CONTINUAL MEMORY: CAN WE REASON AFTER LONG-TERM MEMORIZATION?

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing reasoning tasks often follow the setting of "reasoning while experiencing", which has an important assumption that the raw contents can be always accessed while reasoning. However, human beings frequently adopt another reasoning setting in daily life, referred to "reasoning after memorizing". Concretely, human beings have the ability to unconsciously memorize his/her experiences within limited memory capacity, from which they can recall and respond to subsequent tasks. In this setting, the raw contents are no longer available during reasoning, thus we need to compress and memorize the input stream in one pass, trying to answer general queries that are unseen before. Memory augmented neural networks introduce a write-read memory to perform such human-like memorization and reasoning, but they continually update the memory from current information and inevitably forget the early contents, failing to answer the queries relevant to early information. In this paper, we propose the Continual Memory (CM) to explore this ability of reasoning after long-term memorization. To alleviate the gradual forgetting of early information, we develop self-supervised memorization training with item-level and sequence-level objectives. We demonstrate several interesting characteristics of our continual memory via synthetic data, and evaluate its performance by several downstream tasks, including long-term text QA, long-term video QA and recommendation with long sequences.

1 INTRODUCTION

Studies that dominate reasoning (QA/VQA/Recommendation) benchmarks usually follow the setting of "reasoning while experiencing", where the original input can be always accessed by the time a task or query is known. In such a setting, unlimited inference resources are permitted, providing a perfect chance to replay on any part of the original input, where complex interaction between raw input and queries can be designed to extract relevant information with little loss. The other setting, we refer to "reasoning after memorizing", restricts the inference phase to only access compressed states with limited capacity, where the compressing procedure over the raw input is not aware of the exact task or query. This poses high risks during memorization at losing information to keep integrity to solve unknown problems in the future, as it needs to decide what to memorize to achieve best expected performance among dozens of possible queries.

The setting of "reasoning after memorizing", though does not aim at breaking the regular reasoning benchmark record, is meaningful from the cognition perspective. Similar to this setting, we humans may not even know the tasks in the future when we are experiencing current events, and we also cannot go back to replay when we are solving problems at hand. However, it's our instincts, which continually process information during our entire life with limited and compressed memory storages, that allow us to recall and draw upon past events to frame our behaviors given the present situations.

Memory augmented neural networks (MANNs) (Graves et al., 2014; Weston et al., 2014; Graves et al., 2016) introduce a write-read memory to perform such human-like memorization and reasoning processes, which compress the input data into a fixed-size memory and only read relevant information from the memory during reasoning. However, these approaches train the memorization mechanism only by back-propagated losses in responding to the final goal, which inevitably leads to the gradual forgetting of early contents (Kirkpatrick et al., 2017). That is, when dealing with the long-term input stream, the memory may only focus on short-term contents and naturally neglect

long-term clues. Thus, existing MANNs fail to answer the query relevant to early information due to the lack of long-term memorization training.

To overcome gradual forgetting of early information and increase the generalization ability of the memorization technique, in this paper, we develop extra self-supervised tasks, inspired by the fact that human beings can recall details nearby some specific events and distinguish whether a series of events happened in the history. Concretely, we design the self-supervised memorization training with item-level and sequence-level objectives, which lessen the impact of the gradual forgetting by recalling the recorded history contents from continual memory. The item-level objective aims to predict the masked items in history fragments, which are sampled from the original input stream and parts of items are masked as the prediction target. And the sequence-level objective tries to distinguish whether a historical fragment ever appears in the input stream, where we directly sample positive fragments from the early input stream and replace parts of the items in positive ones as negative fragments. We also give implementations on segment-level maintenance of memory to better capture context clues and improve the modeling efficiency. We illustrate the long-term memorization ability of our continual memory via a synthetic task, and evaluate its performance at solving real-world downstream tasks, including long-term text QA, long-term video QA and recommendation with long sequences, showing that it achieves significant advantages over existing MANNs in the "reasoning after memorizing" setting.

2 Related Works

Memory Augmented Neural Networks (MANNs) introduce external memory to store and access the past information by differentiable write-read operators. Neural Turing Machine (NTM) (Graves et al., 2014) and Differentiable Neural Computer (DNC) (Graves et al., 2016) are the typical MANNs for human-like reasoning under the setting of "reasoning after memorizing", whose inference relies only on the memory with limited capacity rather than starting from the original input data. However, these MANNs are not designed to memorize the past and prepare to answer unseen questions in the future. Instead, they share the same training objective as those methods developed for the setting of "reasoning while experiencing", inevitably incurring catastrophic forgetting (Kirkpatrick et al., 2017; Le et al., 2019a). Different from this type of MANNs, the End-to-End Memory Network (E2E-MN) (Sukhbaatar et al., 2015) assigns each memory slot to an entity, and aggregates information by multiple attention-based layers to answer the given queries. Considering all entities are stored in the memory separately, E2E-MN conducts inference directly on the original input thus belongs to the setting of "reasoing while experiencing".

