Search-in-Context: Efficient Multi-Hop QA over Long Contexts via Monte Carlo Tree Search with Dynamic KV Retrieval

Jiabei Chen[1,2,3], **Guang Liu**[3], **Shizhu He**[1,2], **Kun Luo**[1,2,3],
Yao Xu[1,2,3], **Jun Zhao**[1,2], **Kang Liu**[1,2,4*]
[1] The Key Laboratory of Cognition and Decision Intelligence for Complex Systems,
Institute of Automation, Chinese Academy of Sciences
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences
[3] Beijing Academy of Artificial Intelligence [4] Shanghai Artificial Intelligence Laboratory
{chenjiabei2024,luokun2024}@ia.ac.cn, liuguang@baai.ac.cn,
{yao.xu, jzhao, shizhu.he, kliu}@nlpr.ia.ac.cn

Abstract

Recent advancements in large language models (LLMs) have demonstrated remarkable capabilities in complex reasoning tasks, such as math problem-solving and code generation. However, multi-hop question answering (MHQA) over long contexts, which demands both robust knowledge-intensive reasoning and efficient processing of lengthy documents, remains a significant challenge. Existing approaches often struggle to balance these requirements, either neglecting explicit reasoning or incurring expensive computational costs due to full-attention mechanisms over long contexts. To address this, we propose **Search-in-Context (SIC)**, a novel framework that integrates Monte Carlo Tree Search (MCTS) with dynamic key-value (KV) retrieval to enable iterative, context-aware reasoning. SIC dynamically retrieves critical KV pairs (e.g., 4K tokens) at each step, prioritizing relevant evidence while mitigating the "lost in the middle" problem. Furthermore, the paper introduces a Process-Reward Model (PRM) trained on auto-labeled data to guide the MCTS process with stepwise rewards, promoting high-quality reasoning trajectories without manual annotation. Experiments on three long-context MHQA benchmarks (HotpotQA, 2WikiMultihopQA, MuSiQue) and a counterfactual multi-hop dataset demonstrate SIC's superiority, achieving state-of-the-art performance while significantly reducing computational overhead.

1 Introduction

Recent advancements in large language models (LLMs) (Brown et al., 2020) have significantly improved their capability to tackle complex, reasoning-intensive tasks across diverse domains, including mathematical problem-solving (Jaech et al., 2024; Yang et al., 2024a), repository-level code generation and correction (Hui et al., 2024; Luo et al., 2024), and scientific reasoning (Ma et al., 2024). Such achievements highlight their growing capacity to handle sophisticated tasks that were previously thought to require human-level expertise.

Despite these advances, applying LLMs to multi-hop question answering (MHQA) over long contexts remains a significant challenge (Bai et al., 2023), as it requires models to simultaneously satisfy two key capabilities: **Strong knowledge-intensive reasoning capability:** The model must effectively integrate information across multiple reasoning hops, effectively synthesizing relevant information from intermediate subquestions to enable knowledge-driven inference (Mavi et al., 2022). **Robust long-context processing capability:** The model must efficiently handle extensive contexts (often exceeding 10K tokens) while filtering out irrelevant or distracting information (Fu et al., 2024b), ensuring the accurate identification and extraction of key information across length context necessary for answering the question.

Current approaches struggle to meet these dual requirements simultaneously. Many existing methods rely on long-context LLMs to directly answer MHQA tasks (Bai et al., 2023; Zhang et al., 2024b), underestimating the task's complexity and overlooking the importance of explicit test-time reasoning.

To bridge this gap, some works adopt chain-of-thought (CoT) prompting techniques (Li et al., 2024a; Trivedi et al., 2023; Wei et al., 2022). Unlike reasoning tasks in mathematics or science, where contexts are typically limited to a few hundred tokens, MHQA over long contexts requires models to generate reasoning chains based on inputs exceeding 10K tokens. In such scenarios, generating a reasoning chain necessitates computing full attention over an increasingly large Key-Value (KV) cache at each decoding step, leading to quadratic computational complexity growth (Fu et al., 2024a). Furthermore, models often struggle

* Corresponding Author

Findings of the Association for Computational Linguistics: ACL 2025, pages 26443–26455

with vast amounts of distractive information, a phenomenon commonly referred to as the "lost in the middle" problem (Liu et al., 2024b).

In this paper, we propose a novel framework that integrates Monte Carlo Tree Search (MCTS) with dynamic KV retrieval to empower LLMs with iterative, context-aware exploration over long contexts. Inspired by test-time scaling algorithms (Snell et al., 2024; Qi et al., 2024), our method formulates the retrieval and reasoning as a search tree process, where each node represents a potential reasoning step guided by contextually retrieved evidence. At each iteration, the model selectively utilizes portions of critical KVs (e.g., 4K token budgets) based on a specialized KV retriever, rather than relying on computationally expensive full-attention over the entire KV cache. This process effectively retrieves and prioritizes the most relevant KVs that contribute to uncovering critical information for subsequent reasoning hops, mitigating the "lost in the middle" problem in long context and improving reasoning efficiency. Additionally, we incorporate a Process-Reward Model (PRM) into the MCTS process to guide the model's reasoning. This PRM provides step-by-step rewards to encourage the model to follow high-quality reasoning paths. Importantly, the PRM can be trained using automatically labeled data without requiring manual annotation, ensuring scalability and reducing human intervention.

The main contributions of this paper can be summarized as follows:

- We propose **Search-in-Context (SIC)**, an innovative framework utilizing a modified Monte Carlo Tree Search (MCTS) algorithm to enhance multi-hop QA in long contexts, guided by a trained Process Reward Model which utilizes an automated annotation process.
- We integrate **dynamic key-value (KV) retrieval** into the MCTS process, enabling the model to selectively focus on the most relevant portions of the context (e.g., 4K token budgets) at each step.
- Extensive experiments on three long-context multi-hop reasoning datasets (e.g., HotpotQA , 2WikiMultihopQA , MuSiQue) and a counterfactual multi-hop dataset adapted for long contexts demonstrate the superiority of SIC in long-context multi-hop QA tasks.

