Explicit Memory Learning with Expectation Maximization

Zhangyue Yin[◇] **Qiushi Sun**[♡] **Qipeng Guo**[♣] **Zhiyuan Zeng**[◇]
Qinyuan Cheng[◇] **Xipeng Qiu**[◇†] **Xuanjing Huang**[◇†]
[◇]School of Computer Science, Fudan University
[♡]The University of Hong Kong [♣]Shanghai AI Laboratory
{yinzy21,cengzy23,chengqy21}@m.fudan.edu.cn
qiushisun@connect.hku.hk guoqipeng@pjlab.org.cn
{xpqiu,xjhuang}@fudan.edu.cn

Abstract

Large Language Models (LLMs) have revolutionized the landscape of natural language processing, demonstrating remarkable abilities across various complex tasks. However, their stateless nature limits the capability to retain information across interactions, hindering performance in scenarios requiring historical context recall. To mitigate this, current approaches primarily use explicit memory to allow LLMs to store useful information, which is accessible, readable, and interpretable. Nevertheless, explicit memory lacks the reliable learning mechanisms of implicit memory, which can be optimized end-to-end. To harness the benefits of both, we introduce EM^2, a novel framework enhancing explicit memory updates via the Expectation-Maximization (EM) algorithm. EM^2 treats memory as a latent variable, ensuring continual learning and improvement during updates. Experimental results on streaming inference tasks demonstrate that EM^2 outperforms existing methods without memory or with static external memory. Our in-depth analysis highlights that EM^2 significantly enhances performance across various backbones and memory strategies, providing a robust solution for advancing LLM memory management and enabling explicit memory to learn and improve similarly to implicit memory.

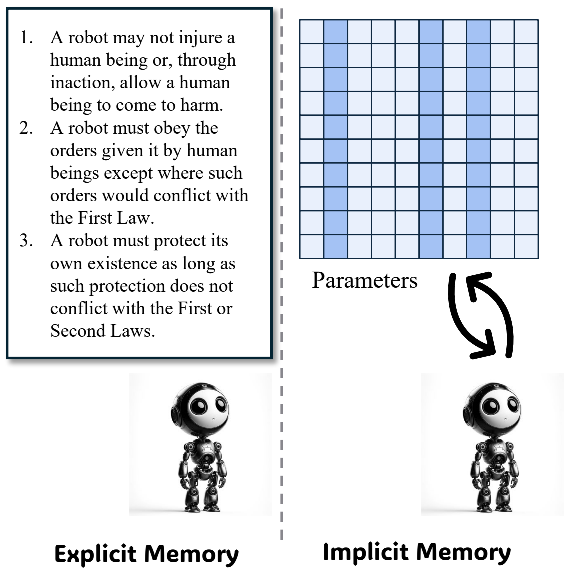


Figure 1: Comparison between Explicit and Implicit Memory. Explicit memory is represented through text, storing information directly accessible and readable. Implicit memory is stored in the form of parameters, which underlie the model's learned behaviors and are not directly interpretable. Deep blue indicates the memory currently being activated.

1 Introduction

The advent of Large Language Models (LLMs) has shifted the landscape of machine learning, unveiling unprecedented capabilities for handling complex tasks across diverse domains (Ouyang et al., 2022; Achiam et al., 2023; Anthropic, 2024; Reid et al., 2024; Shao et al., 2024; Sun et al., 2024b; Zhao et al., 2023, *inter alia*). Despite these advancements, a fundamental limitation of LLMs is their *statelessness*: they do not retain information across invocations (Yao, 2024). This restricts their ability to process and utilize previous interactions in a manner akin to human cognitive processes (Gabrieli, 1998), thereby limiting their utility in scenarios that require retention and recall of historical context (Zhang et al., 2024; Lu et al., 2024; Huang et al., 2023; Durante et al., 2024).

Recent studies have attempted to address this challenge by incorporating external memory mechanisms (Packer et al., 2023; Zhong et al., 2024), which can be categorized into explicit and implicit forms (Barco et al., 2006). As illustrated in Figure 1, explicit memory stores information in a textual format that is directly accessible and readable, such as rules, knowledge, and skills (Gao and Zhang, 2024; Guo et al., 2024). Implicit memory, on the other hand, is parametric, facilitating learning and updates (Wang et al., 2023a;

† Corresponding Authors

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 16618–16635

Ge et al., 2024). While parametric storage allows for end-to-end learning, it often faces issues with training stability (Franke et al., 2018), specification (Sukhbaatar et al., 2015), and interpretability (Zhang et al., 2021). With the increasing ability of LLMs to directly understand text (Brown et al., 2020; Wei et al., 2022a), explicit memory is becoming the dominant method for memory storage in LLMs (Madaan et al., 2022).

Updating is a critical feature of memory (Wang et al., 2024e; Li et al., 2023). Current methods of updating explicit memory include manual revisions (Mei et al., 2024) and self-reflection (Liu et al., 2023a; Shinn et al., 2023; Praas, 2023). Ge et al. (2023) conceptualize LLMs as an operating system (OS) and have developed memory update mechanisms inspired by OS design. Wang et al. (2024a) employ LLMs to autonomously summarize past experiences for enhanced external memory.

It is worth noticing that, LLMs may miss or make mistakes when internalizing knowledge (Yin et al., 2023; Wang et al., 2023b; Yao et al., 2023), and there is no guarantee that newly constructed memory is superior to its predecessors. In contrast, implicit memory, updated through gradients (Graves et al., 2016; Becattini and Uricchio, 2022), ensures learning during the memory update. Current methods for updating explicit memory do not guarantee learning and enhancement during the memory update process, marking a fundamental drawback. The primary reason is the non-differentiability of textual memory, which means that memory updates lack a clear direction.

To address this, we propose EM^2, which treats memory as a latent variable and update it using the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). EM^2 extracts relevant past experiences to guide current predictions and ensures that the memory is continuously optimized, enabling the model to learn and improve effectively over time. Experimental results on streaming inference tasks show that compared to models without external/fixed memory, our dynamic memory updating approach significantly enhances performance.

Our main contributions are as follows:

- We identify that current methods of updating explicit memory lack direction and do not ensure that updated memory is superior to previous versions.
- We introduce EM^2, which updates explicit memory using the EM algorithm to ensure continuous learning and enhancement during the memory update process.
- Experimental results demonstrate that EM^2 significantly improves model performance.

2 Related Work

2.1 Memory Mechanism of LLMs

Memory is fundamental to the development of intelligence (Anderson, 1999). Memory mechanisms in LLMs primarily involve retrieval (Gao et al., 2024), updating (Li et al., 2023), and utilization (Guo et al., 2024) processes. Retrieval aims to fetch relevant and accurate memories from a vast store, directly influencing the outcome's quality (He et al., 2022; Creswell and Shanahan, 2022). Updates include incremental, inductive, and compressive approaches. Incremental updates simply add newly acquired memories without processing them (Hong et al., 2023; Qian et al., 2023). Inductive updates utilize the LLM's capability to amalgamate and summarize memories, thereby narrowing the retrieval scope (Liu et al., 2023a; Didolkar et al., 2024). Compressive updates enhance the efficiency of memory use by condensing texts into vectors (Chevalier et al., 2023; Ge et al., 2024; Mu et al., 2024). The utilization of memory relies on the LLM's contextual understanding and learning capabilities, optimizing model behavior through the injection of text or parameters (Min et al., 2022; Liu et al., 2024; Wang et al., 2024d).

For LLMs, memory can be classified as explicit or implicit (Rovee-Collier et al., 2001; Barco et al., 2006). Explicit memory, also known as declarative memory, refers to forms of memory that can be articulated (Eichenbaum, 1997). It can be stored and retrieved in textual form (Sun et al., 2023; Zhong et al., 2024), offering readability and interpretability (Jiang et al., 2023b; Modarressi et al., 2024). Explicit memory does not depend on a specific model and can be utilized by various models post-generation (Gao and Zhang, 2024; Sun et al., 2024a). Additionally, humans can participate in modifying and refining explicit memory, making it widely applied in LLM memory modules (Wu et al., 2022). Implicit memory, on the other hand, refers to forms of memory that cannot be articulated. This type of memory is stored in parameters and updated through training (Weston et al., 2015; Anything, 2015; Sukhbaatar et al., 2015). Although explicit memory can also be updated through model-driven summarization and induction (Wang et al., 2024a;

Yang et al., 2024), it lacks the clear update targets characteristic of implicit memory, which ensures that the updated state is superior to its previous state.