The development of these MANNs has motivated a wide range of applications in various fields, especially for question answering (Graves et al., 2014; 2016; Weston et al., 2014) and recommendation (Pi et al., 2019; Ebesu et al., 2018). They exploit MANNs mainly to help capture long-range dependencies in dealing with long input sequences, but not paying efforts in dealing with the catastrophic forgetting issue in MANNs. Recently, there are a few works trying to alleviate this problem in training MANNs under the setting of "reasoning after memorizing". (Le et al., 2019a) proposes to measure "remember" ability by the final gradient on the early input, and adopts a uniform writing operation on the memory. (Rae et al., 2019) minimizes the difference between successive memory states along with the reasoning objective, as they assume the steadily changed memory states will benefit remembering older information. Note that our approach is different and parallel to these techniques, since we give no assumptions on what behavior will remember the most. Instead, we optimize towards this goal directly by designing auxiliary tasks in a self-supervised manner.

3 Continual Memory

3.1 Problem Formulation

As shown in Figure 1, given a input stream $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots\}$, we apply a continual memory machine $\mathcal{G}_\Theta(\cdot)$ to compress them into continual memory $\mathbf{M} = \{\mathbf{m}_k\}_{k=1}^{K}$ with K memory slots. By self-supervised memorization training, we try to overcome the gradual forgetting of early information and make it possible to capture the clues at any time in the stream. Concretely, based on continual memory, we develop the history recall model $\mathcal{H}_\xi(\cdot)$ to reconstruct the masked history

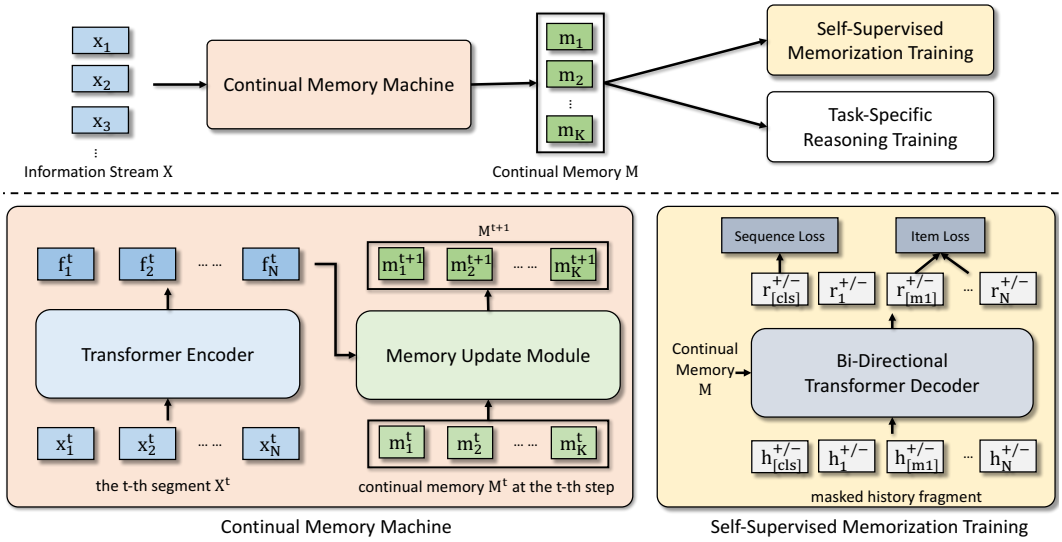

Figure 1: The Framework of Continual Memory and Self-Supervised Memorization Training.

fragments and distinguish positive history fragments from negative ones. Simultaneously, we train the task-specific reason model $\mathcal{R}_\Omega(\cdot)$ based on continual memory. Under the setting of "reasoning after memorizing", we can develop the continual memory $\mathbf{M} = \mathcal{G}_\Theta(\mathbf{X})$ and then infer the answer for any relevant query $\mathbf{Q}$ by $\mathbf{A} = \mathcal{R}_\Omega(\mathbf{M}, \mathbf{Q})$, where $\mathbf{M}$ can be reused for a series of queries.

3.2 Continual Memory Machine

We deal with the input stream $\mathbf{X}$ from the segment level rather than item level, i.e., we cut the input sequence into fixed-length segments and memorize them into the memory slots segment-by-segment. Compared to NTM (Graves et al., 2014), which stores the input stream item-by-item orderly with a RNN-based controller, our segment-level memorization can further capture the bi-directional context of each item and improve the modeling efficiency. We denote the t-th segment as $\mathbf{X}^t = \{\mathbf{x}_n^t\}_{n=1}^N$ and the current memory as $\mathbf{M}^t = \{\mathbf{m}_k^t\}_{k=1}^K$, where we have recorded t-1 segments in $\mathbf{M}^t$. The $\mathbf{x}_n^t$ and $\mathbf{m}_n^t$ have the dimension d_x and d_m, respectively.

