The Supplementary Material: Learning to Rehearse in Long Sequence Memorization

Zhu Zhang [*1,2] **Chang Zhou** [*2] **Jianxin Ma** [2] **Zhijie Lin** [1] **Jingren Zhou** [2] **Hongxia Yang** [2] **Zhou Zhao** [1]

A. 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 the synthetic task, the given query $\mathbf{Q}$ is a one-hot vector and we directly obtain $\mathbf{q}$ by an embedding layer. For long-sequence 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 rehearsal memory $\mathbf{M}$. Concretely, at each step c, we capture the importance memory feature $\mathbf{e}^c \in \mathbb{R}^{d_x}$ 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 $\mathbf{W}_1^c \in \mathbb{R}^{d_{model} \times d_{model}}$, $\mathbf{W}_2^c \in \mathbb{R}^{d_{model} \times d_x}$ 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 $\mathbf{W}^q \in \mathbb{R}^{d_{model} \times (d_x + d_{model})}$ 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-sequence 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 the text QA dataset NarrativeQA provides 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 $\mathcal{L}_r$.

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

Method	Setting	Early	Later
Directly Reason	Directly	13.57	13.41
Multi-Hop Reason	Directly	34.38	34.50
DNC	Memory-Based	20.56	26.59
NUTM	Memory-Based	24.31	29.71
STM	Memory-Based	23.55	29.64
DMSDNC	Memory-Based	24.92	30.74
RM (w/o. rehearsal)	Memory-Based	25.79	31.38
RM	Memory-Based	**28.42**	**31.71**

B. Synthetic Experiment

Synthetic Dataset. We first introduce the setting of the synthetic task. Here we abstract the general concepts of reasoning tasks (QA/VQA/Recommendation) to construct the synthetic task. We define the input sequence as a **Stream** and each item in the sequence as a **Fact**, where the stream and fact can correspond to the text sequence and word token in text QA. We set the number of fact types to R_f, that is, each fact can be denoted by a R_f-d one-hot vector and obtain the fact feature by a trainable embedding layer. Considering reasoning tasks often need to retrieve vital clues related to the query from the given input and then infer the answer, we define the query-relevant facts in the stream as the **Evidence** and regard the **Evidence**-**Query**-**Answer** triple as the **Logic Chain**. Given a stream and a query, we need to infer the answer if the stream contains the evidence. Specifically, we set the number of query types to R_q and each query can be denoted by a R_q-d one-hot vector.

[*]Equal contribution [1]Zhejiang University, China [2]DAMO Academy, Alibaba Group, China. Correspondence to: Zhou Zhao <zhaozhou@zju.edu.cn>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

For each query, we set the number of answer types to R_a. That is, there are totally $R_q * R_a$ query-answer pairs and we need to synthesize $R_q * R_a$ corresponding evidences of each pair. Each evidence is denoted by a sequence of facts $\{\text{fact}_1, \cdots, \text{fact}_{R_c}\}$, which continuously appear in the input stream. And R_c is the length of the evidence. During the evidence synthesis, we first define 20 different groups and uniformly split these facts and queries to 20 groups. Next, if a query belongs to group k, we randomly sample R_c facts from the group as the evidence, and then assign the evidence to a query-answer pair to generate a fixed logic chain.

Eventually, 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. Concretely, we first sample R_l facts as a sequence and then place the evidence in the sequence, where we guarantee each stream-query pair corresponds to a unique answer.

Baselines and Model Details. The **Directly Reason** method first models the input stream by RNN to obtain the stream feature, then concatenates the stream feature with the query feature and predicts the answer by a linear layer. 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 rehearsal memory, we set the d_x 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 $\mathcal{L}_{rec}$.

Evaluation Results. Table 1 reports the performance comparison between our method and baselines, where RM is the full model and RM (w/o. rehearsal) only employs the task-specific reasoning training. Overall, directly reasoning methods have close early and later performance, but memory-based approaches DNC, NUTM, STM, DMSDNC and RM (w/o. rehearsal) achieve the terrible early performance due to the gradual forgetting. By the self-supervised rehearsal training, our RM significantly improves the early accuracy and achieves the best memory-based reasoning performance. This fact suggests our proposed memory rehearsal can alleviate the gradual forgetting of early information and make the memory remember critical information from the input stream. Besides, RM (w/o. rehearsal) outperforms other memory-based methods, which indicates our rehearsal memory machine can better memorize the long-term information even without the rehearsal training. Moreover, we can find the Directly Reason approach achieves the worst performance but the Multi-Hop Reason method has a high accuracy, which demonstrates the performance of directly reasoning methods mainly depends on the complicated interaction between the input contents and queries.