular transformer-based LLMs shows that present days' transformer-based language models effectively utilize only 10-20% of the context and their performance declines sharply with increased reasoning complexity. Retrieval-augmented generation with LLMs at best can provide 60% accuracy for a simple QA task that requires extracting single evidence from the context. Interestingly, a memory-augmented transformer architecture, namely Recurrent Memory Transformers (RMT) (Bulatov et al., 2022) shows the highest performance on BABILong benchmark; suggesting that long-term recurrent memory of the context

[*]Equal contribution.
[†]Work done during internship at IBM Research.

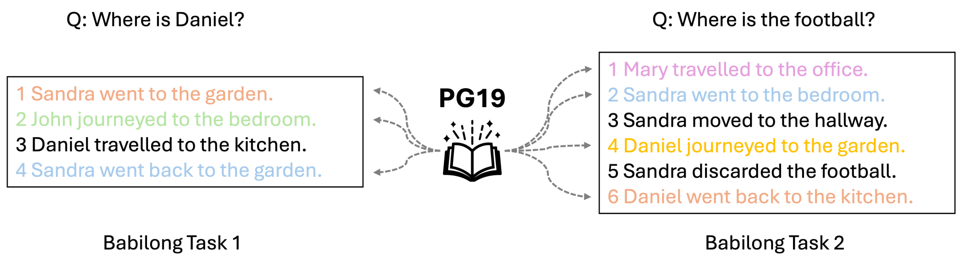

Q: Where is Daniel?
1 Sandra went to the garden.
2 John journeyed to the bedroom.
3 Daniel travelled to the kitchen.
4 Sandra went back to the garden.
Babilong Task 1
PG19
Q: Where is the football?
1 Mary travelled to the office.
2 Sandra went to the bedroom.
3 Sandra moved to the hallway.
4 Daniel journeyed to the garden.
5 Sandra discarded the football.
6 Daniel went back to the kitchen.
Babilong Task 2

Figure 1: Examples of BABILong tasks.

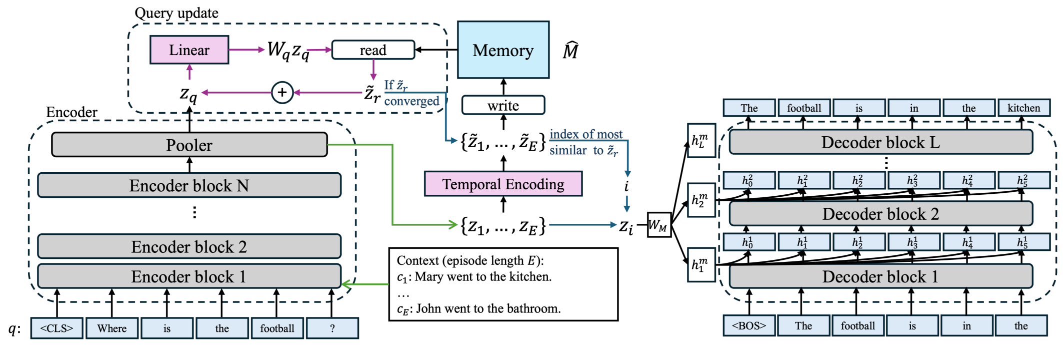
helps. However, as we will demonstrate in this work, RMT needs to be trained on samples with longer context from BABILong's distribution in order to perform well and the model fails to generalize without access to long context examples during training.

In this work, we provide an alternative language model architecture that is designed to naturally handle recurrent processing over long context *that is not seen by the model during training*. Our goal is to provide a more effective and robust solution for handling multi-hop generative QA tasks, which require the model to gather, relate, and reason over disjoint pieces of information from the *unseen* long context to generate an answer. Towards this goal, we propose a memory-augmented LLM architecture enhanced with two basic operations: (i) explicit learning of temporal orders of facts/events present within the context, and (ii) mechanism for iteratively reading from the context and updating the query accordingly. We refer to this new architecture as MemReasoner.

The backbone memory-augmented LLM used in this study is Larimar (Das et al., 2024), which is trained such that the latent encodings of a set of facts, referred as an episode, are written to a memory module. For a given query, the readout from this episodic memory module conditions the generation of the decoder, which is achieved by learning a differentiated attention to the readout during training. During inference, memory is dynamically updated by solving a linear system of equations, which is efficiently done via computing matrix pseudoinverse rather than gradient backpropagation. The memory mechanisms in Larimar assume order invariance of samples within an episode and support only single time read over the episode, which are insufficient for the architecture to handle more complicated tasks like multi-hop question-answering (QA). We note that our approach could in principle be used in conjunction with other LLMs augmented with an episodic memory module beyond Larimar.

Here, we extend the basic episodic memory module to act as a reasoning module by introducing a recurrent network, such as a GRU, which is tasked to capture the sequence of events/facts in the context written to the memory. This step prepares the inputs to the reasoning module with a structured understanding of their temporal relationships, which is critical for reasoning over time-varying information. For example, in the sample shown in Fig 1 (right), understanding that "Sandra moved to the hallway" happens before "Sandra discarded the football" is crucial to answer the question "Where is the football?" (Answer: hallway). Around the reasoning module, we further enable iterative reads from the memory to "hop" between supporting facts and update the query accordingly. This operation allows the model to dynamically retrieve and refine information across multiple computational steps performed over the context episode. These two operations around the latent memory allows in-depth deliberation over the context, which is then used by the decoder for generation. Our main contributions are:

- A novel memory-augmented LLM architecture, namely MemReasoner, which is equipped with temporal processing and iterative read over the context written to an episodic memory module.
- Evaluation of MemReasoner on the single-hop (Task 1) and two-hop (Task 2) QA tasks (see Figure 1) using the challenging BABILong benchmark, establishing that the proposed architecture can generalize to long context that is unseen during training, whereas current vanilla transformer-based LLMs struggle and alternative recurrent models fail to generalize.
- Experiments showing that the proposed MemReasoner architecture indeed learns multi-step processing over the context to solve the QA task, as evident by its robust performance when the answers in the training data differ from those in the test samples within the same task, and when the model trained on bABi task 2 is tested on longer BABILong task 1 samples.