2 Related Work

Multi-hop Reasoning. Multi-hop question answering (MHQA) (Yang et al., 2018; Trivedi et al., 2022; Ho et al., 2020) is a challenging task that requires models to reason over multiple pieces of information, often scattered across different parts of a document or multiple documents, to arrive at the correct answer (Mavi et al., 2022). Conventional approaches (Zhang et al., 2024a; Zhu et al., 2021) adopt the selector-reader framework, where a selector module retrieves relevant documents or passages, and a reader module extracts or generates the final answer based on the retrieved context. Recent developments, however, have marked a significant shift toward a paradigm centered on long-context language models (LMs) (Li et al., 2024a; Trivedi et al., 2023). This emerging approach eliminates the need for a separate selector module, instead relying on a long-context LM to process the entire set of retrieved documents and fulfill the role of the reader.

Long-context Language Modeling. Scaling LLM to process long texts poses significant challenges due to the quadratic computational complexity of attention mechanisms (An et al., 2024). To mitigate the computational and memory constraints, recent research has explored various KV compression techniques (Sun et al., 2024; Yang et al., 2024b). These methods selectively retain subsets of KVs based on predefined reduction strategies, often compressing them to a fixed budget (Tang et al., 2024; Li et al., 2024b; Huang et al., 2024; Shi et al., 2024). For instance, H2O (Zhang et al., 2023b) employs a policy that discards KVs during generation according to a scoring function derived from cumulative attention. InfLLM (Xiao et al., 2024) partitions KVs into fixed-size chunks and retains the top-k most salient chunks based on attention score patterns.

3 Preliminary

In this section, we provide a formal and descriptive definition of our task.

Problem Formulation. Multi-hop Question Answering (MHQA) over long context is a complex reasoning task requiring iterative reasoning across multiple, often disparate sources of information over documents to deduce an answer. We formulate this task as follows: Given the input which contains the query q and contexts $\mathcal{C} = \{c_1, c_2, ..., c_n\}$ where c_i represents a single and independent document,

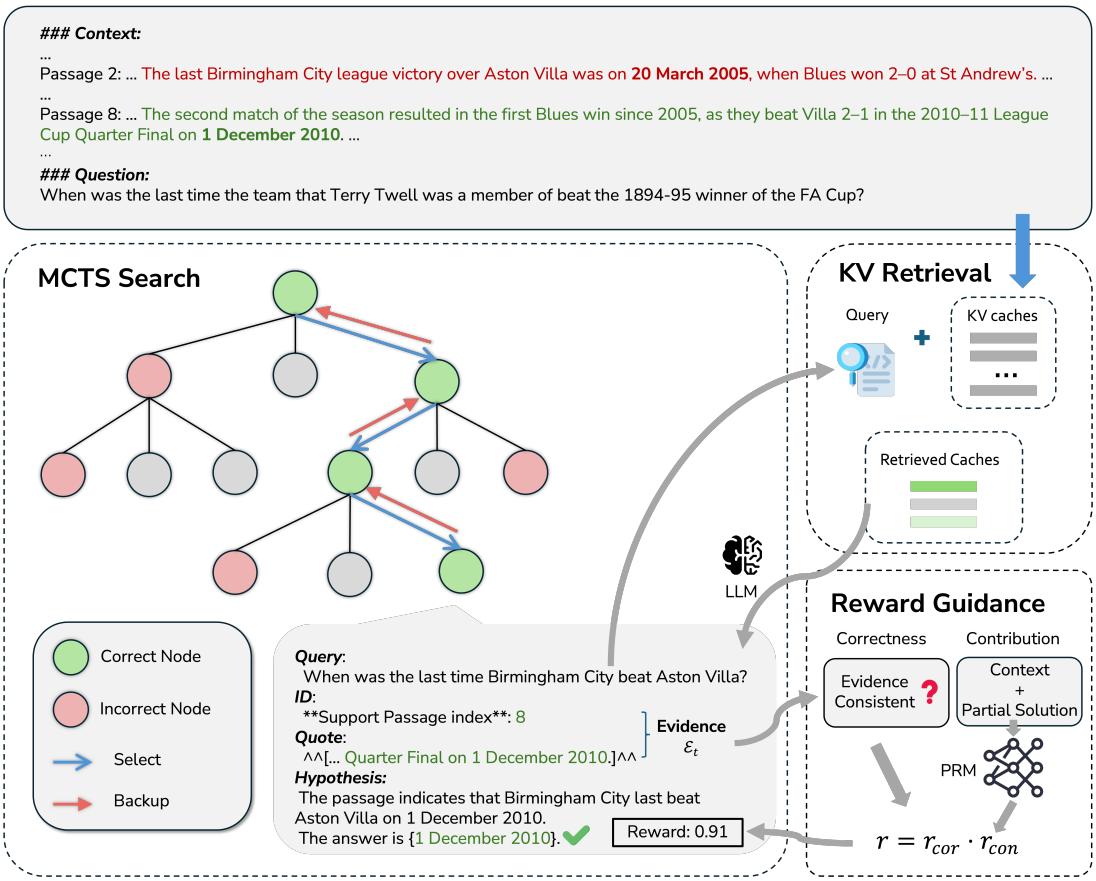

Figure 1: The overall framework of our SIC approach.

the task is aimed at predicting an answer $a \in \mathcal{A}_q$ that satisfies:

$$\exists \mathcal{P}_q \subseteq \mathcal{C}, |\mathcal{P}_q| > 1 \wedge \mathcal{P}_q \models (a \text{ answers } q) \quad (1)$$

where $\mathcal{P}_q = \{\hat{c}_1, \hat{c}_2, ..., \hat{c}_k\}$ represents the minimal sufficient evidence set to deduce the answer.

