Tree of Agents: Improving Long-Context Capabilities of Large Language Models through Multi-Perspective Reasoning

Song Yu[1], Xiaofei Xu[2], Ke Deng[3†], Li Li[1†], Lin Tian[4]
[1]School of Computer and Information Science, Southwest University, Chongqing, China
[2]School of Information Technology, Murdoch University, Perth, Australia
[3]School of Computing Technologies, RMIT University, Melbourne, Australia
[4]University of Technology Sydney, Sydney, Australia

ke.deng@rmit.edu.au lily@swu.edu.cn

Abstract

Large language models (LLMs) face persistent challenges when handling long-context tasks, most notably the *"lost in the middle"* issue, where information located in the middle of a long input tends to be underutilized. Some existing methods that reduce input have the risk of discarding key information, while others that extend context windows often lead to attention dispersion. To address these limitations, we propose *Tree of Agents (TOA)*, a multi-agent reasoning framework that segments the input into chunks processed by independent agents. Each agent generates its local cognition, then agents dynamically exchange information for collaborative reasoning along tree-structured paths. TOA enables agents to probe different reasoning orders for multi-perspective understanding, effectively mitigating position bias and reducing hallucinations. To improve processing efficiency, we incorporate prefix-hash caching and adaptive pruning strategies, achieving significant performance improvements with comparable API overhead. Experiments show that TOA, powered by compact LLaMA3.1-8B, significantly outperforms multiple baselines and demonstrates comparable performance to the latest and much larger commercial models, such as Gemini1.5-pro, on various long-context tasks. Code is available at https://github.com/Aireduce952/Tree-of-Agents.

1 Introduction

The capabilities of Large Language Models (LLMs) have seen rapid improvements in recent years. The advent of models such as OpenAI-o3 (El-Kishky et al., 2025) and DeepSeek-R1 (Guo et al., 2025) marks further progression in model inference capacity. Although LLMs perform well in many scenarios, they still fall short in tasks involving long contexts, such as financial report comprehension (Reddy et al., 2024), novel quizzes (Wang et al., 2025; Xu et al., 2024b) and legal contract analysis (Shen et al., 2022; Shu et al., 2024).

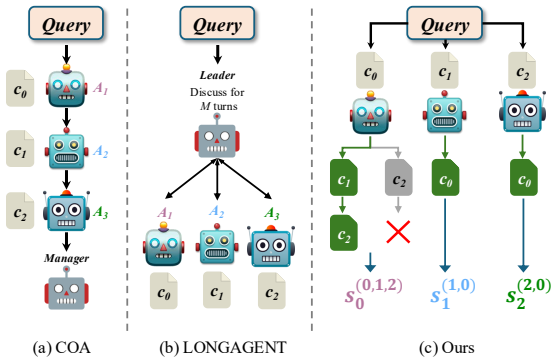

Figure 1: The core difference between TOA and other multi-agent reasoning methods. (a) COA processes chunks sequentially, with a final manager agent making the decision. (b) LONGAGENT uses a leader agent to coordinate multi-turn discussions with others. (c) TOA probes multi-paths in a tree structure to prompt multi-perspective reasoning.

There are three primary challenges in long-context tasks: (1) Position biases processing further impair performance, often referred to as the "lost in the middle" issue (Liu et al., 2024b). (2) As input length increases, the presence of extraneous or redundant information grows, potentially diluting the model's focus and degrading output quality (Shi et al., 2023). (3) The computational demands for both training and inference escalate dramatically with longer sequences (Gao et al., 2024).