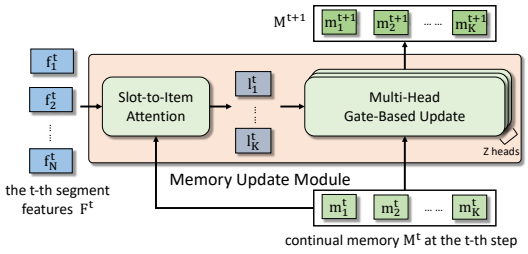

Figure 2: The Process of Memory Updating.

We first model the t-th segment by a Transformer encoder (Vaswani et al., 2017) and obtain the sequence features $\mathbf{F}^t = \{\mathbf{f}_n^t\}_{n=1}^N$ with dimension d_x. After it, we apply a memory update module to write $\mathbf{F}^t$ into $\mathbf{M}^t$. As shown in Figure 2, we apply a slot-to-item attention to align the sequence features to slot features in the current memory $\mathbf{M}^t$, and then develop multi-head gate-based update. Concretely, we first calculate the slot-to-item attention matrix where each element means the relevance of a slot-item pair, and then learn aligned features $\mathbf{L}^t = \{\mathbf{l}_k^t\}_{k=1}^K$ for each slot, given by

$$\alpha_{kn}^t = \mathbf{w}_a^\top \tanh(\mathbf{W}_1^a \mathbf{m}_k^t + \mathbf{W}_2^a \mathbf{f}_n^t + \mathbf{b}^a),\ \hat{\alpha}_{kn}^t = \frac{\exp(\alpha_{kn}^t)}{\sum_{j=1}^K \exp(\alpha_{jn}^t)},\ \mathbf{l}_k^t = \sum_{n=1}^N \hat{\alpha}_{kn}^t \mathbf{f}_n^t, \tag{1}$$

where $\mathbf{W}_1^a \in \mathbb{R}^{d_{model} \times d_m}$, $\mathbf{W}_2^a \in \mathbb{R}^{d_{model} \times d_x}$ and $\mathbf{b}^a \in \mathbb{R}^{d_{model}}$ are the projection matrices and bias. $\mathbf{w}_a^\top$ is the row vector. After it, we project slot features and aligned features into Z subspaces, which is similar to the Multi-Head setting in Transformer (Vaswani et al., 2017), given by

$$\mathbf{m}_{kz}^t = \mathbf{W}_z^M \mathbf{m}_k^t,\ \mathbf{l}_{kz}^t = \mathbf{W}_z^L \mathbf{l}_k^t, \tag{2}$$

where $\mathbf{W}_z^M \in \mathbb{R}^{d_{model}/Z \times d_m}$ and $\mathbf{W}_z^L \in \mathbb{R}^{d_{model}/Z \times d_x}$ are the projection matrices. The $\mathbf{m}_{kz}^t$ and $\mathbf{l}_{kz}^t$ are the slot and aligned sub-features in the z-th subspace. Next, the k-th slot sub-feature $\mathbf{m}_{kz}^t$ is updated with the corresponding sub-feature $\mathbf{l}_{kz}^t$ based on the z-th GRU unit with $\frac{d_{model}}{Z}$-d hidden states, given by

$$\mathbf{m}_{kz}^{t+1} = \mathrm{GRU}_z(\mathbf{m}_{kz}^t, \mathbf{l}_{kz}^t), \tag{3}$$

where $\mathbf{l}_{kz}^t$ is the current input of the z-th GRU unit. Next, the new slot feature $\mathbf{m}_k^{t+1}$ is aggregated from Z subspaces by $\mathbf{m}_k^{t+1} = \mathbf{W}^o \mathrm{Concat}(\mathbf{m}_{k1}^{t+1}, \cdots, \mathbf{m}_{kZ}^{t+1})$, where $\mathbf{W}^o \in \mathbb{R}^{d_m \times d_{model}}$ is the aggregation matrix. After the memorization of T segments, we can obtain continual memory $\mathbf{M}^T$ and we denote it by $\mathbf{M}$ for convenience. Note that, at the inference stage, we can develop and update the continual memory in real time by $\mathcal{G}_\Theta(\cdot)$, thus we do not need the raw contents during subsequent reasoning and have the ability to reason after long-term memorization.

3.3 Self-Supervised Memorization Training

After memorizing the input stream, we conduct self-supervised memorization training to alleviate the gradual forgetting of early information by the history recall model $\mathcal{H}_\xi(\cdot)$ with item-level and sequence-level objectives, where the item-level objective aims to reconstruct the masked positive history fragments and the sequence-level objective tries to distinguish positive history fragments from negative ones.