Query update
Linear
$W_q z_q$
read
Memory
$\hat{M}$
z_q
$\tilde{z}_r$
If $\tilde{z}_r$ converged
write
Encoder
Pooler
Encoder block N
Encoder block 2
Encoder block 1
$\{\tilde{z}_1, ..., \tilde{z}_E\}$
index of most similar to $\tilde{z}_r$
Temporal Encoding
i
$\{z_1, ..., z_E\}$
z_i
W_M
Context (episode length E):
c_1: Mary went to the kitchen.
...
c_E: John went to the bathroom.
q: <CLS> Where is the football ?
h_L^m
h_2^m
h_1^m
Decoder block L
Decoder block 2
Decoder block 1
The football is in the kitchen
<BOS> The football is in the

Figure 2: A diagram of the pipeline for reasoning with MemReasoner. q denotes the query, $c_1, ..., c_E$ denotes the context for answering the query. z_q denotes the encoding of the query while $\{z_1, ..., z_E\}$ denote encodings of each line of the context. We use $\tilde{z}$ to denote temporally encoded latents.

2 Multi-step Reasoning with MemReasoner

2.1 Preliminary

Let $\mathcal{X}$ be the LM input space, $\mathcal{Z}$ be the latent space, and $\mathcal{Y}$ be the LM output space. Larimar (Das et al., 2024) features an encoder e that maps an input to an embedding $z \in \mathcal{Z} \subseteq \mathbb{R}^D$, and a memory module $\mathcal{M}$. The memory M is adaptable in the sense, that it allows "write" and "read" operations as episodes (aka, contexts C, where each context is comprised of E sentences) arrive, i.e., $\hat{M} = write(M, z), z_{\text{read}} = read(\hat{M}, z)$, wherein $\hat{M}$ is the updated memory after an write. And, a decoder d that performs generations conditioned on the memory readout z_{read}.

Now, suppose one is given an input context $C = \{c_1, ..., c_E\}$ with E denoting the length of the context, and the target task is to answer a question q conditioned on the given context C. To approach the task within Larimar framework, the input, both context C and query q, are encoded to their latents ($z_1, \dots, z_E$ and z_q) via the encoder e. Next, let M_0 be the initial memory, write the context to the memory via the $write$ operation mimicking posterior updates in Bayesian inference (Pham et al., 2022), i.e., $\hat{M} = (Z_\xi M_0^\dagger)^\dagger Z_\xi$, where $Z_\xi = [z_1 + \xi_1, z_2 + \xi_2, \dots, z_E + \xi_E]$ and $\xi_i \sim \mathcal{N}(0, \sigma_\xi^2 I)$. The memory update is done via finding least-square solutions to linear systems by estimating matrix pseudoinverses, indicated by † hereafter. Then, the $read$ operation translates the query embedding from the lens of the encoded memory to a query readout z_r via $z_r = (z_q \hat{M}^\dagger + \eta)\hat{M}$, where $\eta \sim \mathcal{N}(0, \sigma_\eta^2 I)$. Lastly, the decoder d decodes the query q conditioned on the readout by using a *learnable* broadcasting parameter W_M that casts z_r to each decoder layer and obtains h_k^m that serves as the past key values for $k = 1, \dots, L$, where L is the number of layers in the decoder. We use this memory-augmented LLM architecture and the operations as backbone for MemReasoner, due to its memory and space-efficient read/write abilities and demonstrated generalizability at test-time. It is worth mentioning the earlier works on memory-augmented neural nets, which use a recurrent neural net together with an external memory, have investigated ideas like temporal feature learning and iterative hops over context, for example see (Weston et al., 2014; Sukhbaatar et al., 2015). However, to our knowledge, this is the first study to enable those operations around the explicit episodic memory of a transformer-based LLM during training and test the resulting model's generalizability on a long-context reasoning benchmark like BABILong.

2.2 Memory with Temporal Order

Recall, the latent encoding of facts $\{z_1, ..., z_E\}$ within a context episode C are written in the memory M in an order-invariant manner. However, many multi-step reasoning tasks require some notion of temporal context. For example, when answering "where is John?" in the context of ". . . John is in the bathroom. . . . John goes to the garden." (". . ." denotes irrelevant facts), there should be a mechanism in place to guarantee that the memory encodes the correct temporal order of the facts, and the readout should reflect "John goes to the garden." as the supporting fact instead of "John is in the bathroom.".

To introduce some temporal notion within the context, in MemReasoner we introduce a temporal encoding module $\mathcal{P}$ that transforms *un-ordered* fact latents $\{z_1, ..., z_E\}$ within a context episode to their *ordered* counterparts $\{\tilde{z}_1, ..., \tilde{z}_E\}$. The temporal encoding module is generic and allows any structure featuring sequentiality within context. In practice, we investigate two general types of encoding methods, *un-parameterized* methods such as Sinusoidal Positional Encoding and *parameterized* methods such as GRUs.

Positional Encoding. We compute positional encodings for each line of context within the episode by using sine and cosine functions similar to Vaswani et al. (2017). Additionally, we experiment with positional encoding which assigns encodings starting from the last element of the episode. The structure ensures that for contexts of different length, the last lines of the contexts are encoded similarly, which is useful for QA tasks in which the most recent information is more relevant for answering the question. Finally, to convert $\{z_1, ..., z_E\}$ to $\{\tilde{z}_1, ..., \tilde{z}_E\}$ with positional encodings, we add the computed positional encodings to the input.

GRU. We also investigate learnable encodings via a bidirectional GRU unit. For these, we treat $\{z_1, ..., z_E\}$ as the sequence passed as input into the GRU and simply let $\{\tilde{z}_1, ..., \tilde{z}_E\}$ be the sequential outputs of the GRU.

These ordered context embeddings $\{\tilde{z}_1, ..., \tilde{z}_E\}$ are then written to memory via Larimar's $write$ operation.

2.3 Iterative Read And Query Update

A typical multi-step reasoning task often inherently requires "hops" between facts until the final solution is found. Additionally, the query embedding can be updated accordingly to reflect the most recent hop.

In order to perform hopping between facts, we first recall the three key components interacting with the memory module $\mathcal{M}$, the fact embeddings $(\{z_1, \ldots, z_E\})$ within a context episode, the query embedding z_q, and the memory readout z_r. Let us further consider M stores facts that have been ordered temporally $\{\tilde{z}_1, ..., \tilde{z}_E\}$.

To enable *iterative read*, we pass z_q through a linear layer to obtain $\hat{z_q}$=$W_q z_q$ before the $read$ operation from the memory, where $W_q \in \mathbb{R}^{D \times D}$ is a learnable parameter that absorbs the scale changes introduced by the position encoding in the memory. Specifically, different from Section 2.1, here we have $z_r = (\hat{z_q}\hat{M}^\dagger + \eta)\hat{M}$.