2.2 Model Inference

The inference methods for LLMs predominantly encompass zero-shot, few-shot, and chain-of-thought (Chung et al., 2024). Zero-shot often requires model fine-tuning to equip LLMs with the capability to generate task-specific outputs directly (Raffel et al., 2020; Liu et al., 2021). Brown et al. (2020) observe that providing models with example prompts can significantly enhance their understanding of specific tasks. Currently, In-Context Learning (Dong et al., 2022) has emerged as a fundamental paradigm for addressing tasks using LLMs (Liu et al., 2023b), effectively leveraging minimal input to guide model responses (Min et al., 2022). Wei et al. (2022c) note that guiding models to generate intermediary reasoning steps will boost their performance for reasoning. This enhanced capability typically emerges only in models of certain scales, a phenomenon often referred to as "emergent abilities" (Wei et al., 2022a). Furthermore, recent studies (Wu et al., 2023; Li et al., 2024; Wang et al., 2024b) find that prompts serve a dual function: they not only activate the model's internal memory but also inject effective external knowledge and guidance. Additionally, updating and infusing memory in prompts offers benefits such as interpretability and flexibility (Chang et al., 2024), further enhancing the utility of LLMs in complex inference scenarios (Sahoo et al., 2024).

3 Preliminary and Task Definition

3.1 Explicit Memory Learning

Memory in AI are designed to mimic the human ability to remember past experiences and utilize this accumulated knowledge to aid in future tasks (Weston et al., 2015). In our model, explicit memory learning is implemented via a memory module $\mathcal{M}$ that stores strategies τ learned over time, which is formally represented as:

$$M_t = \{\tau_1, \tau_2, \ldots, \tau_K\}, \tag{1}$$

where M_t represents the state of the memory module at time t, K is the memory size, and each τ_i is a tactic derived from past experiences. The updating of this memory is governed by a learning function L, which adjusts the memory based on new experiences (X, Y):

$$M_{t+1} = L(M_t, (X_t, Y_t)). \tag{2}$$

Here, (X_t, Y_t) represents the input-output pair at time t, and the function L determines how the memory should be updated, possibly by adding new strategies, modifying existing ones, or removing outdated strategies based on their relevance and effectiveness in the new context.

3.2 Expectation Maximization Algorithm

The Expectation Maximization (EM) algorithm is a powerful statistical tool used for parameter estimation in models with latent variables. It operates in two main steps: the Expectation (E) step and the Maximization (M) step. During the E step, the algorithm estimates the latent variables based on the current estimate of the parameters:

$$Q(\theta|\theta^{(t)}) = \mathbb{E}_{Z\sim p(Z|X,\theta^{(t)})}[\log p(X, Z|\theta)], \tag{3}$$

where $\theta^{(t)}$ denotes the parameters at iteration t, X is the observed data, Z are the latent variables, and $p(Z|X, \theta^{(t)})$ is the probability of the latent variables given the observed data and current parameters.

The M step then updates the parameters to maximize the expected log-likelihood found in the E step:

$$\theta^{(t+1)} = \arg\max_{\theta} Q(\theta|\theta^{(t)}). \tag{4}$$

This iterative process continues until convergence, making it suitable for complex models where direct likelihood maximization is infeasible (Dempster et al., 1977). The EM algorithm is particularly effective in scenarios where the model parameters include both observed and unobserved (latent) components. By alternating between estimating the hidden components given the parameters and then optimizing the parameters given the hidden components, EM facilitates a more accurate estimation of model parameters.

3.3 Task Definition

Given the following stream of data $\mathcal{D} = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$, where X_t represents the observed data at time t and Y_t denotes the corresponding true label, the objective is to construct effective memory M_t that provides accurate predictions $\hat{Y}_t$.

Our primary goal is to minimize the discrepancy between the predicted labels $\hat{Y}_t$ and the actual labels Y_t. This is achieved by enhancing the predictive accuracy of the model under the guidance of the evolving memory M_t. The effectiveness of M_t is crucial as it directly influences the model's ability to adapt to new data and make accurate predictions. Therefore, the challenge lies in designing a learning function L that not only updates the memory efficiently but also ensures that these updates result in the accurate anticipation of future samples based on past and present data insights.

4 Methodology

4.1 Memory based Inference

At time t, the model receives an input X_t. In a zero-shot scenario, without any guidance from memory, the model ξ generates the predicted label $\hat{Y}_t$ in an autoregressive manner as follows:

$$P_\xi(\hat{Y}_t \mid X_t) = \prod_{i=1}^{|\hat{Y}_t|} P_\xi(\hat{y}_i \mid X_t, \hat{y}_{<i}) \quad (5)$$

To leverage past experiences stored in the memory, we enhance model's capability by introducing a memory-based guidance. Given the current input X_t, we extract the most relevant information from the current memory state M_t. This extraction process results in a memory subset m_t, defined as the set of elements in M_t that are most relevant to X_t. The relevance can be quantified based on similarity measures, heuristic rules, or learned relevance functions. The resulting m_t can be formally represented as:

$$m_t = \mathrm{select}(M_t, X_t) \quad (6)$$

where select is a function that retrieves the most relevant memory elements based on X_t.

With m_t as an additional context, the model then generates $\hat{Y}_t$ using both m_t and X_t to guide the prediction:

$$P_\xi(\hat{Y}_t \mid m_t, X_t) = \prod_{i=1}^{|\hat{Y}_t|} P_\xi(\hat{y}_i \mid m_t, X_t, \hat{y}_{<i}) \quad (7)$$

This memory-augmented inference mechanism allows the model to effectively utilize historical data, enhancing its predictive accuracy and adaptability in dynamic environments.

4.2 Memory Module Construction

The Memory Module $\mathcal{M}$ is constructed by accumulating pairs $(X_i, \hat{Y}_i)$ over time. Initially, the memory of the model is empty, representing a state of minimal prior knowledge. As the model processes data and generates predictions, it selectively updates this memory based on the quality and certainty of the information.

To quantify the certainty of each predicted output and determine its eligibility for memory inclusion, we define an uncertainty threshold ϵ. A prediction $\hat{Y}_i$ is considered high-quality if its normalized entropy, which measures the average uncertainty across all predicted components, is below this threshold. The entropy $H(\hat{Y}_i)$ for each prediction is calculated as follows:

$$H(\hat{Y}_i) = -\frac{1}{|\hat{Y}_i|} \sum_{j=1}^{|\hat{Y}_i|} \log P_\xi(\hat{y}_j \mid X_i, \hat{y}_{<j}) \leq \epsilon \quad (8)$$

When the above condition is satisfied, indicating that the generated prediction $\hat{Y}_i$ is of sufficiently high certainty and quality, it is integrated into the memory using the learning function L, as discussed in Section 3.1.

4.3 Memory Update through Learning Function

We employ the EM algorithm to design the learning function L. As depicted in Figure 2 under ② and ③, if the generated $\hat{Y}_i$ satisfies condition 8, it is fed along with the current memory state M_t into the learning function L. The update equation is:

$$M_{t+1} = L(M_t, (X_t, \hat{Y}_t)) \quad (9)$$

We treat strategies τ as latent variables Z and M_t as the parameter θ in Eq. 3, transforming the learning process into an EM learning framework.