Concretely, we sample the preceding segment from the input stream as the positive history fragment $\{h_1^+, h_2^+, \cdots, h_N^+\}$ with N items, where each item h_*^+ corresponds to a feature $\mathbf{x}_*$ in the stream. We then mask 50% of items in the fragment and add an especial item [cls] at the beginning to obtain the masked positive history fragment $H^+ = \{h_{[cls]}^+, h_1^+, h_{[mask_1]}^+, \cdots, h_N^+\}$. In order to guarantee that the model $\mathcal{H}_\xi(\cdot)$ reconstructs the masked fragment by utilizing continual memory rather than only relying on fragment context, we set the mask ratio to 50% instead of 15% in BERT (Devlin et al., 2019). Moreover, we construct the masked negative history fragment $H^- = \{h_{[cls]}^-, h_1^-, h_{[mask_1]}^-, \cdots, h_N^-\}$ by replacing 50% of unmasked items in the positive fragment. Here we construct B masked positive fragments with corresponding B negative ones. Next, we adopt a bidirectional Transformer decoder (Vaswani et al., 2017) without the future masking to model each history fragment H^+/H^-. In the decoder, each history item can interact with all other items. As shown in Figure 1, the continual memory $\mathbf{M}$ is input to the second Multi-Head Attention sub-layer in each decoder layer. Finally, we obtain the features $\{\mathbf{r}_{[cls]}^{+/-}, \mathbf{r}_1^{+/-}, \mathbf{r}_{[mask_1]}^{+/-}, \cdots, \mathbf{r}_N^{+/-}\}$ where each $\mathbf{r}_*^{+/-}$ has the dimension d_x.

Item-Level Reconstruction. We first predict the masked items of positive history fragments to build the item-level loss. Considering there are too many item types, we apply the contrastive training He et al. (2020); Chen et al. (2020) based on the ground truth and other sampled items. For the $N/2$ masked items, we compute the item-level loss by

$$\mathcal{L}_{item} = -\frac{2}{N} \sum_{i=1}^{N/2} \log \frac{\exp(\phi(\mathbf{r}_{[mask_i]}^+, \mathbf{y}_i))}{\exp(\phi(\mathbf{r}_{[mask_i]}^+, \mathbf{y}_i)) + \sum_{j=1}^{J} \exp(\phi(\mathbf{r}_{[mask_i]}^+, \mathbf{y}_j))}, \tag{4}$$

where $\mathbf{y}_i \in \mathbb{R}^{d_x}$ is the feature of ground truth of the i-th masked item, $\mathbf{y}_j \in \mathbb{R}^{d_x}$ is the feature of sampled items and $\phi(\cdot)$ is the inner product.

Sequence-Level Prediction. Next, we predict whether the masked history fragment ever appears in the current input stream, i.e. distinguish positive history segment from negative ones. Concretely, we project the feature $\mathbf{r}_{[cls]}^{+/-}$ into a confident score $s^{+/-}$ and develop the sequence-level loss by

$$\mathcal{L}_{seq} = -\sum_{i=1}^{B} \log(s_i^+) + \sum_{j=1}^{B} \log(1 - s_j^-), \tag{5}$$

where B is the number of positive and negative history fragments.

3.4 TASK-SPECIFIC REASONING TRAINING

Besides self-supervised memorization training, we simultaneously develop task-specific reasoning training. For several downstream tasks, we propose different task-specific reason model $\mathcal{R}_\Omega(\mathbf{M}, \mathbf{Q})$ for any query $\mathbf{Q}$ based on continual memory $\mathbf{M}$. Here we adopt the simple and mature components in the reason model for fair comparison. The details are introduced in Appendix A.1. Briefly, we first learn the query representation $\mathbf{q}$ by a task-specific encoder and then perform the multi-hop attention-based reasoning. Finally, we obtain the reason loss $\mathcal{L}_r$ from $\mathcal{R}_\Omega(\mathbf{M}, \mathbf{Q})$.

Eventually, we combine the memorization and reason losses to train our model, given by

$$\mathcal{L}_{cm} = \lambda_1 \mathcal{L}_{item} + \lambda_2 \mathcal{L}_{seq} + \lambda_3 \mathcal{L}_r, \tag{6}$$

where λ_1, λ_2 and λ_3 are applied to adjust the balance of three losses.

4 EXPERIMENTS

We validate our continual memory on synthetic data and several downstream tasks, including long-term text question answering, long-term video question answering and recommendation with long sequences.

4.1 EXPERIMENT SETTING

Model Setting. We first introduce the common model settings for all downstream tasks. We set the layer number of the Transformer encoder and bi-directional Transformer decoder to 3. The head number in Multi-Head Attention is set to 4. And the subspace number Z in the memory update module is also set to 4. We set λ_1, λ_2 and λ_3 to 1.0, 0.5 and 1.0, respectively. The number B of history fragments is set to 5. During training, we apply an Adam optimizer (Duchi et al., 2011) to minimize the multi-task loss $\mathcal{L}_{cm}$, where the initial learning rate is set to 0.001.

Baseline. We compare our continual memory with the end-to-end reasoning methods corresponding to the setting of "reasoning while experiencing" , and the memory-based reasoning approaches under the "reasoning after memorizing" setting. The end-to-end baselines are different in downstream tasks and the memory-based baselines mainly are DNC (Graves et al., 2016) and NUTM (Le et al., 2019b). For fair comparison, we modify the reason module of DNC and NUTM to be consistent with our continual memory, i.e. we only retain the write head of DNC and NUTM during memory building. And the number of memory slots in DNC and NUTM is also set to K. Besides, we set the core number of NUTM to 4.