To *update the query*, we first update the query latent and let $z_q \leftarrow z_q + \alpha \cdot z_r$, where $\alpha \in \mathbb{R}$ is a hyperparameter to balance the load from the previous readout. The updated query is then fed into the memory module for another $read$ operation to obtain a new $\tilde{z}_r$. The query update procedure is repeated until the readout converges (i.e. $||\tilde{z}_r^t - \tilde{z}_r^{t+1}||_2 < \tau$ where $\tilde{z}_r^t$ denotes the readout at time t and τ is a hyperparameter) or until it reaches a fixed number of maximum iterations.

2.4 Full Workflow

Now that we have discussed all components of MemReasoner, we elaborate the full pipeline in the following and provide a visualization in Figure 2.

Consider an input context $C = \{c_1, ..., c_E\}$, a question q, an encoder e, a temporal encoding module $\mathcal{P}$, an initial memory module $\mathcal{M}$, and a decoder d. We first encode the context C and query q to their latents, $z_1, \ldots, z_E$ and z_q, via encoder e. Then, we follow Section 2.2 and transform $z_1, \ldots, z_E$ to $\tilde{z}_1, ..., \tilde{z}_E$. Next, we write the ordered context $\tilde{z}_1, ..., \tilde{z}_E$ to the memory and obtain $\hat{M}$. Subsequently, we read using the query latent from the memory and perform query and read updates according to Section 2.3. After we have obtained a $\tilde{z}_r$ as a final readout which does not undergo update anymore, we map $\tilde{z}_r$ to the corresponding unordered encoding in M. This is because we only want the additional position information to be used when locating the most relevant contexts, but not during the decoding - if being fed to the decoder, the decoder may overfit to the ordering information in the latents. We do this by first finding the index of the most similar ordered latent encoding $i = \arg\min_{j \in \{1,...,E\}} ||\tilde{z}_r - \tilde{z}_j||_2$ and then obtaining the corresponding encoding z_i from the unordered encodings (prior to undergoing temporal encoding in Figure 2) $\{z_1...z_E\}$. Lastly, the decoder d decodes the prompt P_a given for answer generation conditioned on z_i. We provide the full pseudocode in Algorithm 1 in the appendix.

2.5 Training Objectives

Let $\mathcal{D}_{\text{reason}}$ denote the reasoning data distribution while $\mathcal{D}_{\text{pretrain}}$ denotes the pretraining data distribution. Each sample from $\mathcal{D}_{\text{reason}}$ is of the form (q, C, S, a) where q is the query, $C = \{c_1, ..., c_E\}$ are the facts in the context, S is a set of indices corresponding to supporting facts (we will use S_i to denote the ith supporting fact index in S), and a is the answer. Meanwhile the pretraining distribution corresponds to a generic corpus, e.g. Wikipedia. Let e denote the encoder, d denote the decoder, t denote temporal encoding, $\tilde{z}_r^i$ denote the ith temporally encoded readout from iterative reading with $\tilde{z}_r^0 = q$, z_r^i represent the unordered encoding corresponding to the ith ordered readout, and P_a and P_s denote the prompts for generating the answer and supporting fact respectively. To train the model, we utilize the following loss function in Equation 1.

$$
\begin{aligned}
L = \mathbb{E}_{(q,C,S,a)\sim\mathcal{D}_{\text{finetune}}} \Bigg[& \underbrace{\mathbb{E}_{z_r^{|S|}\sim p(z_r^{|S|}|q,M,\tilde{z}_r^0\ldots\tilde{z}_r^{|S|-1})} \ln p(a|z_r^{|S|}, P_a)}_{\text{reconstruction of answer}} \\
& + \alpha \sum_{i=1}^{|S|} \underbrace{\mathbb{E}_{z_r^i\sim p(z_r^i|M,\tilde{z}_r^0\ldots\tilde{z}_r^{i-1})} \ln p(c_{S_i}|z_r^i, P_s)}_{\text{reconstruction of supporting facts}} + \beta \sum_{s\in S} \underbrace{\ln p(d(e(c_s)))}_{\text{autoencoding of supporting fact}} \\
& + \underbrace{\delta \sum_{i=1}^{|S|} \mathbb{E}_{\tilde{z}_r^i\sim p(\tilde{z}_r^i|q,M,\tilde{z}_r^0\ldots\tilde{z}_r^{i-1})} \ell_{\text{order}}(\tilde{z}_r^i, S_i)}_{\text{ordering loss}} \Bigg] + \rho\, \underbrace{\mathbb{E}_{x\sim\mathcal{D}_{\text{pretrain}}} \ln p(d(e(x)))}_{\text{autoencoding of pretraining dataset}}
\end{aligned}
\tag{1}
$$

α, β, δ and ρ are hyperparameters controlling regularization strength and ℓ_{order} is given by

$$
\begin{aligned}
v(z_r) &= \text{softmax}([-||t(e(c_1)) - z_r||_2, ..., -||t(e(c_E)) - z_r||_2]^\intercal) \\
\ell_{\text{order}}(z_r, s) &= -\ln v(z_r)_s
\end{aligned}
\tag{2}
$$

The first and second terms correspond to the reconstruction loss of the answer and the supporting fact(s) with respect to the corresponding prompt for obtaining the answer P_a and final readout, the third and the fifth terms correspond to the autoencoding loss of the supporting fact(s) and pretraining data. The fourth term is a loss for encouraging the index of the most similar entry (by l2 distance) to the ordered readout at each iteration to match the index of the supporting fact through computing the cross entropy.

3 Experimental Details and Results

3.1 Datasets and Data Pre-processing

We here utilize tasks 1 and 2 from the synthetic bAbi benchmark as our testbed. These datasets were prepared by synthesizing relations among characters and objects across various locations, each represented as a fact, such as "Mary traveled to the garden". Task 1 requires performing a single hop to find answer, whereas task 2 requires gathering two supporting facts in the right order (see Fig 1). These single to multi-hop QA tasks from BABILong benchmark together provide a controlled setting for evaluating LLMs' ability to reason over long context, where the difficulty of the task can be varied by changing the length of irrelevant text. The nature of this benchmark, where the synthetic sentences corresponding to the actual reasoning task are hidden inside irrelevant but lengthy naturally occurring text, keeps it at a low risk of data contamination to training sets of todays' LLMs. And finally, BABILong leaderboard shows tasks 1 and 2, while being simple enough, are challenging enough for off-the-shelf LLMs to solve.