4 Methods

4.1 Decomposed Reasoning with Structured Thought Chains

To enable systematic and controllable multi-step reasoning, we formalize the reasoning process as a sequence of structured steps comprising three components: **query refinement**, **evidence grounding**, and **hypothesis generation**.

Query Refinement. Query refinement is the process of breaking down the original question q into sub-questions q_t that guide the model's reasoning at each step. As shown in Figure 1, for each step, SIC first generates a sub-query q_t to decompose the original question into more focused sub-problems. Alternatively, based on the evidence retrieved in previous steps and the intermediate reasoning outcomes, the model may either refine the sub-query further to explore unresolved aspects or conclude the reasoning process by synthesizing the final answer from the accumulated evidence and logical deductions.

Evidence Grounding. In long-context multi-hop QA, contexts often contain redundant or irrelevant information, making it crucial to dynamically retrieve only the most pertinent evidence for each sub-question q_t. Motivated by the methodology of reasoning with attribution(Li et al., 2024a; Gao et al., 2023; Trivedi et al., 2023), evidence retrieval component consists of two parts: supported passage index id and relevant snippet quotation $quote$, which are formatted into structured evidence $\mathcal{E}_t = (id_t, quote_t)$. The former identifies the most relevant documents from the contexts, while the latter extracts specific evidence that directly addresses q_t. This structured representation ensures that each reasoning step is grounded in verifiable and relevant document snippets, critical for maintaining faithfulness and reducing hallucination.

Hypothesis Generation. After obtaining the refined sub-query q_t and the corresponding evidence $\mathcal{E}_t$, the hypothesis generation component formu-

lates intermediate conclusions h_t to bridge the gap between raw evidence and final answers. This step is critical for transforming raw evidence into insights, ensuring that each reasoning step is both logically coherent and grounded in facts.

This design allows the Monte Carlo Tree Search (MCTS) which is illustrated in Section 4.2 to treat each step as a discrete node in the search space, facilitating guided exploration and pruning of invalid paths.

4.2 Guided Exploration via Process-Aware MCTS

Multi-hop reasoning over long-context documents requires systematically planning sub-queries to break down complex questions into steps (Radhakrishnan et al., 2023). By question decomposition, it is more effective than standard chain-of-thought prompting, as it is easier for LLMs to generate one step rather than a whole solution in a single-turn inference. To address this, we adopt Monte Carlo Tree Search (MCTS) (Coulom, 2007; Kocsis and Szepesvári, 2006; Hao et al., 2024), a powerful planning algorithm that balances exploration and exploitation to navigate the combinatorial search space of multi-step reasoning.

During the search process, the algorithm begins at root node s_0, which unfolds in three iterative stages: *selection*, *expand and evaluation*, *backup*:

- **Selection** Starting from the root node, the algorithm traverses the tree by selecting actions (sub-queries q_t) that maximize the criterion according to $q_t = \arg\max_q(Q(s_t, q) + U(s_t, q))$ where $Q(s_t, q)$ illustrates the cumulative reward and $U(s_t, q)$ is calculated by a variant of PUCT algorithm (Rosin, 2011):

$$U(s_t, q) = \alpha \cdot \pi_{\theta_k}(q|s_t) \frac{\sqrt{\Sigma_b N(s_t, b)}}{1 + N(s_t, q)} \quad (2)$$

  where α balances the exploration and exploitation, $N(s, q)$ is the visit count of selecting sub-query q at node s. And the prior $\pi(q|s_t)$ is defined as the exponential of mean log-probability of all tokens in sub-query q.

- **Expand and Evaluation** When a leaf node s_t is reached, the tree is expanded by generating new candidate sub-queries q_{t+1} with sampling. For each candidate q_{t+1}, we then use the LLM to predict the next state through structured thought decoding as described in Section 4.1. Thus each node s_{t+1} can be represented as:

$$s_{t+1} = \langle q_{t+1}, \mathcal{E}_{t+1}, h_{t+1} \rangle \quad (3)$$

  After obtaining the next node state, the reward function evaluates s_t, computing a reward score $r(s_t, q_t)$ based on correctness and contribution to the final correct answer. The reward design will be discussed in further detail later.

- **Backup** Once the terminal state is reached (e.g., the final answer is validated or a computational budget is exhausted), the backup phase propagates rewards backward along the reasoning path, updating the visit count N, the cumulative reward Q and the state value V:

$$Q(s_t, q_t) \leftarrow r(s_t, q_t) + \gamma V(s_{t+1}) \quad (4)$$

$$V(s_t) \leftarrow \frac{\sum_q N(s_{t+1}) Q(s_t, q)}{\sum_q N(s_{t+1})} \quad (5)$$

$$N(s_t) \leftarrow N(s_t) + 1 \quad (6)$$

  where γ is the discount for future state values.

Reward Design. The reward function is designed to balance **factual correctness** (grounding in retrieved evidence) and **reasoning contribution** (progress toward resolving the question). It combines two components:

- **Factual Correctness** For evidence $\mathcal{E}_t$ in each node, factual correctness evaluates r_{cor} whether the quotation in the evidence exists and aligns with the supported evidence indices. This evolves a two-step verification process: validate that every document index cited in $\mathcal{E}_t$ is present in the context $\mathcal{C}$ and ensure the referenced content of the corresponding snippet exists in the supported index passage. Alignment is measured using fuzzy match:

$$r_{cor} = \mathbb{I}(id_t \in \mathcal{C}_{id}) \cdot \mathbb{I}(FM(quote_t, c_{id_t}) \geq \tau) \quad (7)$$

  where $\mathcal{C}_{id}$ denotes the document index set, τ represents the threshold for fuzzy matching.