4.3.1 Construction of Representative Validation Set

To evaluate the updates efficiently, we construct a representative validation set $\mathcal{V}$ from the dataset $\mathcal{D}$ not yet included in the memory M_t. We select cluster centers from $\mathcal{D} \setminus M_t$ to form $\mathcal{V}$, reducing redundancy and improving the efficiency of memory updates. The selection can be represented by:

$$V_t = \mathrm{centers}(\{(X_1, \hat{Y}_1), \ldots, (X_t, \hat{Y}_t)\} \setminus M_t) \quad (10)$$

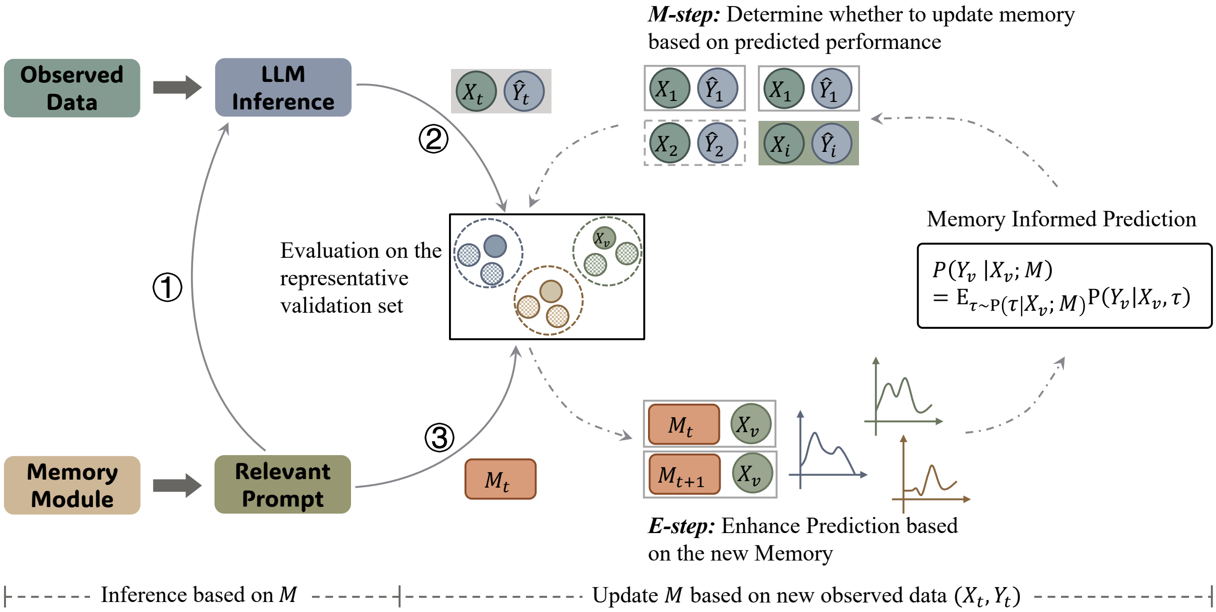


Figure 2: Overview of EM^2 for memory-guided prediction in streaming data. At each timestep t, the model receives an input X_t. ① utilizes the memory M_t to select relevant demonstrations that guide the generation of the prediction $\hat{Y}_t$. ② and ③ depict the integration of the newly generated $\hat{Y}_t$ and the current memory M_t into the memory updating process, ensuring that the memory evolves with the latest data insights and contributes to future predictions.

4.3.2 E-step: Inference Procedure

Let $\mathcal{V}_t = \{(X_v, Y_v)\}$. Based on Equation 3, the prediction for Y_v given X_v and the memory M is calculated as:

$$\begin{aligned} P(Y_v \mid X_v; M) &= \sum_{\tau} P(Y_v, \tau | X_v; M) \\ &= \sum_{\tau} P(Y_v | X_v, \tau) P(\tau | X_v; M) \\ &= \mathbb{E}_{\tau \sim P(\tau|X_v;M)}[P(Y_v|X_v, \tau)] \end{aligned} \tag{11}$$

4.3.3 M-step: Learning Procedure

The memory is updated based on the maximization step defined as:

$$M_{t+1} = \underset{m \subset M_t \cup \Gamma(X_t, \hat{Y}_t)}{\arg\max} \sum_{i=1}^{|\mathcal{V}_t|} P(Y_i | X_i; m), \tag{12}$$

where Γ represents a function extracting knowledge from $(X_t, \hat{Y}_t)$ to generate τ_t, which can be formally represented as:

$$\tau_t = \Gamma(X_t, \hat{Y}_t) \tag{13}$$

This step ensures that the updated memory M_{t+1} performs better on $\mathcal{V}_t$ than the previous state M_t, effectively capturing the beneficial strategies for future predictions.

5 Experiment

5.1 Evaluation Datasets

To assess the efficacy of our approach, we evaluate it across three distinct types of tasks: word math problems, commonsense question answering (QA), and symbolic analysis. We utilize the following datasets for these evaluations:

- *Word Math Problem:* GSM8K (Cobbe et al., 2021), MultiArith (Roy and Roth, 2015), SingleEq (Koncel-Kedziorski et al., 2016), AddSub (Hosseini et al., 2014), SVAMP (Patel et al., 2021), AQUA (Ling et al., 2017) and MATH (Hendrycks et al., 2021).
- *Commonsense QA:* StrategyQA (Geva et al., 2021), CommonsenseQA (CSQA; Talmor et al., 2019), BoolQ (Clark et al., 2019), the AI2 Reasoning Challenge (ARC-c; Clark et al., 2018).
- *Symbolic Understanding:* Date Understanding, Penguins in a Table, Colored Objects, and Object Counting sourced from Big-Bench (Suzgun et al., 2023).

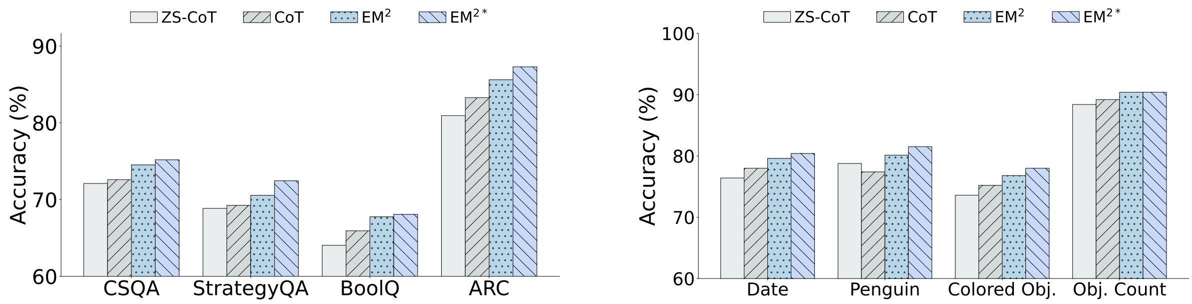
For a more detailed description of the datasets, please refer to Appendix A.

5.2 Experiment Settings

Implementation Details. The inference process of the model not only demonstrates its understand-

	GSM8K	MultiArith	SingleEq	AddSub	SVAMP	AQuA	MATH	Average
				Single Inference				
ZS-CoT	76.80	94.83	89.96	84.30	81.45	40.55	29.02	77.98
CoT	79.61	96.50	92.32	85.31	82.76	42.32	-	79.80
ComplexCoT	78.01	96.67	91.92	84.81	81.48	42.51	29.50	79.23
EM^2	82.63	97.77	**92.71**	86.32	83.91	45.27	30.12	81.43
EM^{2*}	**83.09**	**97.83**	**92.71**	**87.59**	**84.19**	**46.45**	**30.22**	**81.98**
				Multiple Inference				
ZS-CoT	84.98	97.50	92.71	88.61	87.18	47.24	32.22	83.03
CoT	85.59	98.00	94.29	91.13	91.76	51.57	-	85.39
ComplexCoT	85.29	98.16	93.70	89.87	89.62	50.78	32.46	84.57
EM^2	86.35	**98.83**	**95.86**	93.41	92.51	53.14	33.82	86.68
EM^{2*}	**86.43**	**98.83**	95.66	**94.43**	**92.55**	**53.93**	**33.96**	**86.97**

Table 1: Results on Math Word Problems (Accuracy in %). The best outcomes are emphasized in **bold**. Average represents the average performance across all datasets, excluding MATH. EM^2 denotes initialization using ZS-CoT, while EM^{2*} indicates initialization with CoT demonstrations, highlighted with a skyblue background. To ensure a fair comparison, the LLaMA-3-8B model (Dubey et al., 2024) is used as the backbone across all methods.