We finetune MemReasoner separately on original bAbi task 1 and task 2 training split, each consisting of 10k samples (Weston et al., 2015). We then evaluate on the test set of the corresponding task from bAbi as well as from BABILong (Kuratov et al., 2024), in which the core reasoning facts from bAbi is distributed over arbitrarily long documents. Here we benchmark MemReasoner on BABILong test samples of up to 128k tokens.

For preprocessing bAbi data, we treat each training sample comprised of multiple facts as a single context episode, and individual sentence within that context as an instance within that episode. Each fact within an episode contains up to 64 tokens.For BABILong and for Wikipedia, if sentences are longer than 64 tokens, we split the sentences at multiples of 64 tokens.

We initiate MemReasoner finetuning from Larimar checkpoint pretrained on Wikitext (obtained by following the training protocol described in Das et al. (2024)), which uses a Bert-large as the encoder and a GPT2-large as the decoder. The number of parameters in MemReasoner is 1.4B. The slot size in the memory is 512. During finetuning, we randomly sample a batch of pretraining data (Wikipedia) of the same size as the batch of finetuning data (bAbi) for computing the autoencoding loss on the pretrain dataset of 2M samples. We generate the answer to the question by passing a prompt to the decoder (i.e. in the case of bAbi Task1-2, the prompt has the from "<BOS> X is in the" where X denotes subject of the query).

We train MemReasoner models for 200 epochs using Adam optimizer with learning rate 5e-6. We set batch size to be 10. Additionally, we set query update parameter $\alpha = 1$. The maximum episode length varies from 14 (bAbi Task 1) to 72 (bAbi Task 2). Which means that MemReasoner has been exposed to a maximum of 90 and 573 tokens during finetuning on task 1 and task 2, respectively, whereas at test-time the model is exposed to contexts that are up to 128k tokens long. Since bAbi Task 1 is a single hop task, we do not perform query update during either training or inference. When fine-tuning on bAbi Task 2, we perform a fix number of 2 hop (equivalent to 1 query update) during the training. With bAbi Task 2 fine-tuned MemReasoner, we re-use the "2 hop" setting at inference on all tasks, including bAbi Task 2 and BABILong Task1/2. We consistently use query update parameter α=8 throughout our experiments and include an ablation study on α in the appendix. Due to the page limit, we also defer ablation studies on the episodic memory, temporal encoding schemes, and the number of training epochs to the appendix.

3.2 Baseline Methods

Off-the-shelf Baselines. We show published results from Yang et al. (2023) obtained using GPT-3 (175 B parameters) as an off-the-shelf baseline, with few-shot and chain-of-thought prompting, for comparison with MemReasoner on original bAbi test set. We also report performances of a recurrent memory transformer-0.77B and of a Mamba-1.4B model, which we fine-tune on bAbi samples, on bAbi test set. For BABILong benchmarking, we include the following models from BABILong leaderboard: (1) Meta-Llama-3-8B-Instruct with a 8K context window size, (2) Phi3-mini-128k-instruct – a long-context LLM consisting of 3.8B parameters and a 128k token long context window, and (3) Llama3-ChatQA-1.5-8B with a nvidia/dragon-multiturn-query-encoder – a RAG framework.

Fine-tuned Baselines. We add RMT-137M and Mamba-130M performances from BABILong leaderboard, which has been finetuned on a set of samples that belongs to the same distribution as BABILong (with PG19 padding) but is not included in BABILong benchmarking test set. These models were finetuned by using a curriculum schedule that progressively increases sequence lengths: 1, 2, 4, 6, 8, 16 and 32 segments (Kuratov et al., 2024).

We further benchmark RMT and Mamba models finetuned on bAbi on BABILong test samples. The goal is to figure out if those alternative recurrent models perform well on BABILong leaderboard due to their true learning ability of the underlying task or due to their exposure to BABILong samples during finetuning. We fine-tune off-the-shelf RMT(0.14b/0.77b) and Mamba (0.13b/1.4b) models using the next token prediction loss on bAbi Task 1 and 2 separately till the testing accuracy on the task is sufficiently high (near 100%). In practice, we use 5 epochs to reach above 99% accuracy on RMT and 20 epochs for the accuracy to plateau on Mamba, all using Adam optimizer with learning rate $1e-5$. We also add a Larimar-1.3B baseline, which is finetuned on bAbi and Wikipedia samples with first and fifth terms from eqn. 1. The purpose of comparing MemReasoner with respect to Larimar is to disambiguate the benefits of temporal feature learning, iterative query, and read updates on top of the episodic memory. Larimar fine-tuning shares the same training setups as MemReasoner. Throughout the paper, we report task accuracy as the performance metric, so higher the better.

3.3 Results

3.3.1 Performance on bAbi Test Set

Table 1 reports the performances of MemReasoner, which is independently finetuned on original bAbi task 1 and task 2, along with the baselines on the corresponding bAbi test set of 1k samples. Results show that, while prompting techniques such as few-shot learning and chain-of-thought prompting (Yang et al., 2023) work well on task 1 which requires a single hop to find the entity location, those baselines perform much poorly on task 2 that requires learning temporal dependence and performing multiple hops across facts to generate the final answer of object location. MemReasoner, as well as RMT, Mamba and Larimar, all finetuned on bAbi achieves near-perfect accuracy on both tasks. Importantly, Larimar baseline falls behind MemReasoner on both tasks, while the gap being

Model type	Task 1	Task 2
CoT - GPT-3	97.3	72.2
Few-shot - GPT-3	98.4	60.8
RMT-.77B (bAbi)	**100**	99.4
Mamba-1.4B (bAbi)	**100**	95
Larimar-1.3B (bAbi)	60.6	44.9
MemReasoner-1.4B (bAbi)	**100**	**100**

Table 1: Performance on bAbi tasks. Best model is highlighted in bold. GPT-3 (=text-davinci-003) baselines are from Yang et al. (2023). Finetuning data, if any, seen by a model is specified within parentheses.