- **Reasoning Contribution** The contribution of a reasoning step s_t is defined as its potential to reduce uncertainty toward the correct answer. Traditional outcome-based reward models (ORMs) evaluate solutions holistically

(Yao et al., 2023), lacking granular feedback on intermediate steps. To address this, motivated by Math-Shepherd (Wang et al., 2024) which demonstrates that automated process supervision—leveraging Monte Carlo Tree Search (MCTS) principles, we extend this insight to design our contribution metric and the process-based reward model (PRM) in step-level. The reasoning contribution is scored with the original question q, provided contexts $\mathcal{C}$ and partial solutions $s_{1:t}$:

$$r_{con} = PRM([\mathcal{C}; q; s_{1:t}]) \in [0, 1] \quad (8)$$

For this PRM training, the step-wise labels are automatically constructed via this process: for each step s_t, a completer will generate K subsequent reasoning process from this step:$\{s_{t+1,j}, ..., s_{D_j,j}, a_j\}_{j=1}^{K}$ where a_j and D_j are the answer and the number of reasoning steps for j-th solution. Then we use the frequency of reaching the correct answer a^* as the contribution label for the step s_t:

$$y_{s_t} = \frac{\Sigma_{j=1}^{K} \mathbb{I}(a_j = a^*)}{K} \quad (9)$$

After obtaining the label for each step, we can train the PRM using cross-entropy loss.

Therefore, the final reward score for each step s_t is calculated as $r(s_t, q_t) = r_{cor} \cdot r_{con}$.

4.3 Context-Aware Evidence Retrieval with Dynamic KV Cache

In long multi-hop QA scenarios, the reasoning process is divided into two stages: prefilling of long context and generation of multiple reasoning chains. The prefilling stage is performed only once, after which the KVs are cached to speed up generation. During the generation of reasoning chains, the cached KVs are reused multiple times. However, this leads to high computational overhead, as each decoding step requires full attention computation over the entire lengthy cached KVs.

To address this challenge, we propose a reasoning-oriented, trainable KV retriever to conduct KV cache compression during the generation of multiple reasoning chains, only using portions of critical KVs for decoding. Existing KV compression approaches (Xu et al., 2024; Zhang et al., 2023b) typically rely on heuristic estimations of full attention based on cumulative attention scores,

Input:

Read the following text and find key information for the question.

The history of netball can be traced to the early development of basketball. A year after basketball was invented in 1891...

...

(/Positive) The Australian and New Zealand national teams have traditionally dominated the international game, although England and Jamaica are becoming increasingly competitive against their Antipodean counterparts. **(Positive/)**

...

Netball includes having 4 defenders including the centre and 4 attackers also including the centre.

Now, find key information for the question.

Question: Which teams have traditionally dominated the international game in New Zealand?

Figure 2: Synthetic data for KV retriever training.

which yield suboptimal performance in multi-hop reasoning tasks.

Compression Process. KV compression is performed to select relevant KVs for a given subquestion during the reasoning chain generation. To facilitate retrieval, we first partition the LLM's input context $X = \{x_i\}_{i=1}^{l}$ into contiguous chunks:

$$\{x_1, ..., x_l\} \xrightarrow{\text{partition}} \{X_1, ..., X_m\}, X_i = \{x_j^i\}_{j=1}^{w} \quad (10)$$

where w is the chunk size (128 in practice). A special landmark token ($\langle$LMK$\rangle$) is appended to each chunk, forming $X_i^{'} = \{x_1^i, ..., x_w^i, \langle lmk \rangle^i\}$. These landmark tokens serve as representations of their respective chunks and are used for KV retrieval.

After pre-filling of the long context, the KVs are cached and reused for generations of multiple reasoning chains. During the generation of new tokens, KV retrieval is conducted for each intermediate subquestion within the reasoning chain:

$$C : \{X_1', ..., X_k'\} = p^{kv}(X' : \{X_1', ..., X_m'\} | q) \quad (11)$$

where C represents the compressed KVs used for attention computation, replacing the expensive full-attention mechanism. The query q corresponds to the current subquestion.

KV Retriever. We propose a reasoning-oriented, trainable KV retriever to retrieve critical KVs for each reasoning step. It introduces a set of trainable parameters to the self-attention module of LLM.

During the self-attention computation, the hidden states of normal tokens (n) and landmark tokens (b) are sliced out and projected into query, key, and value vectors respectively:

$$\begin{aligned} \boldsymbol{Q}^n &= \boldsymbol{W}_Q^n \boldsymbol{H}^n, \quad \boldsymbol{K}^n = \boldsymbol{W}_K^n \boldsymbol{H}^n, \quad \boldsymbol{V}^n = \boldsymbol{W}_V^n \boldsymbol{H}^n, \\ \boldsymbol{Q}^b &= \boldsymbol{W}_Q^b \boldsymbol{H}^b, \quad \boldsymbol{K}^b = \boldsymbol{W}_K^b \boldsymbol{H}^b, \quad \boldsymbol{V}^b = \boldsymbol{W}_V^b \boldsymbol{H}^b \end{aligned} \quad (12)$$

where $\boldsymbol{W}_*^{\boldsymbol{n}}$ are the LLM's original projection matrices and $\boldsymbol{W}_*^{\boldsymbol{b}}$ are the newly introduced matrices designed specifically to handle landmark tokens.

KV importance estimation employs similarity between the query vector of target chunk's landmark token and the key vectors of past chunks' landmark tokens:

$$p^{kv}(X') = \text{top-}k\left\{\langle \boldsymbol{q}_m^{\text{lmk}}, \boldsymbol{k}_j^{\text{lmk}}\rangle\right\}_{j=1}^{m-1} \quad (13)$$

where $\langle *, * \rangle$ denotes the dot product operation.

Training. Training the KV retriever poses a challenge due to the lack of appropriately labeled long-context data for retrieval supervision.