(a) Commonsense QA

(b) Symbolic Understanding

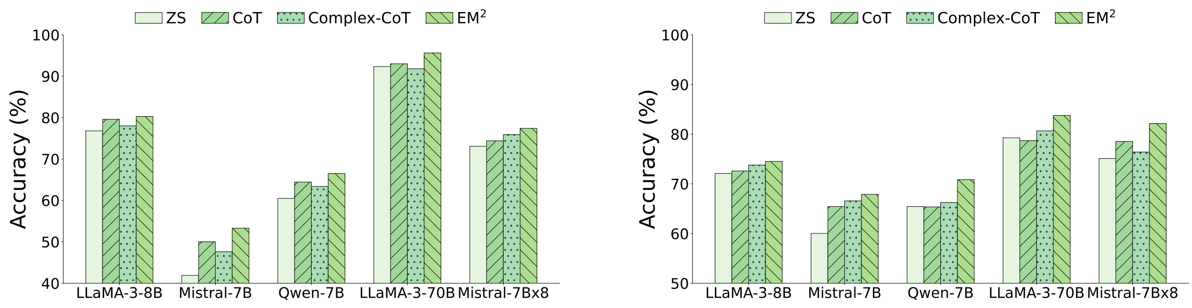
Figure 3: Performance comparison on (a) commonsense question answering and (b) symbolic understanding tasks. The charts illustrate that EM^2 demonstrates a distinct advantage over both no and fixed-memory mechanisms.

ing and analysis of problems but often encapsulates latent knowledge (Buehner et al., 2005). Therefore, we store the model's reasoning process along with the problem as the model memory. In the main experiments, memory is vectorized using `text-embedding-3-large`, and relevancy is calculated using cosine distance as specified in Eq. 6. To ensure fair comparisons, we limit the selection to a maximum of 8 examples. These vectors are also employed to determine the clustering centers as outlined in Eq. 10. For more details and ablation studies, see Appendix B and C.

Baselines. To validate the efficacy of our approach, we compare it against three baseline methods representing different levels of memory integration: models without memory, with fixed memory, and with retrieval-based memory.

- *No Memory:* The Zero-shot CoT (ZS-CoT; Kojima et al., 2022) utilizes the prompt "Let's think step by step" to activate the model's internal reasoning capabilities without relying on external memory aids.
- *Fixed Memory:* The Chain-of-Thought (CoT; Wei et al., 2022b) employs fixed prompts to guide the model through a reasoning process. ComplexCoT (Fu et al., 2023) extends this by using complex prompts that guide the model to generate more detailed reasoning processes.
- *Retrieval Memory:* The Memory-of-Thought (MoT; Li and Qiu, 2023) incorporates a two-stage memory retrieval process, which includes coarse-grained semantic retrieval followed by fine-grained model filtering to select relevant memories. AutoCoT (Zhang et al., 2023) selects examples based on relevance and diversity metrics tailored to the query.

In contrast to the main experiment where memory updates are conducted using test samples, MoT and AutoCoT require pre-inference on training data. To ensure a fair comparison, we align the settings with these methods to in Section 5.4.



(a) GSM8K

(b) CSQA

Figure 4: Performance comparison of different memory mechanisms across various LLMs.

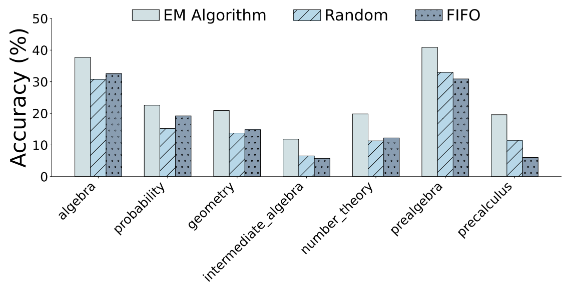


Figure 5: Performance of different memory updating mechanisms on the MATH dataset.

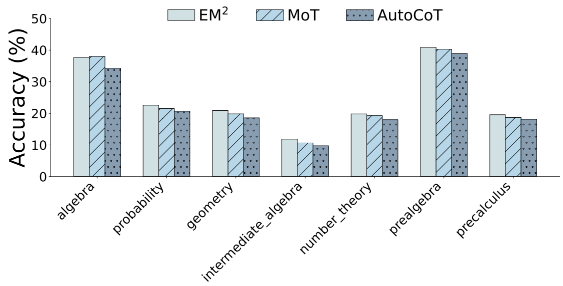


Figure 6: Performance comparison of retrieval-based memory methods on the MATH dataset.

Backbones. In the main experiment, we employ LLaMA-3-8B (Dubey et al., 2024). For analysis, we extend our investigations to include more LLMs, including LLaMA-3-70B (Dubey et al., 2024), Mistral-7B (Jiang et al., 2023a), Mixtral (Jiang et al., 2024a), and Qwen-2 (Bai et al., 2023).

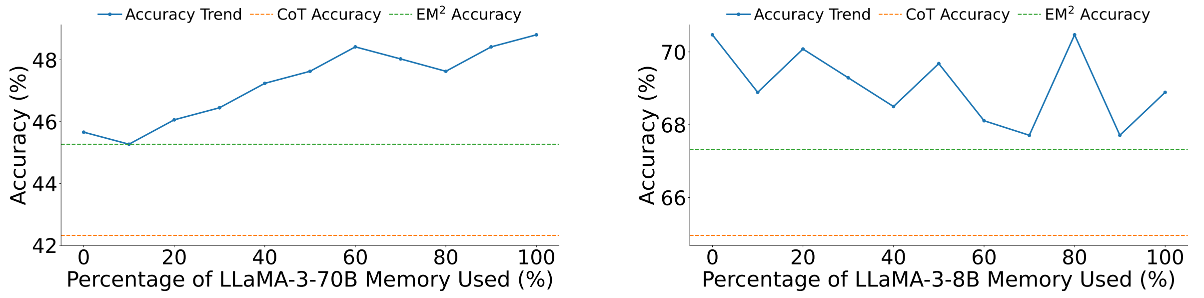
5.3 Main Results

Word Math Problem. Table 1 presents the results of math word problems. Compared to methods with no memory or fixed memory, our memory learning approach exhibits significant advantages. Notably, on the GSM8K dataset, EM^2 outperforms the ZS-CoT by 5.83% and CoT by 3.02%. This improvement is attributed to the dynamic memory updating mechanism of EM^2. We utilize two initialization methods: ZS-CoT, where the initial memory is empty, and CoT, which provides eight high-quality demonstrations at initialization. While the CoT initialization ensures better initial performance, the efficacy of both approaches converges as the memory accumulates. For instance, on the SingleEq dataset, results from both initialization methods are identical. Further, we analyze multiple inference scenario (Wang et al., 2023c) and observe that EM^2 retains a clear advantage. Moreover, as more memories are integrated, the performance gap between the two initialization methods narrows.

Commonsense QA and Symbolic. The experimental results for commonsense QA and symbolic understanding tasks are shown in Figure 3. We observe that EM^2 effectively enhances model performance on both types of tasks. Notably, EM^2 demonstrates a more pronounced advantage in challenging tasks, such as those involving complex, non-factoid information in the BoolQ dataset, and tasks requiring implicit multi-step reasoning in the StrategyQA dataset. This improvement can be attributed to EM^2's memory updating and retrieval mechanisms, which ensure the selection of high-quality and relevant demonstrations.