Model type	Avg. $\leq$ 8k	Avg. $\geq$ 16k	0k	1k	2k	4k	8k	16k	32k	64k	128k
RMT-.14B (BABILong)*	100	97	100	100	100	100	100	100	99	96	94
Mamba-.13B (BABILong)*	100	100	100	100	100	100	100	100	100	100	100
Few-shot - Meta-Llama-3-8B-Instruct*	84.4	-	98	**93**	**90**	**79**	62	-	-	-	-
Few-shot - Phi-3-mini-128k-instruct*	78.4	38	97	84	72	69	70	60	53	38	1
RAG - Llama3-ChatQA-1.5-8B*	59.6	60	60	62	60	58	58	60	60	56	64
RMT-.14B (bAbi)	20	-	**100**	0	0	0	0	0	0	0	-
RMT-.77B (bAbi)	20.2	-	**100**	0	1	0	0	0	0	-	-
Mamba-.13B (bAbi)	20.4	-	85	11	5	0	1	0	0	0	-
Mamba-1.4B (bAbi)	44.2	-	**100**	60	42	19	0	0	0	0	-
Larimar-1.3B (bAbi)	44.8	14.3	63	59	55	28	19	14	16	13	14
MemReasoner-1.4B (bAbi)	**84.6**	**68.5**	99	91	83	76	**74**	**71**	**68**	**70**	**65**

Table 2: BABILong Task 1 Results. Baseline results marked with "*" are from Kuratov et al. (2024). The finetuning data, if any, seen by each model is specified within parentheses.

Model type	Avg. $\leq$ 8k	Avg. $\geq$ 16k	0k	1k	2k	4k	8k	16k	32k	64k	128k
RMT-.14B (BABILong)*	98.8	68.5	100	100	99	98	97	94	82	59	39
Mamba-.13B (BABILong)*	98.0	94.5	98	98	98	98	98	98	98	95	87
Few-shot - Meta-Llama-3-8B-Instruct*	40.2	-	47	46	49	39	20	-	-	-	-
Few-shot - Phi-3-mini-128k-instruct*	40.6	15.5	57	38	38	36	**34**	**23**	**22**	15	2
RAG - Llama3-ChatQA-1.5-8B*	21.6	8.75	28	25	22	19	14	13	9	7	6
RMT-.14B (bAbi)	20	-	98	2	0	0	0	0	0	0	-
RMT-.77B (bAbi)	32.2	-	99	18	17	15	12	15	13	-	-
Mamba-.13B (bAbi)	16.2	-	64	10	3	3	1	0	0	0	-
Mamba-1.4B (bAbi)	31.6	-	94	44	15	5	0	0	0	0	-
Larimar-1.3B (bAbi)	31	**20.3**	42	41	29	22	21	19	16	**22**	**24**
MemReasoner-1.4B (bAbi)	**60.6**	18.5	**100**	**73**	**61**	**46**	23	20	19	17	20

Table 3: BABILong Task 2 Results. Baseline results marked with "*" are from Kuratov et al. (2024).

bigger on more complicated task 2, implying the read/write to episodic memory alone is not sufficient.

3.3.2 Performance on BABILong Test Set

Table 2 and Table 3 report accuracy of MemReasoner, together with baseline methods, on BABILong task 1 and task 2 samples, respectively. '-' means unavailable due to out of memory errors or maximal input length constraints. For task 1, the following observations can be made: (i) at half of model's context window, the accuracy of Llama-3-8B-Instruct drops to 80% and Phi-3-mini-128k drops to 63% of the corresponding model's performance at 0k samples, indicating LLMs are not good at utilizing their full context window. With RAG, the performance stays at a flat $\approx$ 60% all throughout. Interestingly, while RMT and Mamba, when finetuned on BABILong samples of up to 16k tokens, are the best models reported on BABILong leaderboard, they perform poorly on BABILong samples beyond 0k as we finetune them on bAbi samples. This suggests exposure to BABILong during training helps RMT and Mamba, as the models have seen facts embedded inside the background distractor text from PG19. Larimar finetuned on bAbi, while performing much poorly on bAbi test set and BABILong 0k set to begin with, the accuracy on longer BABILong samples is higher than bAbi-tuned RMT and Mamba baselines. In contrast, MemReasoner trained on bAbi generalizes well on BABILong for task 1, providing an average accuracy of 84.6% and 68.5% on $\leq$ 8k and $\geq$ 16k BABILong samples, respectively, though it was never exposed to BABILong samples during training.

For more complicated task 2, which requires learning temporal dependence between the facts and finding and using two supporting facts in correct order for generation, both few-shot prompting

Model type	Task 1	Task 2
RMT-.77B (bAbi)	45.8	23.7
Mamba-1.4B (bAbi)	67	44
Larimar-1.3B (bAbi)	24.9	7.8
MemReasoner-1.4B (bAbi)	**87.2**	**52.7**

Table 4: Robustness to location changes in bAbi test set.

Model type	0k	1k	2k	4k
RMT-.77B (bAbi)	**100**	13	11	13
Mamba-1.4B (bAbi)	81	8	0	0
Larimar-1.3B (bAbi)	45	19	20	11
MemReasoner-1.4B (bAbi)	83	**58**	**50**	**45**

Table 5: Performance on bAbi task 2 $\rightarrow$ BABILong task 1 generalization.

and RAG with different base LLM show poor performance to begin with, and sharply degrade with context length increase of test samples. Again, RMT and Mamba, when fitted to BABILong, perform well on test samples (though RMT shows performance degradation as samples get longer), both again fail to generalize from bAbi to BABILong. For example, the accuracy drops from near 100% at 0k to 18% for RMT and to 36% for Mamba at 1k, implying they haven't learned to solve the task. Poor results at short context length for Larimar also indicates model's failure to learn the task. MemReasoner, in comparison, provides an accuracy of 100% at 0k, 73% at 1k, and 46% at 4k, while performance degrades to $\approx$ 18.5% beyond 16k. The modest ($\approx$ 18.5%) performance of bAbi-tuned MemReasoner at 16k or longer context suggests that there remains significant room for MemReasoner to improve, which will be investigated in future.

3.3.3 Generalization to Out-of-distribution Test Sets

To test if the models have indeed learned to solve the tasks, we create a new testbed where the construct of the tasks remains the same, but the answer changes from training to test set. Specifically, we change the location information present in the answer set of bAbi training $\rightarrow$ test as follows: office $\rightarrow$ library, garden$\rightarrow$ garage, kitchen$\rightarrow$ cafe, bathroom $\rightarrow$ attic, bedroom$\rightarrow$ basement, hallway $\rightarrow$ gym. This now becomes a more stringent test, to which we subject all alternative architectures including MemReasoner. As shown in Table 4, RMT performs the worst in this setting across both tasks. On task 1, MemReasoner shows $\approx$ 20% higher accuracy than Mamba, whereas on task 2 MemReasoner wins by $\approx$ 8%.