As shown in Figure 2, we synthesize 10K pairwise long-context data (up to 8K tokens) using text from Wikipedia (Lehmann et al., 2015) to train the KV retriever. This dataset contains coherent contexts, enabling the retriever to effectively learn to locate target evidence relevant to a given query. For each long text, we randomly select a span containing several consecutive sentences and use ChatGPT to generate a question based on this span. This process provides a retrieval supervision signal, guiding the KV retriever in identifying the target evidence corresponding to the query.

Given that the evidence for a query (chunk m) is located on chunk i, we train the KV retriever using the following contrastive learning objective:

$$L_1 = -\log \frac{\exp(\langle \boldsymbol{q}_m^{lmk}, \boldsymbol{k}_i^{lmk}\rangle)}{\sum_{j=1}^{m-1}\exp(\langle \boldsymbol{q}_m^{lmk}, \boldsymbol{k}_j^{lmk}\rangle)} \quad (14)$$

where $\boldsymbol{q}_*^{lmk}$ and $\boldsymbol{k}_*^{lmk}$ are the query and key vectors of landmark tokens of corresponding chunks in the self-attention module.

KV compression is conducted at each decoder layer, allowing for a broader global contextual view while enabling the decoder to focus on key information within the lengthy text. This process effectively reduces noise and distractions, enhancing evidence retrieval performance.

5 Experiments

5.1 Experiments Setup

Datasets and Evaluation Metrics. We conduct our experiments on multi-hop long-context QA, *i.e.*, **HotpotQA** (Yang et al., 2018), **2WikiMultihopQA** (Ho et al., 2020) and **MusiQue** (Trivedi et al., 2022) from LongBench (Bai et al., 2023). Additionally, we also incorporate CofCA (Wu et al., 2024), a counterfactual MHQA benchmark. To adapt to the demands of this task, we randomly sample 100 examples each from the 2-hop, 3-hop, and 4-hop subsets of CofCA. We then extend their context lengths to 10K by adding irrelevant documents, thereby constructing a new variant called **CofCA-10K**. This dataset helps reduce the risk of data contamination, thereby providing a more robust evaluation of the model's multi-hop reasoning capabilities. Table 1 presents the statistics about these datasets. Following previous works, we adopt the F1 score as our evaluation metric.

Dataset		# Total Samples	Max Tokens	Avg. Tokens
HotpotQA		200	16323	12780
MusiQue		200	16320	15543
2WikiMultihopQA		200	16336	7097
CofCA	2-hop	100	11176	10853
	3-hop	100	11239	10851
	4-hop	100	11095	10867

Table 1: Statistics of our test datasets. The number of tokens is calculated by the tokenizer of Llama-3.1-8B-Instruct.

Baselines. The baselines we compare can be divided into two categories: (1) *single-turn prompting*, including standard prompting (IO) (Brown et al., 2020) which generates answers directly and Chain-of-Thought (CoT) prompting (Wei et al., 2022). (2) *multi-turn tree search approaches*. We select Tree-of-Thoughts (ToT) (Yao et al., 2023) and rStar (Qi et al., 2024) as baselines, using Breadth-First-Search (BFS) and MCTS, respectively.

Implementation Details. We use two LLMs as the backbone in our experiments: Mistral-7B-Instruct-v0.2 (Jiang et al., 2023) and Llama-3.1-8B-Instruct(Grattafiori et al., 2024). Specifically, we first train the KV retriever independently using the synthetic dataset in Section 4.3, enabling it to perform reasoning-oriented KV compression for each reasoning hop. To ensure that the model adheres to structured node outputs illustrated in Section 4.1 and adapts to the sparse KV contexts retrieved by the retriever, we fine-tune our backbones for 1 epoch using 1K samples from a mixed training set derived from the original training sets of HotpotQA , MusiQue, and 2WikiMultiHopQA. For each question in this training set, we utilize DeepSeek-V3 (Liu et al., 2024a) to sample 5 structured reasoning trajectories. Thus, the whole generated training set both for the policy model and PRM contains 50K solutions. These trajectories provide clear, step-by-step intermediate reasoning paths, ensuring that the model learns to produce outputs that align with the desired structured format. The generation prompt

Model	Methods	MusiQue	HotpotQA	2Wiki	CofCA-10K			Avg.
					2 hop	3 hop	4 hop	
Mistral-7B-Instruct-v0.2	IO	19.12	44.44	26.93	48.60	33.79	38.28	35.19
	CoT	26.57	40.78	39.45	39.28	40.9	35.14	37.02
	SIC*	32.54	53.47	59.75	43.45	58.14	54.03	50.23
	ToT (N=16)	27.48	38.89	36.91	41.98	37.98	35.36	36.43
	rStar (N=16)	37.90	50.89	51.91	51.60	46.46	39.96	46.45
	SIC (N=16)	**51.80**	61.54	70.66	53.93	**67.19**	**60.45**	60.93
	+Retrieval (4K, N=4)	51.46	**63.41**	**72.87**	**54.83**	65.42	59.54	**61.26**
Llama3.1-8B-Instruct	IO	32.09	57.27	46.08	63.97	46.65	45.99	48.66
	CoT	39.60	54.31	59.79	48.95	56.8	55.19	52.44
	SIC*	47.86	61.77	65.98	62.46	54.84	52.35	57.54
	ToT (N=16)	38.24	55.2	64.15	52.94	55.03	53.8	53.22
	rStar (N=16)	47.20	62.02	72.90	57.30	57.55	47.67	57.44
	SIC (N=16)	**59.87**	**67.11**	**77.75**	66.66	62.65	59.71	65.63
	+Retrieval (4K, N=4)	57.10	66.38	76.83	**66.94**	**65.31**	**63.90**	**66.07**
Llama-3.1-70B-Instruct	IO	40.75	64.39	62.68	64.00	53.53	47.55	55.48

Table 2: Performance (%) comparison of different baselines on four datasets. SIC* represents the backbone model after fine-tuning and using CoT prompting with greedy decoding. N denotes the iteration number of the tree search algorithm. Under the setting of using dynamic KV retrieval, our context window is set to **4K**, while other baselines rely on the full-attention mechanism with a context window of 32K. The boldface indicates the best result.

can be found in Appendix A.