5.4 Analysis and Discussion

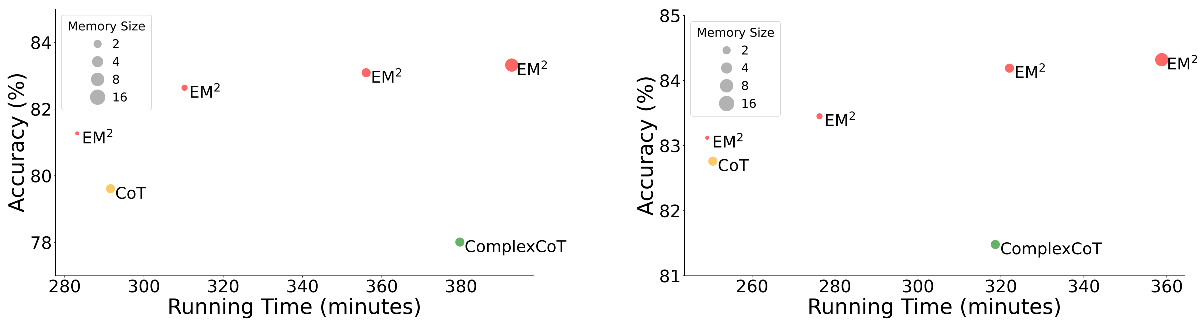
Performance on Various Models. The performance of EM^2 across a range of models is analyzed in Figure 4, focusing on two representative datasets: GSM8K and CSQA. We observe that EM^2 consistently delivers significant performance enhancements across different models. Notably, models with greater computational capabilities benefit more substantially from the EM^2 approach. For instance, despite having a similar number of parameters, Qwen-7B exhibits a greater improvement than Mistral-7B. Moreover, EM^2 proves to be ver-



(a) 8B model accesses 70B model's memory

(b) 70B model accesses 8B model's memory

Figure 7: Impact of memory swapping on model performance. The horizontal axis represents the proportion of memory injected. The horizontal lines indicate the baseline accuracies for models with fixed memory and EM^2 initialized with ZS-CoT.



(a) GSM8K

(b) SVMAP

Figure 8: Comparison of EM^2 with varying memory sizes and fixed memory methods in terms of runtime and accuracy. The horizontal axis represents the runtime in minutes, the vertical axis shows accuracy, and the size of the points indicates the size of the memory.

satile, not only enhancing the performance of dense models but also boosting the efficacy of Mixture of Experts (MoE) models like Mixtral. This adaptability underscores EM^2's effectiveness in leveraging complex memory dynamics across different architectural frameworks.

Analysis of Memory Updating Mechanism. The impact of different memory updating strategies on accuracy is analyzed in Figure 5. We experimented with replacing the learning function in Section 4.3 with two simpler updating strategies: random selection and First-In-First-Out (FIFO) (Manurung, 2019). Results on the MATH dataset, particularly in the precalculus subset, show that these changes significantly reduce model performance. The primary reason for this decline can be attributed to the inherent limitations of Random and FIFO strategies, which rely on randomness and sample order, respectively, and cannot guarantee the effectiveness of memory updates. This analysis highlights the efficacy of the EM^2 approach, which employs the EM algorithm to ensure gradual and effective optimization of memory.

Comparison of Memory Retrieval Method. In Figure 6, we compare the EM^2 with two memory retrieval methods. Both MoT and AutoCoT require pre-inference on the training dataset to gather examples for retrieval. To ensure a fair comparison, we incorporate training samples into EM^2, first performing memory updates and constructing a representative validation set on the training dataset, before introducing the test set for accuracy calculations. Results on the MATH dataset demonstrate that EM^2 achieves superior performance compared to traditional memory retrieval methods. Despite having a narrower search scope compared to the broader retrieval range of MoT and AutoCoT, the EM^2's updating strategy ensures the retention of high-quality memories. Moreover, continuous updates maintain alignment between the memory distribution and the test distribution, thereby resulting in enhanced performance.

Analysis of Memory Sharing. The memory constructed by EM^2 is model-agnostic, enabling the transfer and sharing of memories between models. In Figure 7, we explore the effects of exchanging

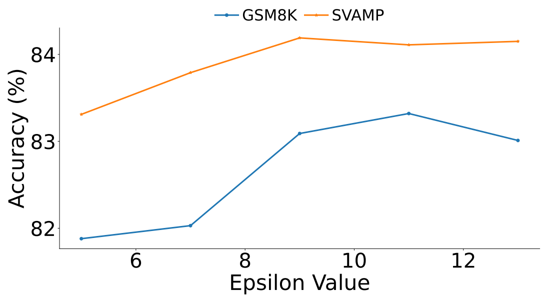


Figure 9: Impact of varying the threshold ϵ on model performance.

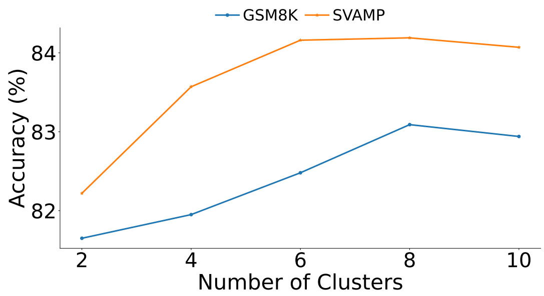


Figure 10: Impact of varying the number of clusters on model performance.

memories between LLaMA-3-8B and LLaMA-3-70B. Each model first performs inference on the training dataset, after which their memories are swapped. As shown in Figure 7a, there is a gradual improvement in the performance of the 8B model as the proportion of memory from the 70B model increases. This indicates that smaller models can benefit from high-quality memories sourced from larger models. Conversely, Figure 7b reveals that the performance of the 70B model remains unaffected by the memory from the 8B model, as lower-quality memories do not enter our memory module.

Analysis of Memory Size. In Figure 8, we analyze the impact of memory size on accuracy and running time. We observe that on the GSM8K and SVAMP datasets, when the number of demonstrations in memory m_t is reduced to two, the running time becomes comparable to the method with CoT (Wei et al., 2022c). Thanks to the effective memory updating strategy of EM^2, the performance remains significantly superior to the CoT method even with the reduced number of demonstrations. The ComplexCoT method (Fu et al., 2023), which requires multi-step detailed derivations, demands more reasoning time. We note that the running times of ComplexCoT and EM^2 with a memory size of eight are comparable, yet EM^2 significantly outperforms ComplexCoT in terms of accuracy. The additional computational time for EM^2 is attributed to the M-step in Section 4.3.3, whereas the memory update does not involve costly decoding processes, thus not incurring significant overhead.

Analysis of Threshold ϵ. In Figure 9, we analyze the impact of variations in the threshold ϵ from Eq. 8 on model performance. The results on datasets such as GSM8K and SVAMP indicate that a lower threshold allows low-quality information to enter the memory, which in turn degrades the model's performance. Conversely, setting the threshold too high significantly reduces the amount of information entering the memory, diminishing the diversity of the stored data. Therefore, setting ϵ to 9 offers an optimal balance between high-quality information and diversity within the memory.

Analysis of Number of Clusters. In Figure 10, we evaluate the impact of different cluster counts on model performance. The results on the GSM8K and SVAMP datasets show that a smaller number of clusters reduces the diversity of samples in the representative validation set, which in turn can lower model performance. Initially, when there are fewer samples available, it is challenging to form a meaningful number of clusters. Therefore, setting the number of clusters to eight is found to be appropriate for achieving a good balance between clustering quality and the meaningful segmentation of data.

6 Conclusion

In this paper, we analyze the advantages of explicit memory over implicit memory and highlight a critical limitation of the former: its inability to ensure the effectiveness of updates as reliably as implicit memory. To address this, we introduce EM^2, which treats memory as a latent variable and iteratively updates it using the EM algorithm, thereby ensuring that updated memories are superior to their predecessors. Experiments show that EM^2 offers significant advantages over models without memory and those with fixed memory. Importantly, the performance of EM^2 scales with the model's capabilities, suggesting that more powerful models can leverage EM^2 to achieve even greater benefits. Additionally, EM^2 is model-agnostic, which allows for the transfer and sharing of memory across different models. Analyses reveal that weaker LLMs can significantly benefit from high-quality memories derived from larger counterparts.

Limitations

Generalization to a Broader Range of Tasks. While we have analyzed EM^2 across three distinct types of tasks, there is potential to extend this approach to a wider array of generative tasks (Gozalo-Brizuela and Garrido-Merchán, 2023), such as code generation (Jiang et al., 2024b), machine translation (Ganesh et al., 2023), and various agent-based tasks (Wang et al., 2024c). Additionally, the form of memory could also be diversified to include structured data, triplets, user historical information, and more. Our current scope has not yet explored these domains, and we see the exploration of EM^2's potential in more diverse tasks as an avenue for future work.

Application to Commercial Models. EM^2 requires access to internal model information, such as perplexity, to assess the effectiveness of new memories. However, for commercial models that only provide text outputs, such as OpenAI's GPT models (Achiam et al., 2023) or Anthropic's Claude models (Anthropic, 2024), despite their powerful capabilities, applying EM^2 remains challenging.