Finally, we also check if the models trained on 2-hop bAbi task 2 can solve the simpler 1-hop task 1 but on the corresponding BABILong samples. Results are shown in Table 5, indicating that the best performing model on 0k BABILong task 1 samples is RMT, while MemReasoner being a second. However, both RMT and Mamba perform very poorly on longer (1-4k tokens) BABILong samples, whereas MemReasoner's accuracy remains strong.

3.4 Conclusion

In this work, we introduce a new memory-augmented LLM architecture that comes with two essential abilities required to perform robust multi-step reasoning, *i.e.*, learning temporal relations and to hop meaningfully between facts within a context. Our formulation and implementation of the multi-step reasoning mechanisms around the episodic memory is generic and in principle model-agnostic, and therefore can be leveraged to enhance other memory-augmented LLMs. We examine MemReasoner on BABILong, a benchmark purposed to test models' reasoning ability when relevant facts are distributed in background of very large textual corpora. This deceptively lengthy nature of BABILong samples, along with the presence of distracting text that is naturally occurring, makes the underlying reasoning task more challenging on which even bigger LLMs that have seen samples with long context during training fails. We show here that, MemReasoner trained on bAbi samples provides strong performance on BABILong, compared to the off-the-shelf powerful LLM baselines and alternative recurrent architectures that are also finetuned on bAbi data. We further show that MemReasoner generalizes better in the setting where answers in training set differs from those in the test within the same task. MemReasoner also shows good adaptation from two-hop to single-hop QA task, whereas the test samples are much longer and mixed with natural irrelevant text. These results suggest, designing alternative architectures with new loss objectives that encourage the model to learn the underlying reasoning skills is a potential path toward more robust reasoners.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091, 2022.

Payel Das, Subhajit Chaudhury, Elliot Nelson, Igor Melnyk, Sarath Swaminathan, Sihui Dai, Aurélie Lozano, Georgios Kollias, Vijil Chenthamarakshan, Jiří, Navrátil, Soham Dan, and Pin-Yu Chen. Larimar: Large language models with episodic memory control, 2024. URL `https://arxiv.org/abs/2403.11901`.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. Longrope: Extending llm context window beyond 2 million tokens, 2024. URL `https://arxiv.org/abs/2402.13753`.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2023.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models?, 2024. URL `https://arxiv.org/abs/2404.06654`.

Daniel Kahneman. *Thinking, fast and slow*. Farrar, Straus and Giroux, New York, 2011. ISBN 9780374275631 0374275637. URL `https://www.amazon.de/Thinking-Fast-Slow-Daniel-Kahneman/dp/0374275637/ref=wl_it_dp_o_pdT1_nS_nC?ie=UTF8&colid=151193SNGKJT9&coliid=I3OCESLZCVDFL7`.

Muhammad Khalifa, Lajanugen Logeswaran, Moontae Lee, Honglak Lee, and Lu Wang. Grace: Discriminator-guided chain-of-thought reasoning, 2023. URL `https://arxiv.org/abs/2305.14934`.

Georgios Kollias, Payel Das, and Subhajit Chaudhury. Generation constraint scaling can mitigate hallucination, 2024. URL `https://arxiv.org/abs/2407.16908`.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *arXiv preprint arXiv:2406.10149*, 2024.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. Same task, more tokens: the impact of input length on the reasoning performance of large language models, 2024. URL `https://arxiv.org/abs/2402.14848`.

Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yue Zhang. Evaluating the logical reasoning ability of chatgpt and gpt-4, 2023a. URL `https://arxiv.org/abs/2304.03439`.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023b.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work?, 2022. URL `https://arxiv.org/abs/2202.12837`.

Melanie Mitchell. How do we know how smart ai systems are?, 2023.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 2024.

Marianna Nezhurina, Lucia Cipolina-Kun, Mehdi Cherti, and Jenia Jitsev. Alice in wonderland: Simple tasks showing complete reasoning breakdown in state-of-the-art large language models, 2024. URL `https://arxiv.org/abs/2406.02061`.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization, 2024. URL `https://arxiv.org/abs/2404.19733`.

Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. Art: Automatic multi-step reasoning and tool-use for large language models, 2023. URL `https://arxiv.org/abs/2303.09014`.

Kha Pham, Hung Le, Man Ngo, Truyen Tran, Bao Ho, and Svetha Venkatesh. Generative pseudo-inverse memory. In *International Conference on Learning Representations*, 2022.

Ofir Press, Noah Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=R8sQPpGCv0`.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. Parallel context windows for large language models, 2023. URL `https://arxiv.org/abs/2212.10947`.

Sebastian Ruder. Challenges and opportunities in nlp benchmarking, 2021.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. URL `https://arxiv.org/abs/2302.04761`.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. Chain of thoughtlessness? an analysis of cot in planning, 2024. URL `https://arxiv.org/abs/2405.04776`.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL `https://arxiv.org/abs/2104.09864`.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks, 2015. URL `https://arxiv.org/abs/1503.08895`.

Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Large language models still can't plan (a benchmark for LLMs on planning and reasoning about change). In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022. URL `https://openreview.net/forum?id=wUU-7XTL5XO`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Chaojie Wang, Yanchen Deng, Zhiyi Lyu, Liang Zeng, Jujie He, Shuicheng Yan, and Bo An. Q*: Improving multi-step reasoning for llms with deliberative planning, 2024. URL `https://arxiv.org/abs/2406.14283`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart Van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. Effective long-context scaling of foundation models, 2023. URL `https://arxiv.org/abs/2309.16039`.

Zhun Yang, Adam Ishay, and Joohyung Lee. Coupling large language models with logic programming for robust and general reasoning from text. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

Tao Yuan, Xuefei Ning, Dong Zhou, Zhijie Yang, Shiyao Li, Minghui Zhuang, Zheyue Tan, Zhuyu Yao, Dahua Lin, Boxun Li, Guohao Dai, Shengen Yan, and Yu Wang. Lv-eval: A balanced long-context benchmark with 5 length levels up to 256k, 2024. URL `https://arxiv.org/abs/2402.05136`.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning, 2022. URL `https://arxiv.org/abs/2203.14465`.