The training set created in this process is also utilized for training the PRM. For each single step, we use Llama-3.1-8B-Instruct as the completer, with a sampling number of K=16. Additionally, we select Llama-3.1-8B as the base model to train the reward model using the entire training set mentioned above, which serves as the verifier in our algorithm. For all tree search methods, we set the depth $d = 8$ and the width $w = 5$.

5.2 Main Results

Table 2 shows the F1 score of our framework and all baselines on four MHQA datasets. From the table, we can find that the SIC method outperforms other baselines across all four MHQA datasets, demonstrating its superior ability to handle multi-hop question answering tasks in long-context scenarios.

For single-turn baselines, we observe that our SIC* method fine-tuned with just 1K data samples for one epoch achieves significant performance improvements on nearly all datasets. For multi-turn tree search approaches, our framework, which integrates the MCTS algorithm with dynamic KV retrieval, achieves significant improvements across all datasets. Notably, SIC using SLMs even exceeds the performance of Llama-3.1-70B-Instruct, highlighting its potential to enhance the capabilities of smaller models in long-context multi-hop reasoning tasks.

Moreover, with dynamic KV retrieval, the model operates within a 4K context window, yet it still outperforms the full-attention approach that processes the entire context. This highlights the efficiency and effectiveness of our method in prioritizing relevant information, reducing computational overhead, and achieving superior results in multi-hop reasoning tasks over long contexts.

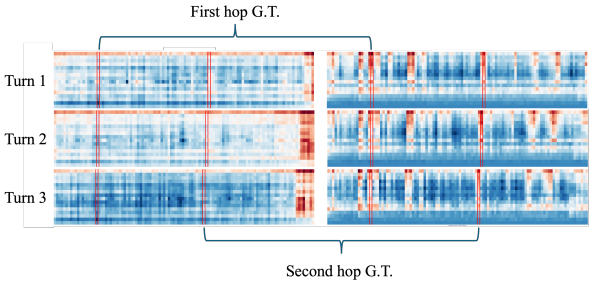

Figure 3: Attention score map for a CofCA-10K 3hop sample. Left: from original full attention. Right: score from our dynamic KV retriever. Red squares indicate key information for the multi-hop question. For each turn, the x-axis represents the sequence position (up to 10K tokens), and the y-axis represents each decoder layer.

Models	Verifiers	MusiQue	HotpotQA	2WikiMQA	CofCA	Avg.
Mistral-7B-Instruct-v0.2	SC@maj16	44.88	58.97	65.81	56.08	56.42
	BoN16(Ours)	49.26	60.05	69.53	56.75	58.90
Llama-3.1-8B-Instruct	SC@maj16	50.51	62.23	73.70	58.99	61.36
	BoN16(Ours)	56.38	62.85	73.94	60.99	63.54

Table 3: Performance of different LLMs on four MHQA datasets using different verification strategies. SC@maj denotes the self-consistency technique, which samples multiple reasoning paths and selects the most consistent answer by majority voting. BoN means best-of-N sampling using a verifier.

5.3 Analysis

Results for CofCA CofCA is a counterfactual dataset that emphasizes the model's reasoning ability rather than its memorization capacity due to data contamination. In Table 2, SIC achieves the best performance on the dataset without any specific training on it. This demonstrates the robustness and generalizability of our approach, as it relies on the inherent reasoning capabilities of the framework rather than dataset-specific fine-tuning. However, IO prompting on CofCA-2hop outperforms both CoT and SIC*. This is likely due to the simplicity of the dataset, where explicit reasoning does not significantly improve the performance. This outcome aligns with the findings in (Li et al., 2024a).

Dynamic KV retriever As shown in Figure 3, while full-attention mechanism fails to capture the essential KVs, our KV retriever identifies key KVs effectively with each turn. Notably, the key information for the second hop is located in the middle of the context, a region typically challenging for models due to the "lost in the middle" problem. The ability of our KV retriever to successfully identify this information demonstrates that dynamic KV retrieval can somewhat alleviate this issue. More comparison of dynamic KV retriever with other baseline retrievers is shown in Appendix B.1.

5.4 Ablation Study

Effectiveness under Different Rollouts. For tree search algorithms, the number of rollouts (iterations) directly impacts both the quality of candidate solutions and the computational cost. Increasing the number of rollouts allows the algorithm to explore a larger portion of the search space, potentially uncovering higher-quality reasoning paths. However, this comes at the expense of increased inference time and resource consumption. To investigate how the number of rollouts affects our SIC's performance, we evaluate the performance of the HotpotQA under different rollouts, as illustrated in Figure 4.

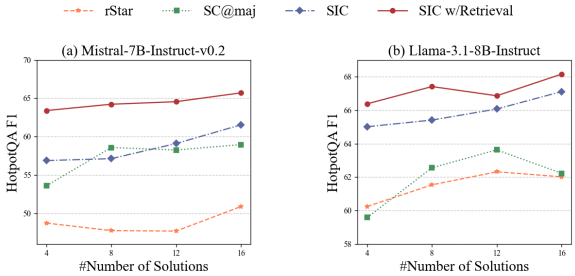

Figure 4: Performance comparison on the HotpotQA dataset under the different number of solutions.

It can be found that SIC benefits from rollouts, regardless of whether dynamic KV retrieval is used, which applies to both Llama and Mistral models. Another observation we can conclude is that self-consistency (SC) tends to saturate and even decline on Llama-3.1-8B-Instruct. The reason is that for chain-of-thought prompting, hallucinations in intermediate steps can occur and compound, leading to entirely incorrect conclusions (Zhang et al., 2023a; Wan et al., 2024).