Incorporating Human Supervision. As mentioned in Section 5.4, higher-quality memories can significantly enhance model performance. This paper primarily focuses on memories constructed autonomously by the model. An intriguing question is whether human-supervised memory enhancement and correction could further improve performance. Additionally, how to effectively incorporate human supervision (Wu et al., 2022), such as step-by-step guidance (Lightman et al., 2023), remains an open question for future research.

Ethics Statement

Data Privacy. Our approach constructs memory from the model's own outputs and does not require the collection or acquisition of personal data. The prompts and data used in our experiments do not involve any personal or privacy-sensitive information, ensuring compliance with privacy standards.

Environmental Protection. The construction of large language models and the generation of data and memory are likely to become more prevalent, consuming significant computational resources and potentially increasing carbon emissions. We advocate for sustainable AI development, emphasizing the reduction of carbon footprints and the promotion of green AI initiatives to mitigate environmental impacts.

Adherence to Ethical Guidelines. We adhere to ethical guidelines and ensure that our data usage complies with the corresponding dataset licenses. Detailed statistics about the datasets and their respective licenses is listed in Table 2.

Acknowledge

This work was supported by the National Natural Science Foundation of China (No. 62236004). The computations in this research were performed using the CFFF platform of Fudan University. We would like to express our sincere gratitude to all the reviewers for their valuable suggestions and assistance, which have significantly contributed to the improvement of this manuscript.

References

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

Mike Anderson. 1999. *The development of intelligence*. Psychology Press.

AI Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*.

Ask Me Anything. 2015. Dynamic memory networks for natural language processing. *Kumar et al. arXiv Pre-Print*, 97.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Angel Barco, Craig H Bailey, and Eric R Kandel. 2006. Common molecular mechanisms in explicit and implicit memory. *Journal of neurochemistry*, 97(6):1520–1533.

Federico Becattini and Tiberio Uricchio. 2022. Memory networks. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page

7380–7382, New York, NY, USA. Association for Computing Machinery.

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

Markus Buehner, Stefan Krumm, and Marion Pick. 2005. Reasoning= working memory≠ attention. *Intelligence*, 33(3):251–272.

Kaiyan Chang, Songcheng Xu, Chenglong Wang, Yingfeng Luo, Tong Xiao, and Jingbo Zhu. 2024. Efficient prompting methods for large language models: A survey. *Preprint*, arXiv:2404.01077.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Antonia Creswell and Murray Shanahan. 2022. Faithful reasoning using large language models. *Preprint*, arXiv:2208.14271.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22.

Aniket Didolkar, Anirudh Goyal, Nan Rosemary Ke, Siyuan Guo, Michal Valko, Timothy Lillicrap, Danilo Rezende, Yoshua Bengio, Michael Mozer, and Sanjeev Arora. 2024. Metacognitive capabilities of llms: An exploration in mathematical problem solving. *Preprint*, arXiv:2405.12205.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Zane Durante, Qiuyuan Huang, Naoki Wake, Ran Gong, Jae Sung Park, Bidipta Sarkar, Rohan Taori, Yusuke Noda, Demetri Terzopoulos, Yejin Choi, et al. 2024. Agent ai: Surveying the horizons of multimodal interaction. *arXiv preprint arXiv:2401.03568*.

Howard Eichenbaum. 1997. Declarative memory: Insights from cognitive neurobiology. *Annual review of psychology*, 48(1):547–572.

Jörg Franke, Jan Niehues, and Alex Waibel. 2018. Robust and scalable differentiable neural computer for question answering. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 47–59, Melbourne, Australia. Association for Computational Linguistics.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*.

John DE Gabrieli. 1998. Cognitive neuroscience of human memory. *Annual review of psychology*, 49(1):87–115.

Sahana Ganesh, Vedant Dhotre, Pranav Patil, and Dipti Pawade. 2023. A comprehensive survey of machine translation approaches. In *2023 6th International Conference on Advances in Science and Technology (ICAST)*, pages 160–165.

Hang Gao and Yongfeng Zhang. 2024. Memory sharing for large language model based agents. *Preprint*, arXiv:2404.09982.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations*.

Yingqiang Ge, Yujie Ren, Wenyue Hua, Shuyuan Xu, Juntao Tan, and Yongfeng Zhang. 2023. Llm as os, agents as apps: Envisioning aios, agents and the aios-agent ecosystem. *Preprint*, arXiv:2312.03815.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Roberto Gozalo-Brizuela and Eduardo C. Garrido-Merchán. 2023. A survey of generative ai applications. *Preprint*, arXiv:2306.02781.

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

Jing Guo, Nan Li, Jianchuan Qi, Hang Yang, Ruiqiao Li, Yuzhen Feng, Si Zhang, and Ming Xu. 2024. Empowering working memory for large language model agents. *Preprint*, arXiv:2312.17259.

Hangfeng He, Hongming Zhang, and Dan Roth. 2022. Rethinking with retrieval: Faithful large language model inference. *Preprint*, arXiv:2301.00303.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533. Association for Computational Linguistics.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1051–1068, Singapore. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. Mistral 7b. *Preprint*, arXiv:2310.06825.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024a. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023b. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024b. A survey on large language models for code generation. *Preprint*, arXiv:2406.00515.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. Association for Computational Linguistics.

Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix Yu, and Sanjiv Kumar. 2023. Large language models with controllable working memory. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1774–1793, Toronto, Canada. Association for Computational Linguistics.

Xiaonan Li and Xipeng Qiu. 2023. MoT: Memory-of-thought enables ChatGPT to self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6354–6374, Singapore. Association for Computational Linguistics.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2024. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. In *The Twelfth International Conference on Learning Representations*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 158–167. Association for Computational Linguistics.

Frederick Liu, Terry Huang, Shihang Lyu, Siamak Shakeri, Hongkun Yu, and Jing Li. 2021. Enct5: A framework for fine-tuning t5 as non-autoregressive models. *arXiv preprint arXiv:2110.08426*.

Lei Liu, Xiaoyan Yang, Yue Shen, Binbin Hu, Zhiqiang Zhang, Jinjie Gu, and Guannan Zhang. 2023a. Think-in-memory: Recalling and post-thinking enable llms with long-term memory. *Preprint*, arXiv:2311.08719.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023b. Pretrain, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Yiheng Liu, Hao He, Tianle Han, Xu Zhang, Mengyuan Liu, Jiaming Tian, Yutong Zhang, Jiaqi Wang, Xiaohui Gao, Tianyang Zhong, Yi Pan, Shaochen Xu, Zihao Wu, Zhengliang Liu, Xin Zhang, Shu Zhang, Xintao Hu, Tuo Zhang, Ning Qiang, Tianming Liu, and Bao Ge. 2024. Understanding llms: A comprehensive overview from training to inference. *Preprint*, arXiv:2401.02038.

Jianqiao Lu, Wanjun Zhong, Wenyong Huang, Yufei Wang, Qi Zhu, Fei Mi, Baojun Wang, Weichao Wang, Xingshan Zeng, Lifeng Shang, Xin Jiang, and Qun Liu. 2024. Self: Self-evolution with language feedback. *Preprint*, arXiv:2310.00533.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve GPT-3 after deployment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2833–2861, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jonson Manurung. 2019. Application of fifo algorithm (first in first out) to simulation queue. *Infokum*, 7(2, Juni):44–47.

Kai Mei, Zelong Li, Shuyuan Xu, Ruosong Ye, Yingqiang Ge, and Yongfeng Zhang. 2024. Aios: Llm agent operating system. *Preprint*, arXiv:2403.16971.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ali Modarressi, Abdullatif Köksal, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. 2024. Memllm: Finetuning llms to use an explicit read-write memory. *Preprint*, arXiv:2404.11672.

Jesse Mu, Xiang Li, and Noah Goodman. 2024. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.

Charles Packer, Vivian Fang, Shishir G Patil, Kevin Lin, Sarah Wooders, and Joseph E Gonzalez. 2023. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Robert Praas. 2023. Self-reflection on chain-of-thought reasoning in large language models.

Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2023. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*.