A Appendix

A.1 Related Work

LLM Reasoning Logical reasoning, a critical aspect for advancing many scientific fields, involves deducing new conclusions from existing facts and rules. To derive the final answer, such reasoning challenges often require multiple steps to be executed effectively and in the right order. For instance, with facts like "John picked up the football" and "John went to the bedroom", a logical process will be to deduce that the football's current location is bedroom. Despite showing advanced ability to learn from instructions and in-context demonstrations to answer questions (Brown et al., 2020; Min et al., 2022), LLMs struggle with complex logical reasoning, especially multi-step reasoning (Liu et al., 2023a). This failure has been attributed to the autoregressive nature of LLMs (Stechly et al., 2024), which can be characterized by "System 1" (Kahneman, 2011), a mode of thought that is fast, instinctive but less accurate. To address this limitation, recent work proposes prompting LLMs to mimic generating intermediate chain of thought (CoT) reasoning steps (Wei et al., 2022), providing access to external tools/verifiers (Schick et al., 2023), or a combination of both (Paranjape et al., 2023), to mimic the process of generating deliberative and logical thinking steps, i.e., the "System 2" mode. Another direction currently being explored is to train reward models to rank the candidate solutions or rank the intermediate steps (Khalifa et al., 2023; Wang et al., 2024). Different from these works, MemReasoner does not rely on deliberate prompt engineering or access to external tools, neither does it require feedback from an external reward model. Instead, inspired by the distinction between System 1 and System 2-like thinking, MemReasoner utilizes the decoder for fast generation and the memory module for slow reasoning, which are two components tightly integrated via training. In that sense, MemReasoner is closer to the line of works that use (generated) rationales for supervised finetuning or for preference tuning of LLMs to enhance their reasoning abilities (Zelikman et al., 2022; Pang et al., 2024). However, it remains unexplored how those approaches perform on iterative reasoning tasks over lengthy context that is unseen during training.

Long-context Modeling The scope of the present study encompasses two distinct challenges around multi-step reasoning tasks, namely (1) processing very long context and (2) "hopping" over that context in a temporally-aware manner to link disjoint pieces of information and generate answers based on that. On the first challenge, vanilla transformer-based models struggle due to quadratic time and space complexity of self-attention and the increasing memory requirement of the key-value cache during generation. Recently, there has been significant progress in long-context modeling with transformers by using a mix of local and global attention (Munkhdalai et al., 2024), by continued pretraining on longer sequences (Xiong et al., 2023; Ding et al., 2024), by context window sliding and segmentation (Ratner et al., 2023), and by applying position extrapolation or interpolation to extend input length beyond the training phase (Press et al., 2022; Su et al., 2023). Promising alternative directions include the development of novel recurrent architectures Bulatov et al. (2022) and state-space-models (Gu & Dao, 2023). Nevertheless, many of these techniques require training on longer sequences. Additionally, a number of studies and benchmarks suggest that the long-context LLMs may not be able to fully utilize their context window, and therefore performance degrades on simple retrieval and complicated reasoning tasks as the input length grows and/or the position of the answer varies within the context (Hsieh et al., 2024; Yuan et al., 2024; Liu et al., 2023b; Levy et al., 2024).

Status Check on LLM Reasoning Consequently, in parallel to impressive advances in LLMs abilities, caution has been raised on the discrepancy between claimed reasoning abilities as per standardized benchmarks and true reasoning skills. The scientific community has advocated for careful investigations of issues such as data contamination, performance robustness and generalization, and flawed reasoning benchmark that supports "shortcut learning" (Mitchell, 2023). Recently, a number of tasks and benchmarks have been developed to address these issues (Valmeekam et al., 2022; Kuratov et al., 2024; Nezhurina et al., 2024). Along this line, we here show the generalization robustness of MemReasoner across (i) "unseen" context that consists of varying length of irrelevant natural text and (ii) answer distribution that is different from the training distribution.

A.2 Algorithm

Algorithm 1:

```
] Function IterativeRead( $q, \{c_1, ..., c_E\}, \alpha, \tau$ ):
    // $q$ denotes the query tokens while $\{c_1, ..., c_E\}$ denote the $E$ lines of
       context tokens, $\alpha$ is a hyperparameter for the query update, $\tau$ is a
       threshold hyperparameter for terminating iterations, $P_a$ is the
       prompt given to the decoder for answer generation
    // encode query and context lines with encoder
    $z_q \leftarrow$ encode($q$) for $i \leftarrow 1$ to $E$ do
        $z_i \leftarrow$ encode($c_i$)
    end
    // apply temporal encoding over the sequence of context lines and
       write to memory
    $\tilde{z}_1, ..., \tilde{z}_E \leftarrow$ temporalEncoding($z_1, ..., z_E$)
    $\hat{M} \leftarrow$write($\tilde{z}_1, ..., \tilde{z}_E$)
    // iterative read and query update
    $\tilde{z}_r \leftarrow$ queryUpdate($z_q, \alpha, \tau$)
    // Map to latent prior to performing temporal encoding
    $i^* \leftarrow \arg\min_{i \in \{1...E\}} ||\hat{z}_i - \hat{z}_r||_2$
    return decode($z_{i^*}$, $W_M$, $P_a$)  // generate the answer with the decoder, $W_M$
     is a learnable parameter which interfaces the $z_{i^*}$ with the decoder


Function temporalEncoding($\{z_1, ..., z_E\}$, method):
    // temporally encode the sequence $\{z_1, ..., z_E\}$
    if method = $PE$ then
        return $\{z_i + PE(i) | \forall i \in \{1, ..., E\}\}$
    else if method = GRU then
        return GRU($\{z_1, ..., z_E\}$)

Function queryUpdate($z_q, \alpha, \tau$):
    // given the query encoding $z_q$ and threshold $\tau$, perform iterative
       reading and update query
    $\tilde{z}_r \leftarrow$ read ($W_q z_q, M$)                    // $W_q$ is learned parameter
    $z_q = z_q + \alpha \tilde{z}_r$                                        // query update
    $\tilde{z}_{r,\text{next}} \leftarrow$ read ($W_q z_q, M$)
    do
        $\tilde{z}_r \leftarrow \tilde{z}_{r,\text{next}}$
        $z_q = z_q + \gamma \tilde{z}_r$
        $\tilde{z}_{r,\text{next}} \leftarrow$ read ($W_q z_q, M$)
    while $||\tilde{z}_{r,next} - \tilde{z}_r||_2 > \tau$
    return $\tilde{z}_r$
```

A.3 Additional dataset preprocessing details

In the unprocessed bAbi data, a single data instance consists of a sequence of lines representing facts to reason over with questions interspersed throughout the facts. We preprocess the bAbi data such that after preprocessing, a single training sample consists of a single question with facts for reasoning being the lines before it, with previous questions replaced by an empty line. On average this leads to about 2 empty lines per training sample. For batches containing training samples with different lengths of context episodes, we pad shorter episodes with rows of the encoder padding token at the beginning.