Effectiveness of the Verifier. To evaluate the effectiveness of the verifier, we compare our verifier, which uses best-of-N (BoN) sampling, with the self-consistency (SC) (Wang et al., 2022) approach. In the BoN method, the verifier selects the best-performing trajectory from multiple sampled paths based on the last step contribution scores, while SC aggregates results by majority voting after sampling diverse reasoning paths. As shown in Table 3, our trained verifier outperforms self-consistency across all datasets with both models. Notably, on the MusiQue dataset, the performance improvements are significant, with Llama and Mistral achieving gains of 5.87 and 4.38, respectively.

Moreover, the training data used for the verifier was generated using Llama-3.1-8B-Instruct, yet it

still demonstrates strong generalization and provides effective guidance for Mistral. This indicates that the verifier's learned scoring mechanism is robust and transferable, even when applied to different backbone models. Moreover, the comparison between PRM and self-reward signal within our framework can be found in Appendix B.2.

6 Conclusion

In this paper, we propose SIC, a novel framework integrating Monte Carlo Tree Search (MCTS) with dynamic key-value (KV) retrieval to address the dual challenges of efficiency and reasoning in large language models (LLMs) for multi-hop question answering (MHQA) over long contexts. By modeling the reasoning process as a search tree and incorporating dynamic KV retrieval, SIC iteratively focuses on critical contextual segments (e.g., 4K tokens per step), mitigating the "lost in the middle" problem while reducing computational complexity. Our comprehensive experiments across two models and four datasets validate the superiority of SIC in long-context multi-hop QA tasks.

7 Limitation

While our framework demonstrates promising results on long-context multi-hop QA tasks, several limitations remain for future work. The iterative nature of the Monte Carlo Tree Search (MCTS) process, though effective for refining reasoning trajectories, incurs increased inference latency and computational cost compared to single-pass methods, limiting its practicality for real-time applications. Additionally, our framework has primarily been tested on datasets with contexts around 10-20K tokens, leaving its applicability to significantly longer texts (e.g., 100K+ tokens or even book-length documents) an open question for future work.

8 Ethical consideration

Our work is built upon open-source LLMs. Consequently, it inherits similar ethical and social risks as those associated with the base LLM. These risks include but are not limited to biases in language generation, potential reinforcement of societal stereotypes, and the generation of harmful or toxic content. Despite efforts in LLM pre-training and fine-tuning to mitigate such risks, no model is entirely free from unintended biases, as these often stem from the underlying training data.

Acknowledgments

This work was supported by the National Key R&D Program of China (No. 2022ZD0160503) and Beijing Natural Science Foundation (L243006) and the National Natural Science Foundation of China (No.62376270).

References

Chenxin An, Fei Huang, Jun Zhang, Shansan Gong, Xipeng Qiu, Chang Zhou, and Lingpeng Kong. 2024. Training-free long-context scaling of large language models. *arXiv preprint arXiv:2402.17463*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

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.

Rémi Coulom. 2007. Efficient selectivity and backup operators in monte-carlo tree search. In *Computers and Games*, pages 72–83, Berlin, Heidelberg. Springer Berlin Heidelberg.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. 2024a. Data engineering for scaling language models to 128k context. *arXiv preprint arXiv:2402.10171*.

Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. 2024b. Not all heads matter: A head-level kv cache compression method with integrated retrieval and reasoning. *arXiv preprint arXiv:2410.19258*.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6465–6488, Singapore. Association for Computational Linguistics.

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

Shibo Hao, Yi Gu, Haotian Luo, Tianyang Liu, Xiyan Shao, Xinyuan Wang, Shuhua Xie, Haodi Ma, Adithya Samavedhi, Qiyue Gao, Zhen Wang, and Zhiting Hu. 2024. Llm reasoners: New evaluation, library, and analysis of step-by-step reasoning with large language models. *arXiv preprint arXiv:2404.05221*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173, Singapore. Association for Computational Linguistics.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Yuxiang Huang, Binhang Yuan, Xu Han, Chaojun Xiao, and Zhiyuan Liu. 2024. Locret: Enhancing eviction in long-context llm inference with trained retaining heads. *arXiv preprint arXiv:2410.01805*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. 2024. Qwen2.5-coder technical report. *arXiv preprint arXiv: 2409.12186*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.

Yanyang Li, Shuo Liang, Michael Lyu, and Liwei Wang. 2024a. Making long-context language models better multi-hop reasoners. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2462–2475, Bangkok, Thailand. Association for Computational Linguistics.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024b. Snapkv: Llm knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024b. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Qinyu Luo, Yining Ye, Shihao Liang, Zhong Zhang, Yujia Qin, Yaxi Lu, Yesai Wu, Xin Cong, Yankai Lin, Yingli Zhang, et al. 2024. Repoagent: An llm-powered open-source framework for repository-level code documentation generation. *arXiv preprint arXiv:2402.16667*.

Yubo Ma, Zhibin Gou, Junheng Hao, Ruochen Xu, Shuohang Wang, Liangming Pan, Yujiu Yang, Yixin Cao, Aixin Sun, Hany Awadalla, et al. 2024. Sciagent: Tool-augmented language models for scientific reasoning. *arXiv preprint arXiv:2402.11451*.

Vaibhav Mavi, Anubhav Jangra, and A. Jatowt. 2022. Multi-hop question answering. *Foundations and Trends in Information Retrieval*.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. *arXiv preprint arXiv: 2408.06195*.

Ansh Radhakrishnan, Karina Nguyen, Anna Chen, Carol Chen, Carson Denison, Danny Hernandez, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Sam McCandlish, Sheer El Showk, Tamera Lanham, Tim Maxwell, Venkatesa Chandrasekaran, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. 2023. Question decomposition improves the faithfulness of model-generated reasoning. *arXiv preprint arXiv: 2307.11768*.

Christopher D Rosin. 2011. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61(3):203–230.