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

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy P. Lillicrap, Jean-Baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, Ioannis Antonoglou, Rohan Anil, Sebastian Borgeaud, Andrew M. Dai, Katie Millican, Ethan Dyer, Mia Glaese, Thibault Sottiaux, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, James Molloy, Jilin Chen, Michael Isard, Paul Barham, Tom Hennigan, Ross McIlroy, Melvin Johnson, Johan Schalkwyk, Eli Collins, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, Clemens Meyer, Gregory Thornton, Zhen Yang, Henryk Michalewski, Zaheer Abbas, Nathan Schucher, Ankesh Anand, Richard Ives, James Keeling, Karel Lenc, Salem Haykal, Siamak Shakeri, Pranav Shyam, Aakanksha Chowdhery, Roman Ring, Stephen Spencer, Eren Sezener, and et al.

2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *CoRR*, abs/2403.05530.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Carolyn K Rovee-Collier, Harlene Hayne, and Michael Colombo. 2001. *The development of implicit and explicit memory*. John Benjamins Publishing Company Amsterdam.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1743–1752. The Association for Computational Linguistics.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. *Preprint*, arXiv:2402.07927.

Yunfan Shao, Zhichao Geng, Yitao Liu, Junqi Dai, Hang Yan, Fei Yang, Zhe Li, Hujun Bao, and Xipeng Qiu. 2024. Cpt: a pre-trained unbalanced transformer for both chinese language understanding and generation. *SCIENCE CHINA Information Sciences*, 67(5):152102–.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Preprint*, arXiv:2303.11366.

Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Qiushi Sun, Zhirui Chen, Fangzhi Xu, Kanzhi Cheng, Chang Ma, Zhangyue Yin, Jianing Wang, Chengcheng Han, Renyu Zhu, Shuai Yuan, et al. 2024a. A survey of neural code intelligence: Paradigms, advances and beyond. *arXiv preprint arXiv:2403.14734*.

Qiushi Sun, Zhangyue Yin, Xiang Li, Zhiyong Wu, Xipeng Qiu, and Lingpeng Kong. 2023. Corex: Pushing the boundaries of complex reasoning through multi-model collaboration. *arXiv preprint arXiv:2310.00280*.

Tianxiang Sun, Xiaotian Zhang, Zhengfu He, Peng Li, Qinyuan Cheng, Xiangyang Liu, Hang Yan, Yunfan Shao, Qiong Tang, Shiduo Zhang, Xingjian Zhao, Ke Chen, Yining Zheng, Zhejian Zhou, Ruixiao Li, Jun Zhan, Yunhua Zhou, Linyang Li, Xiaogui Yang, Lingling Wu, Zhangyue Yin, Xuanjing Huang, Yu-Gang Jiang, and Xipeng Qiu. 2024b. Moss: An open conversational large language model. *Machine Intelligence Research*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

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

Bo Wang, Tianxiang Sun, Hang Yan, Siyin Wang, Qingyuan Cheng, and Xipeng Qiu. 2024a. In-memory learning: A declarative learning framework for large language models. *Preprint*, arXiv:2403.02757.

Jianing Wang, Qiushi Sun, Xiang Li, and Ming Gao. 2024b. Boosting language models reasoning with chain-of-knowledge prompting. *Preprint*, arXiv:2306.06427.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024c. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6).

Liang Wang, Nan Yang, and Furu Wei. 2024d. Learning to retrieve in-context examples for large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1752–1767, St. Julian's, Malta. Association for Computational Linguistics.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023a. Augmenting language models with long-term memory. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Xiaohua Wang, Yuliang Yan, Longtao Huang, Xiaoqing Zheng, and Xuan-Jing Huang. 2023b. Hallucination detection for generative large language models by bayesian sequential estimation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15361–15371.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023c. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian McAuley. 2024e. Memoryllm: Towards self-updatable large language models. *Preprint*, arXiv:2402.04624.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022c. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. *Preprint*, arXiv:1410.3916.

Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He. 2022. A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems*, 135:364–381.

Yang Wu, Yanyan Zhao, Zhongyang Li, Bing Qin, and Kai Xiong. 2023. Improving cross-task generalization with step-by-step instructions. *SCIENCE CHINA Information Sciences*, pages –.

Hongkang Yang, Zehao Lin, Wenjin Wang, Hao Wu, Zhiyu Li, Bo Tang, Wenqiang Wei, Jinbo Wang, Zeyun Tang, Shichao Song, Chenyang Xi, Yu Yu, Kai Chen, Feiyu Xiong, Linpeng Tang, and Weinan E. 2024. Memory^3: Language modeling with explicit memory. *Preprint*, arXiv:2407.01178.

Shunyu Yao. 2024. Language agents: From next-token prediction to digital automation.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240, Singapore. Association for Computational Linguistics.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. 2023. Do large language models know what they don't know? In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8653–8665, Toronto, Canada. Association for Computational Linguistics.

Yu Zhang, Peter Tino, Ales Leonardis, and Ke Tang. 2021. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2024. A survey on the memory mechanism of large language model based agents. *Preprint*, arXiv:2404.13501.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *Preprint*, arXiv:2303.18223.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory.

A Statistics and Details of Datasets

In our experiments, we selected 14 datasets across three different task categories. These tasks share the common requirement that the model must engage in reasoning and analysis before generating answers. Detailed statistics for each dataset, including the type of answers, the number of evaluation samples, the number of CoT prompting (Wei et al., 2022b) demonstrations used, and the corresponding licenses, are provided in Table 2.

B Implementation Details

Baseline Implementation. In our main experiments, we compare EM^2 against several baseline methods: ZS-CoT (Kojima et al., 2022), CoT (Wei et al., 2022b), and ComplexCoT (Fu et al., 2023). For ZS-CoT, the phrase "Let's think step by step" is appended to each question to activate the model's reasoning process, a method also adopted for EM^2 in Table 1. For CoT and ComplexCoT, we used the official prompts. The prompts used for CoT also serve as the memory initialization for EM^{2*},

Dataset	Task	Answer Format	# Ex.	# Eval.	License
GSM8K (Cobbe et al., 2021)	WMP	Number	8	1,319	MIT License
MultiArith (Roy and Roth, 2015)	WMP	Number	8	600	Unspecified
SingleEq (Koncel-Kedziorski et al., 2016)	WMP	Number	8	508	Unspecified
AddSub (Hosseini et al., 2014)	WMP	Number	8	395	Unspecified
SVAMP (Patel et al., 2021)	WMP	Number	8	1,000	MIT License
AQUA (Ling et al., 2017)	WMP	Multi-choice	4	254	Apache-2.0
MATH (Hendrycks et al., 2021)	WMP	Multi-choice	8	5,000	MIT license
StrategyQA (Geva et al., 2021)	Commonsense	T/F	6	2,290	MIT license
CommonsenseQA (Talmor et al., 2019)	Commonsense	Multi-choice	7	1,221	Unspecified
BoolQ (Clark et al., 2019)	Commonsense	T/F	4	3,270	CC BY-SA 3.0
ARC-c (Clark et al., 2018)	Commonsense	Multi-choice	4	299	CC BY-SA 4.0
Date Understanding (Suzgun et al., 2023)	Symbolic	Multi-choice	3	250	MIT license
Penguins in a Table (Suzgun et al., 2023)	Symbolic	Multi-choice	3	146	MIT license
Colored Objects (Suzgun et al., 2023)	Symbolic	Multi-choice	3	250	MIT license
Object Counting (Suzgun et al., 2023)	Symbolic	Multi-choice	3	250	MIT license

Table 2: Detailed statistics of the datasets utilized in our experiments. # EX. indicates the number of CoT prompting demonstrations used from each dataset. # EVAL. denotes the total number of evaluation samples in each dataset. The datasets are categorized by task type: WMP (Word Math Problem), Commonsense QA, and Symbolic Understanding, as discussed in Section 5.1.

with the number of prompts per dataset detailed in Table 2.

For multiple inference setting, we employ the Self-Consistency method (Wang et al., 2023c) to select the final answer. For MoT (Li and Qiu, 2023) and AutoCoT (Zhang et al., 2023), we replicated results on LLaMA-3 (Dubey et al., 2024) using the official implementation provided by the original authors.