4.2 SYNTHETIC EXPERIMENT

Dataset. First, we introduce the setting of the synthetic task and define the general concepts as below. The more details of data synthesis are introduced in Appendix A.2.

- *Fact* and *Stream*: A fact represents an item in the input stream, denoted by a R_f-d one-hot vector where R_f is the number of fact types. The input stream contains a series of facts and we denote the stream length by R_l.
- *Query*: A query is represented by a R_q-d one-hot vector, i.e., there are R_q different queries.
- *Logic Chain*, *Evidence* and *Answer*: We synthesize a series of logic chains and the i-th chain is denoted by $(evidence^i, query^i, answer^i)$. The $evidence^i$ is a sequence of facts $\{fact_1^i, \cdots, fact_{R_c}^i\}$, which orderly appear in the input stream. And R_c is the length of the evidence. Given $query^i$, we expect the model to reason the correct $answer^i$ based on $evidence^i$, where we have R_a different answers. There are totally $R_q * R_a$ chains corresponding to each query-answer pair.
- *Consecutive and Discrete Evidence*: We synthesize two datasets with the consecutive and discrete evidence, respectively. The facts in the consecutive evidence continuously appear in the stream, but facts in the discrete evidence are distributed in different parts of the stream.

Table 1: Performance Comparisons on Synthetic Data. R_f=400, R_l=200, R_q=40, R_a=30, R_c=5.

Method	Setting	Discrete Evidence			Consecutive Evidence		
		Early	Later	All	Early	Later	All
Two-Stream	End-to-End	9.45	9.32	9.39	13.57	13.41	13.49
Multi-Hop Reason	End-to-End	32.16	33.42	32.29	34.38	34.50	34.44
DNC	Memory-Based	13.37	22.54	17.95	20.56	26.59	23.58
NUTM	Memory-Based	17.84	23.59	20.72	24.31	29.71	27.01
CM (Only Reason)	Memory-Based	18.71	**25.13**	21.92	25.79	31.38	28.63
CM (Full)	Self-Sup. Memory	**22.14**	24.98	**23.56**	**28.42**	**31.71**	**30.07**
CM (Only $\mathcal{L}_{item}$)	Self-Sup. Memory	21.79	24.46	23.13	27.88	31.12	29.50
CM (Only $\mathcal{L}_{seq}$)	Self-Sup. Memory	19.75	23.81	21.78	26.63	30.25	28.44
CM (Single Head)	Self-Sup. Memory	21.24	24.12	22.68	27.79	31.14	29.47

Baselines and Model Details. The **Two-Stream** method directly models the input stream by RNN and fuse the stream feature with the query feature to reason the answer. The **Multi-Hop Reason** method further applies multiple attention layers after RNN-based stream modeling to capture the query-relevant clues. In the main experiment, we set the dataset hyper-parameters R_f, R_l, R_q, R_a and R_c to 400, 200, 40, 30, and 5, respectively. The facts of the evidence may appear in different stages of the input stream. **Early** means the facts appear in the preceding 50% of the stream and **Later** means the facts appear in the subsequent 50%. For our continual memory, we set the d_x, d_m and d_{model} to 128. The number K of memory slots and length N of segments are set to 20 and 10, respectively. And we sample all other facts as negative items in $\mathcal{L}_{item}$.

Evaluation Results. Table 1 reports the performance comparison between our method and baselines, where CM(Full) is the full model with the memorization and reasoning training and CM(Only Reason) only employ the task-specific reasoning training. Overall, end-to-end methods have the close early and later performance, but memory-based approaches DNC, NUTM and CM(Only Reason) achieve the terrible early performance due to the gradual forgetting. By the self-supervised memorization training, our CM(Full) significantly improves the early accuracy and achieves the best memory-based reasoning performance. This fact suggests our proposed memorization training can alleviate the gradual forgetting of early information. Besides, CM(Only Reason) outperforms other memory-based methods DNC and NUTM, which indicates our continual memory machine can better memorize the long-term information even without the memorization training. Moreover, we can find the Two-Stream approach achieves the worst performance but the Multi-Hop Reason method has a high accuracy, which demonstrates the performance of end-to-end methods mainly depends on the complicated interaction between the raw contents and queries.

Ablation Study. We next perform ablation studies on the memorization loss and multi-head update strategy. Concretely, we first remove the sequence-level or item-level loss to produce two ablation models CM(Only $\mathcal{L}_{item}$) and CM(Only $\mathcal{L}_{seq}$). As shown in Table 1, CM(Full) outperforms two ablation models on all metrics, which illustrates two self-supervised losses are both helpful for long-term memorization. Moreover, CM(Only $\mathcal{L}_{item}$) achieves better accuracy than CM(Only $\mathcal{L}_{seq}$), demonstrating the importance of the item-level loss. Next, we discard the multi-head setting from the memory update module to generate the ablation model CM(Single Head). From the results, we can find the CM(Single Head) has the performance degradation, indicating the multi-head update can improve the memorization ability of continual memory.