Zhenmei Shi, Yifei Ming, Xuan-Phi Nguyen, Yingyu Liang, and Shafiq Joty. 2024. Discovering the gems in early layers: Accelerating long-context llms with 1000x input token reduction. *arXiv preprint arXiv:2409.17422*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv: 2408.03314*.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. 2024. Shadowkv: Kv cache in shadows for high-throughput long-context llm inference. *arXiv preprint arXiv:2410.21465*.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. 2024. Cot rerailer: Enhancing the reliability of large language models in complex reasoning tasks through error detection and correction. *arXiv preprint arXiv: 2408.13940*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, D. Schuurmans, Quoc Le, Ed H. Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *International Conference on Learning Representations*.

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

Jian Wu, Linyi Yang, Zhen Wang, Manabu Okumura, and Yue Zhang. 2024. Cofca: A step-wise counterfactual multi-hop qa benchmark. *arXiv preprint arXiv: 2402.11924*.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, Song Han, and Maosong Sun. 2024. Infllm: Unveiling the intrinsic capacity of llms for understanding extremely long sequences with training-free memory. *arXiv preprint arXiv:2402.04617*.

Yuhui Xu, Zhanming Jie, Hanze Dong, Lei Wang, Xudong Lu, Aojun Zhou, Amrita Saha, Caiming Xiong, and Doyen Sahoo. 2024. Think: Thinner key cache by query-driven pruning. *arXiv preprint arXiv:2407.21018*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024a. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv: 2409.12122*.

Dongjie Yang, XiaoDong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. 2024b. Pyramidinfer: Pyramid kv cache compression for high-throughput llm inference. *arXiv preprint arXiv:2405.12532*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Jiahao Zhang, Haiyang Zhang, Dongmei Zhang, Liu Yong, and Shen Huang. 2024a. End-to-end beam retrieval for multi-hop question answering. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1718–1731, Mexico City, Mexico. Association for Computational Linguistics.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. 2023a. How language model hallucinations can snowball. *arXiv preprint arXiv: 2305.13534*.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. 2024b. ∞Bench: Extending long context evaluation beyond 100K tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277, Bangkok, Thailand. Association for Computational Linguistics.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. 2023b. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and reading: A comprehensive survey on open-domain question answering. *arXiv preprint arXiv: 2101.00774*.

A Prompt List

The prompt used for creating our training dataset and generating reasoning steps is shown in Figure 5.

B Result Analysis

B.1 Comparison of KV Retriever with Baseline Retrievers on Multi-Hop QA Datasets

In this section, we present comparison of our dynamic KV retriever against two baseline retrievers: BM25, a sparse retriever using term matching, and BGE, a dense retriever based on embeddings. Evaluations are conducted on HotpotQA and MuSiQue datasets with a $4K$-token budget per reasoning step, using Llama-3.1-8B-Instruct as the backbone model. Performance is measured by F1 score.

The results, presented in Table 4, demonstrate that our KV retriever consistently outperforms both BM25 and BGE on HotpotQA and MuSiQue. Unlike traditional Retrieval-Augmented Generation (RAG) methods, such as BM25 (a sparse retriever) and BGE (a dense retriever), our KV retriever is reasoning-oriented and trainable. It dynamically selects critical key-value pairs (e.g., $4K$ tokens) tailored to each reasoning step within our SIC framework. This adaptability enables the KV retriever to focus on the most relevant evidence for multi-hop question answering over long contexts. The superior performance over static methods like BM25 and embedding-based approaches like BGE highlights the KV retriever's ability to enhance evidence retrieval, making it particularly effective for complex, multi-step reasoning tasks.

Table 4: F1 scores of retrievers on HotpotQA and MuSiQue.

Retriever	HotpotQA	MuSiQue
BM25	61.60	49.88
BGE	62.41	51.53
KV Retriever (ours)	**66.38**	**57.10**

B.2 Comparison between PRM and Self-reward signal

To assess the contribution of the PRM in our framework, we conducted an ablation study comparing its performance against a simpler self-rewarding signal, following the RAP (Hao et al., 2023) framework. Experiments were performed on the HotpotQA and MuSiQue datasets using the Llama-3.1-8B-Instruct model as the backbone,with a $4K$-token retrieval budget, and $N = 4$ reasoning steps. Performance is evaluated using the F1 score.

Table 5 shows that incorporating PRM yields significant performance improvements over the self-rewarding signal. These results highlight PRM's critical role in guiding high-quality reasoning trajectories, surpassing the capabilities of a simpler reward signal in multi-hop question-answering tasks.

Table 5: F1 scores comparing PRM and self-rewarding signal on HotpotQA and MuSiQue.

Methods	HotpotQA	MuSiQue
SIC w/ Retrieval		
w/ Self-Reward	62.53	51.08
w/ PRM	**66.38**	**57.10**

Prompt for Generation

Answer the multi-hop question based on the given context.
Context
{context}
Instruction
Answer the multi-hop question based on the given text. Break the original question into smaller, logical sub-questions, ensuring each step addresses a single, specific aspect of the question and is fully supported by the provided context. Follow the detailed reasoning format below:
Step 1: A sub-question that directly reflects a critical part of the question, ensuring no words or nuances are missed.
Supported passage index:<Number>
Provide the original text or context using "^^[]^^" to highlight the most key text first. Explain the reasoning process in detail, ensuring it is logical, clear and natural. Finally, give the subanswer in this step in a natural way and enclose the exact answer in "{{}}" in the sentence. Note that the subanswer is the answer to the brief title and should be a short phrase.
...
Step n: Now we can answer the question.
Please provide a detailed reasoning step to connect all subanswers logically to derive the final answer. The final answer should be a concise, non-sentence answer and enclosed in "**{{}}**" like "The final answer is **{{XXX}}**".
Question
{question}
Answer
Let's think step by step.

Figure 5: Prompt used in SIC framework.