Generation Setting. During our experiments, we obverse that different tasks and LLMs required specific temperature settings to achieve optimal performance. For the LLaMA-3-8B model, ZS-CoT perform better with greedy decoding, while CoT necessitated a higher temperature, typically around 0.5, for best results. For larger models, such as LLaMA-3-70B, setting the temperature to approximately 0.7 was found to be more suitable to foster superior outputs.

For multiple sampling settings, we established the number of samplings at five. We set the memory capacity at 20. To construct a representative validation set, we use the same number of clusters as in AutoCoT (Zhang et al., 2023). Specifically, we select ten samples from each cluster. Clustering ensures the diversity of selected samples while reducing the computational overhead for each update. Initially, when the number of samples is less than 50, we select all samples not already in memory to serve as the validation set. The clustering is performed using the KMeans algorithm with the number of clusters set to eight. We set the threshold ϵ in Eq 8 to 9. We utilize GitHub Copilot for assisting in the code writing process. Further details and ablation analysis can be found in Section C.

C Further Analysis

In this section, we delve into the impact of various hyperparameters on the performance of our algorithm. Additionally, we expand our analysis to include a broader range of clustering algorithms and embedding models to provide a comprehensive understanding of how these factors influence the effectiveness of our approach. All analyses are conducted using the LLaMA-3-8B (Dubey et al., 2024).

Memory Size. In Figure 11, we assess the impact of varying memory sizes on both performance and computation time, using datasets from three different tasks. The experimental results indicate that increasing memory size contributes to improved performance; however, the marginal gains decrease as the memory size continues to expand. Concurrently, there is a significant increase in computational overhead, as evidenced by the increase in processing time measured on a single RTX 4090. The results, displayed in the bar graph within the figure, clearly show that larger memory sizes substantially extend run times. Considering the costs associated with memory retrieval and updates, choosing an appropriate memory size is crucial. Therefore, we set an upper limit of 20 for memory size to balance performance and computational efficiency.

https://openai.com/index/new-embedding-models-and-api-updates

	GSM8K	MultiArith	SingleEq	AddSub	SVAMP	AQuA	Average
EM^2	82.63	97.77	92.71	86.32	83.91	45.27	81.43
Cluster Algorithm							
DBSCAN	83.47	96.50	93.50	85.82	83.45	44.09	81.13
Embedding Models							
Sentence Bert	81.65	94.67	91.73	84.81	82.62	46.85	80.38
Ada-002	82.78	94.33	92.32	88.86	83.70	45.66	81.27
Update Mechanism							
Random	76.42	93.00	83.85	84.81	79.25	40.16	76.25
FIFO	74.37	91.83	85.23	85.06	80.09	39.37	76.00

Table 3: Ablation analysis on six word math problem datasets. We evaluate the impact of different clustering algorithms, embedding models, and updating mechanisms on performance. "Ada-002" refers to the "`text-embedding-ada-002`" model.

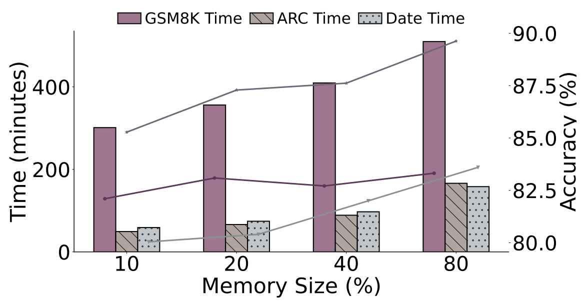


Figure 11: Impact of memory size on performance and running time. The bar graph represents running time, while the line graph indicates accuracy.

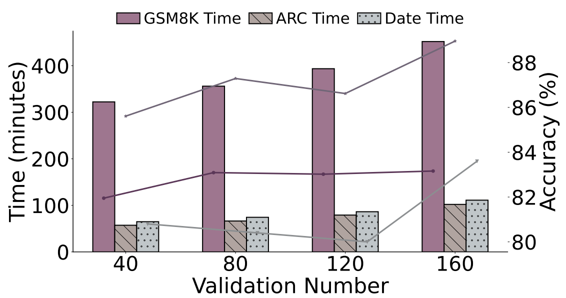


Figure 12: Impact of the number of validation set samples. The bar graph illustrates running time, while the line graph shows accuracy.

Validation Set Size. In Figure 12, we examine the effects of validation set size on both performance and computation time, employing the same evaluation metrics used for memory size assessment. Our analysis across representative datasets such as GSM8K, ARC, and Date Understanding shows that increasing the size of the validation set can lead to performance improvements. However, these improvements are not substantial; for instance, on the GSM8K dataset, increasing the number of validation samples beyond 80 does not yield significant performance gains. Similarly to the increase in memory size, a larger validation set also leads to longer run times, although not as dramatically. Considering the trade-offs between performance gains and computational costs, it is crucial to select an appropriate validation set size. Therefore, we set the upper limit for validation samples to ten times the number of classes to maintain a balance between effectiveness and efficiency.

Cluster Algorithm and Embedding Models. In Table 3, we assess the impact of different clustering algorithms and embedding models on model performance. Our experiments conducted across six math word problem datasets demonstrate that EM^2 is robust to the choice of clustering algorithm and embedding models. Specifically, when replacing the KMeans clustering algorithm with DBSCAN, using the default settings of DBSCAN, we observe no significant changes in performance across the datasets. Similarly, substituting `text-embedding-3-large` with Sentence-BERT (Reimers and Gurevych, 2019) or `text-embedding-ada-002` dose not result in any noticeable performance degradation across the datasets. Interestingly, `text-embedding-ada-002` even shows a slight average performance improvement over `text-embedding-3-large`. This phenomenon suggests that the choice of clustering algorithm and embedding models primarily influences the construction of the representative valida-

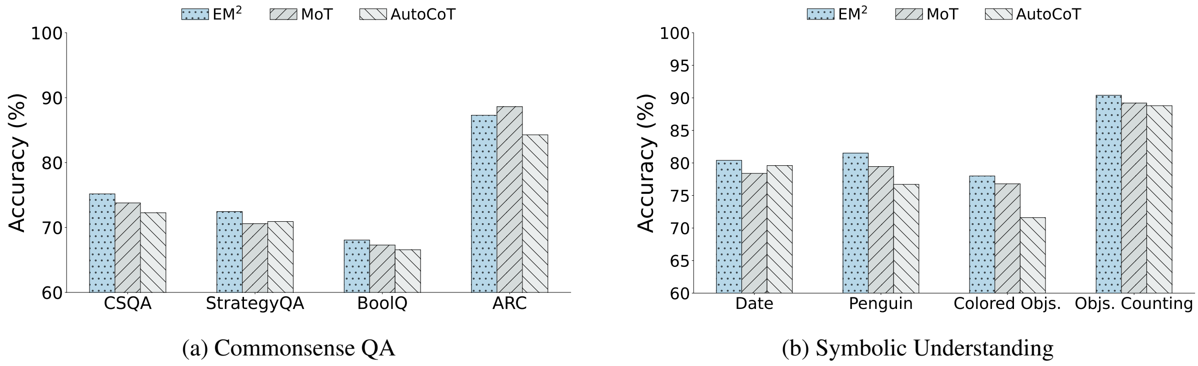


Figure 13: Comparison of the EM^2 method with Retrieval Memory on (a) commonsense question answering and (b) symbolic understanding tasks.

tion set and does not severely impact the memory updating mechanism of EM^2.

Analysis of Memory Updating Mechanism. In Section 5.4, we analyze the impact of altering the memory updating mechanism to Random and FIFO (First-In-First-Out) on the MATH dataset. The results presented in Table 3 demonstrate that similar significant performance declines occur on other math word problem datasets when employing Random and FIFO updating mechanisms. This underscores the importance of designing effective memory updating strategies.

Comparison of Memory Retrieval Method. In Figure 13, we extend our comparison of EM^2 with the Memory Retrieval Method to additional tasks. Maintaining the same experimental settings as in Section 5.4, we conducted experiments on Commonsense QA and Symbolic Understanding tasks. The results indicate that EM^2 demonstrates a clear advantage on the majority of the datasets, showing an average improvement of 2.82% over AutoCoT. This highlights the effectiveness of the dynamic memory updating strategy of EM^2.