Hyper-Parameters Analysis. We then explore the effect of the slot number K. We set K to [5, 10, 15, 20, 25] and display the results in Figure 3. We note that the model performance gradually improves with the increase of slot number and it reaches the bottleneck when the slot number is set to 25. Besides, the early performance has more significant improvement than the later performance.

Longer Stream and Evidence. To further validate the characteristics of our continual memory, we synthesize a more complicated dataset where R_f, R_l, R_q, R_a and R_c are set to 2000, 1000, 40, 30, and 10, respectively. That is, the dataset contains the longer input stream and more complex evidence. We set the number K of memory slots and length N of segments to 20. As shown in Figure 4, we display the reasoning performance of each model when the facts in the evidence appear

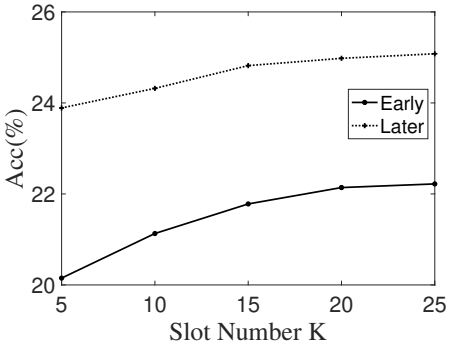

Figure 3: Effect of the Memory Slot Number.

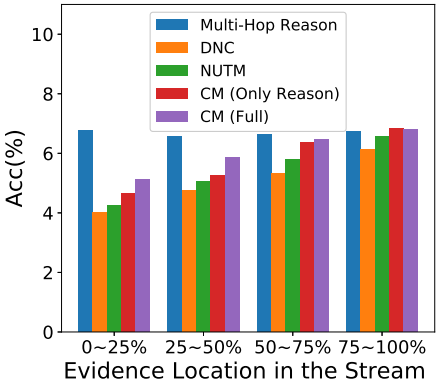

Figure 4: Performance on Synthetic Data with Longer Stream and Evidence.

in different locations of the input stream. For example, 25∼50% means the facts appear between 25% and 50% of the stream. We can observe that our CM(Full) has an obvious performance improvement compared to memory-based methods in the 0∼25% and 25∼50% stage, but achieves a slight improvement in the 75∼100% stage. This verifies our self-supervised training is beneficial for long-term memorization.

Table 2: Performance Comparisons for Long-Term Text Question Answering on NarrativeQA.

Method	Setting	Val	Test
AS Reader	End-to-End	26.9	25.9
E2E-MN	End-to-End	29.1	28.6
DNC	Memory-Based	25.8	25.2
NUTM	Memory-Based	27.7	27.2
CM (Only Reason)	Memory-Based	28.0	27.6
CM (Full)	Self-Sup. Memory	**28.8**	**28.1**

Table 3: Performance Comparisons for Long-Term Text Question Answering on TVQA.

Method	Setting	Acc.
Multi-Stream	End-to-End	63.14
MSAN	End-to-End	65.77
DNC	Memory-Based	55.92
NUTM	Memory-Based	58.41
CM (Only Reason)	Memory-Based	59.87
CM (Full)	Self-Sup. Memory	**61.12**

4.3 Long-term Text Question Answering

Dataset, Baseline and Model Details. We apply two multi-choice datasets NarrativeQA (Kočiskỳ et al., 2018) and TVQA (Lei et al., 2018) for long-term text question answering. The two datasets provide the answer candidates thus they are suitable for the setting "reasoning after memorizing". Note that the TVQA dataset also provides the video contents as input but we only use the subtitles in the videos. For NarrativeQA, the **AS Reader** (Kadlec et al., 2016) applies a pointer network to generate the answer and **E2E-MN** (Sukhbaatar et al., 2015) employs the end-to-end memory network to conduct multi-hop reasoning. For TVQA, the **Multi-Stream** (Lei et al., 2018) method develops the query-subtitle interaction for reasoning. And **MSAN** (Kim et al., 2020) first localizes the clues relevant to the question and then predicts the answer. For our continual memory, we set the d_x, d_m and d_{model} to 256. The number K of memory slots and length N of segments are both set to 20. And we sample all other words as negative items in $\mathcal{L}_{item}$.

Evaluation Results. As shown in Table 2 and Table 3, the results are similar to synthetic experiments, i.e. our continual memory obtains the best performance among the memory-based approaches. And CM(Full) achieves the further improvement compared to CM(Only Reason), demonstrating the self-supervised memorization training can boost the reasoning ability of continual memory. Moreover, our proposed method achieves the accuracy close to the excellent end-to-end method E2E-MN in the NarrativeQA dataset, but still has a large performance gap with MSAN in the TVQA dataset. This is mainly because the MSAN method applies the two-stage reasoning with elaborate interactions between texts and queries.

Table 4: Performance Comparisons for Long-Term Video Question Answering on ActivityNet-QA.