A.4 Comparison of inference-time complexity

Let H_1, H_2 and d_1, d_2 be the number of transformer layers and hidden state dimension in the encoder and decoder, respectively. Let E denote the number of context lines in a sample, L be the max context length, L_1 be the max query length, D be the latent space dimension, and m be the memory size. The inference-time computational complexity for MemReasonr can be estimated by the encoder complexity $\mathcal{O}(H_1((EL^2+L_1^2)d_1+(EL+L_1)d_1^2))$, temporal encoding complexity $\mathcal{O}(Ed^2)$, memory operation complexity $\mathcal{O}(Edm^2)$, decoding complexity $\mathcal{O}(H_2(|P_a|^2d_2 + |P_a|d_2^2))$, and broadcasting complexity $\mathcal{O}(d_1dE)$ and $\mathcal{O}(d_2dH_2)$. For a typical GPT decoding, the inference-time computational complexity is $\mathcal{O}(H_2((EL+L_1)^2d_2 + (EL+L_1)d_2^2))$.

A.5 Ablation Studies

A.5.1 Memory

In Table 6, we conduct the ablation study on the episodic memory module in MemReasoner on bAbi and BABILong, task 1 and 2. Specifically, MemReasoner w/o memory module uses the same architecture of encoder and decoder (BERT-Large and GPT2-Large respectively) but does not use the memory module for encoding the context. Instead, the MemReasoner w/o memory uses the encoder to encode only the question and this is passed in to the decoder as kv-cache. Additionally, the context and question are passed to the decoder as part of the prompt with the format:

```
Context:
{context}
Question:
{question}
Answer:
```

where {context} and {question} represent the context and the question for the datapoint. We train the model with reconstruction loss to ensure that the model is able to fill in the answer given this prompt and with autoencoding loss on the pretraining dataset (see last term of Equation 1) in order to reduce overfitting on bAbi data. We train MemReasoner w/o memory module for 5 epochs.

MemReasoner w/o memory module trained on bAbi task 1 obtains almost perfect accuracy on bAbi task 1 and BABILong task 1. However, its generalization ability to long context (BABILong 1k and 2k) is much inferior to MemReasoner (MemReasoner\memory 0% vs. MemReasoner 91% on BABILong 1k). Similar trends can also be seen from bAbi task 2 trained MemReasoner\memory, implying the significance of the episodic memory module and the operations around it in MemReasoner.

Model type	Task 1	0k	1k	2k	Task 2	0k	1k	2k
MemReasoner\memory	**100**	**100**	0	-	99.3	**100**	29	-
MemReasoner	**100**	99	**91**	**83**	**100**	**100**	**73**	**61**

Table 6: Ablation study on the episodic memory

A.5.2 Temporal Encoding

In Table 7, we experiment with different temporal encoding schemes, including non-parametric method (Positional Encoding) and parametric method (GRU). In the table, we show MemReasoner's accuracy on BABILong Task 1. It can be seen that GRU encoding has significant advantage over Positional Encoding, with much slower decay in the accuracy as the context length increases. Additionally, though showing higher accuracy compared with Positional Encoding, uni-directional GRU's accuracy decreases faster than bi-directional GRUs. Since 1-layer bi-directional GRU has similar performance with 2-layer bi-directional GRU, we choose the lighter model and use 1-layer bi-directional GRU throughout the experiments in this paper.

Encoding scheme	0k	1k	2k
Positional Encoding	**100**	27	20
2-layer bi-directional GRU	**100**	90	80
2-layer uni-directional GRU	94	75	61
1-layer bi-directional GRU	99	**91**	**83**

Table 7: Ablation study on the temporal encoding schemes.

A.5.3 Query Update α

In Table 8, we exploit test-time inference hyper-parameter α and its effect in reasoning tasks' performance. We draw inspiration from Kollias et al. (2024), where authors investigated the effect of scaling readout vectors to improve generation quality. In Line 20 of Algorithm 1, when using an $\alpha > 1$, we equivalently scale up the readout vectors which greatly help our generalization to Task 1 BABILong according to Table 8 (e.g. from 14% to 45% on 4k context token task).

Query update	Task 2 bAbi	Task 2 BABILong									Task 1 BABILong			
α	location change	0k	1k	2k	4k	8k	16k	32k	64k	128k	0k	1k	2k	4k
1	52.6	**100**	46	25	18	18	13	16	12	13	78	21	17	14
4	**54.2**	**100**	**73**	**61**	**46**	**26**	**22**	**19**	**19**	**27**	**83**	47	44	40
8	52.7	**100**	**73**	**61**	**46**	23	20	**19**	17	20	**83**	**58**	**50**	**45**

Table 8: Ablation study on the query update parameter α.

A.5.4 Training epochs

In Table 9, we evaluate MemReasoner's performance when fine-tuned on bAbi task 2 as a function of the number of training epochs. Specifically, with fewer epochs, MemReasoner demonstrates stronger robustness to location change, reaching an accuracy of 79% at the 66th epoch, which decreases to around 50% as the training continues (at 100/200th epoch). On the other side, MemReasoner's accuracy on shorter context tasks in BABILong Task 1 and 2 (i.e. 0-4k) improves as the training continues.

	Task 2 bAbi	Task 2 BABILong									Task 1 BABILong			
#epochs	location change	0k	1k	2k	4k	8k	16k	32k	64k	128k	0k	1k	2k	4k
66	**78.0**	99	70	54	30	27	23	17	**18**	17	58	51	45	37
100	47.3	**100**	70	57	38	**28**	**31**	**25**	12	19	82	**58**	**50**	**46**
200	52.7	**100**	**73**	**61**	**46**	23	20	19	17	**20**	**83**	**58**	**50**	45

Table 9: Ablation study on the number of training epochs