Method	Setting	Acc.
E-VQA	End-to-End	25.2
E-MN	End-to-End	27.9
E-SA	End-to-End	31.8
HCRN	End-to-End	37.6
DNC	Memory-Based	30.3
NUTM	Memory-Based	33.1
CM (Only Reason)	Memory-Based	34.6
CM (Full)	Self-Sup. Memory	**35.7**

Table 5: Performance Comparisons for Lifelong Sequence Recommendation on XLong.

Method	Setting	AUC
GRU4REC	End-to-End	0.8702
Caser	End-to-End	0.8390
RUM	End-to-End	0.8649
DIEN	End-to-End	0.8793
HPMN	Memory-Based	0.8645
MIMN	Memory-Based	0.8731
CM (Only Reason)	Memory-Based	0.8756
CM (Full)	Self-Sup. Memory	**0.8824**

4.4 Long-Term Video Question Answering

Dataset, Baseline and Model Details: The ActivityNet-QA dataset (Yu et al., 2019) contains 5,800 videos from the ActivityNet (Caba Heilbron et al., 2015). The average video duration of this dataset is about 180s and is the longest in VQA datasets. We compare our method with three basic end-to-end baselines **E-VQA**, **E-MN**, **E-SA** from (Yu et al., 2019) and the SOTA end-to-end model **HCRN** (Le et al., 2020). For our continual memory, we set the d_x, d_m and d_{model} to 256. The number K of memory slots and length N of segments are both set to 20. And in $\mathcal{L}_{item}$, we select 30 other frame features from the video as the sampled items.

Evaluation Results. As shown in Table 4, the CM(Only Reason) obtains the better performance than other memory-based models DNC and NUTM, and CM(Full) further achieves the 1.1% absolute improvement, showing the effectiveness of our model designs and self-supervised memorization training. Moreover, our method outperforms the basic end-to-end baselines and slightly worse than the SOTA method HCRN. This suggests our continual memory can reduce the gap between memory-based and end-to-end reasoning paradigms.

4.5 Lifelong Sequence Recommendation

Dataset, Baseline and Model Details. The lifelong sequence recommendation aims to predict whether the user will click a given item based on long sequences, thus it can be regarded as a long-term reasoning task. The XLong dataset (Ren et al., 2019) is sampled from the click logs on Alibaba. The length of historical behavior sequences in this dataset is 1000. We compare our method with four end-to-end methods **GRU4REC** (Hidasi et al., 2015), **Caser** (Tang & Wang, 2018), **DIEN** (Zhou et al., 2019), **RUM** (Chen et al., 2018) and two memory-based methods **HPMN** (Ren et al., 2019) and **MIMN** (Pi et al., 2019), where the HPMN method builds the memory by hierarchical RNNs and the MIMN method introduces a write-read memory as in (Graves et al., 2014). For our continual memory, we set the d_x, d_m and d_{model} to 64. The number K of memory slots and length N of segments are both set to 20. And in $\mathcal{L}_{item}$, we select 200 items from the large item set as the sampled items.

Evaluation Results. As shown in Table 5, our CM(Full) method not only outperforms other memory-based approaches, but also achieves better performance than state-of-the-art end-to-end baselines. This is because our continual memory can aggregate and organize the long-term interests from user behavior sequences and these interests can be activated during next-item prediction. But the end-to-end approaches may fail to learn such informative interest representations.

5 Conclusions

In this paper, we propose the continual memory to explore the ability of reasoning after long-term memorization. We compress the input stream into continual memory with the self-supervised memorization training and task-specific reasoning training. Based on continual memory, we can capture the clues for the subsequent queries and give the correct responses. Extensive experiments on a series of downstream tasks verify the effectiveness of the continual memory. For future work, we will further explore the property of continual memory.

REFERENCES

Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 961–970, 2015.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. Sequential recommendation with user memory networks. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pp. 108–116, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference on The North American Chapter of the Association for Computational Linguistics*, 2019.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Travis Ebesu, Bin Shen, and Yi Fang. Collaborative memory network for recommendation systems. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 515–524, 2018.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538 (7626):471–476, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*, 2016.

Junyeong Kim, Minuk Ma, Trung Pham, Kyungsu Kim, and Chang D Yoo. Modality shifting attention network for multi-modal video question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10106–10115, 2020.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018.

Hung Le, Truyen Tran, and Svetha Venkatesh. Learning to remember more with less memorization. *arXiv preprint arXiv:1901.01347*, 2019a.

Hung Le, Truyen Tran, and Svetha Venkatesh. Neural stored-program memory. *arXiv preprint arXiv:1906.08862*, 2019b.

Thao Minh Le, Vuong Le, Svetha Venkatesh, and Truyen Tran. Hierarchical conditional relation networks for video question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9972–9981, 2020.

Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L Berg. Tvqa: Localized, compositional video question answering. *arXiv preprint arXiv:1809.01696*, 2018.

Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. Practice on long sequential user behavior modeling for click-through rate prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2671–2679, 2019.

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.

Kan Ren, Jiarui Qin, Yuchen Fang, Weinan Zhang, Lei Zheng, Weijie Bian, Guorui Zhou, Jian Xu, Yong Yu, Xiaoqiang Zhu, et al. Lifelong sequential modeling with personalized memorization for user response prediction. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 565–574, 2019.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pp. 2440–2448, 2015.

Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 565–573, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

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

Zhou Yu, Dejing Xu, Jun Yu, Ting Yu, Zhou Zhao, Yueting Zhuang, and Dacheng Tao. Activitynet-qa: A dataset for understanding complex web videos via question answering. In *Proceedings of the American Association for Artificial Intelligence*, volume 33, pp. 9127–9134, 2019.

Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. Deep interest evolution network for click-through rate prediction. In *Proceedings of the American Association for Artificial Intelligence*, volume 33, pp. 5941–5948, 2019.

A APPENDIX

A.1 TASK-SPECIFIC REASON MODELS

In this section, we introduce the task-specific reason model $\mathcal{R}_\Omega(\mathbf{M}, \mathbf{Q})$, where $\mathbf{M}$ is the built memory and $\mathbf{Q}$ is the given query. Specifically, we first model the query feature $\mathbf{q} \in \mathbb{R}^{d_{model}}$ by a task-specific encoder. For synthetic experiments, the given query $\mathbf{Q}$ is a one-hot vector and we directly obtain $\mathbf{q}$ by an embedding layer. For long-term text and video QA tasks, the query $\mathbf{Q}$ is a sentence and we apply a bi-directional GRU to learn the sentence feature $\mathbf{q}$. As for the recommendation task with long sequences, the given query is a target item with the unique id and we likewise learn an embedding layer to obtain the feature $\mathbf{q}$.

Next, we develop the multi-hop attention-based reasoning on continual memory $\mathbf{M}$. Concretely, at each step c, we capture the importance memory feature $\mathbf{e}^c \in \mathbb{R}^{d_m}$ from $\mathbf{M}$ based on the current query $\mathbf{q}^{c-1}$ using an attention method, given by

$$\gamma_k^c = \mathbf{w}_c^\top \tanh(\mathbf{W}_1^c \mathbf{q}^{c-1} + \mathbf{W}_2^c \mathbf{m}_k + \mathbf{b}^c),\ \hat{\gamma}_k^c = \frac{\exp(\gamma_k^c)}{\sum_{j=1}^K \exp(\gamma_j^c)},\ \mathbf{e}^c = \sum_{k=1}^K \hat{\gamma}_k^c \mathbf{m}_k,$$

where $\mathbf{W}_1^c \in \mathbb{R}^{d_{model} \times d_{model}}$, $\mathbf{W}_2^c \in \mathbb{R}^{d_{model} \times d_m}$ and $\mathbf{b}^c \in \mathbb{R}^{d_{model}}$ are the projection matrices and bias. And $\mathbf{w}_c^\top$ is the row vector. We then produce the next query $\mathbf{q}^c = \mathbf{W}^q[\mathbf{e}^c; \mathbf{q}^{c-1}] \in \mathbb{R}^{d_{model}}$, where $\mathbf{W}^q \in \mathbb{R}^{d_{model} \times (d_m + d_{model})}$ is the projection matrix and $\mathbf{q}^0$ is the original $\mathbf{q}$. After C steps, we obtain the reason feature $\mathbf{q}^C$. The hyper-parameter C is set to 2, 2, 2 and 1 for synthetic experiments, text QA, video QA and sequence recommendation, respectively.

After it, we design the final reasoning layer for different tasks. For synthetic experiments and long-term video QA with fixed answer sets, we directly apply a classification layer to select the answer and develop the cross-entropy loss $\mathcal{L}_r$. But text QA datasets NarrativeQA and TVQA provide different candidate answers for each query, we first model each candidate feature $\mathbf{a}_i$ by another bi-directional GRU and then concatenate $\mathbf{a}_i$ with $\mathbf{q}^C$ to predict the conference score for each candidate. Finally, we also learn the cross-entropy loss $\mathcal{L}_r$ based on answer probabilities. As for the sequence recommendation task, we can directly compute a confidence score based on $\mathbf{q}^C$ by a linear layer and build the binary loss function $\mathcal{L}_r$.

A.2 Details of the Data Synthesis

In the synthetic task, we need to produce the data samples according to the dataset hyper-parameters R_f, R_l, R_q, R_a and R_c. Actually, we further define R_g different groups and each fact belongs to a group. As shown in Section 4.2, we first synthesize the $R_q * R_a$ fixed logic chains corresponding to each query-answer pair, where we have R_q types of queries and R_a answers for each query. Note that the facts in a logic chain belong to the same group, so these logic chains can also be categorized in different groups. Next, we synthetic 400 data samples for each logic chain to train the models. Each sample contains the input stream with R_l items, a query and an answer, where we guarantee each stream-query pair corresponds to a unique answer. We also define the consecutive and discrete evidence in Section 4.2. The facts in the discrete evidence are distributed in different stages of the stream, where we still make these facts exist in a certain interval that does not exceed 20% length of the input stream.