To address these issues, existing approaches can be categorized into three main directions, as shown in Table 1. The first direction focuses on optimizing the model itself, either by enhancing long dependency capture through improvements in attention mechanisms (Chen et al., 2024b; Yen et al., 2024), training strategies (Zhang et al., 2024a; Chen et al., 2024a), or position encoding (Zhu et al., 2024; Jin et al., 2024). The second direction aids the model in solving long contexts and reducing memory usage

†Corresponding authors.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 4574–4592
November 4-9, 2025

Category	Example Work	PnP	No Train	Focus	Cost	Inter.
Model Modification	Position Encoding (Zhu et al., 2024)	✘	✘	✘	High	Low
	Attention Mechanism (Chen et al., 2024b)	✘	✘	✘	High	Low
	Training Strategies (Zhang et al., 2024a)	✘	✘	✘	High	Low
Input Reduction	RAG (Zhao et al., 2024b)	✘	✘	✔	Medium	Medium
	Prompt Compression (Jiang et al., 2024)	✘	✔	✔	Medium	Low
Multi-Agent Reasoning	LONGAGENT (Xiao et al., 2024)	✘	✘	✔	High	High
	COA (Zhang et al., 2024b)	✔	✔	✔	High	High
	Tree of Agents (Ours)	✔	✔	✔	High	High

Table 1: Comparison of various existing methods in long context tasks. Plug-and-play(PnP): whether the method can be applied directly without additional adaptation. Focus: ability to identify and prioritize critical information in long contexts. Inter.: interpretability of each method.

by enhancing key information density or reducing input length through retrieval augmented generation (Zhao et al., 2024b; Xu et al., 2024a) or prompt compression (Jiang et al., 2024). The third direction addresses long text processing by employing multi-agent reasoning (Zhang et al., 2024b; Zhao et al., 2024a) to break down the input into manageable chunks, which are subsequently integrated to arrive at the final result.

Nonetheless, processing long contexts efficiently remains a challenging problem (Hsieh et al., 2024). Extending the context window for a model may hurt its performance on short texts (Ding et al., 2024). Retrieval-based method relies on the quality of external retrieval tools and may introduce noise (Li et al., 2024). Prompt compression may lose key information, especially when information is unevenly distributed in long texts. Such methods are more suitable for QA or information extraction tasks and have limited performance for tasks that require global understanding (Jiang et al., 2023). Multi-agent reasoning requires a well-designed agent communication mechanism to get the right results. As illustrated in Figure 1, COA lets agents process text chunks sequentially and use a manager to integrate the results (Zhang et al., 2024b), and LONGAGENT uses a leader agent to coordinate multi-turn discussions with others (Xiao et al., 2024). However, these approaches generally do not support multi-perspective understanding of the document. This study proposes a novel multi-perspective reasoning framework — **Tree of Agents** (TOA), inspired by cognitive science theories and empirical findings on problem-solving. As shown in Figure 1, TOA splits a long text into chunks and employs multiple agents to explore different orders of chunks along multi-paths of a tree structure. This design draws from Kahneman's dual-process theory (Kahneman, 2011), which advocates for a multi-perspective approach, and Newell and Simon's theory of problem-solving as an iterative exploration process (NEWELL, 1959). Unlike sequential processing, which often struggles in tasks requiring strategic exploration or early decisions, our method disrupts the natural reading order to enable agents to re-assess and refine their understanding through cross-validation.

Agents in TOA collaborate by exchanging local cognition and applying the mechanism for consensus formation. The multi-perspective reasoning significantly improves performance on various tasks but increases computational cost. To mitigate this issue, we introduce two efficiency-enhancing strategies: 1) A prefix-hash-based cache mechanism that reduces redundant cognition generations; 2) An adaptive pruning strategy that terminates useless reasoning paths early.

We conduct extensive experiments with eight baselines on two long-context reasoning datasets and one benchmark, demonstrating the effectiveness of our approach. The main contributions are:

1) We propose TOA, the first tree-structured multi-agent reasoning framework that addresses long-context modeling via multi-perspective reasoning.

2) We introduce two optimization strategies that reduce the computational cost while maintaining the advantage of the proposed multi-perspective reasoning.

3) This study empirically validates that TOA consistently improves long-context reasoning for various tasks. More interestingly, our experiments show that the different reading order of a long text can lead to different perceptions, and thus different answers to queries.

2 Related Work

This section summarizes key directions in long-context modeling for LLMs.

2.1 Specialized Long-Context Models

Training specialized models capable of handling long-contexts is a straightforward approach to improving capabilities. Notable examples include Claude3 (Anthropic, 2024) and Gemini-1.5pro (Team et al., 2024), which support 200K and 2M context windows, respectively. While these models can process long texts more effectively, they require expensive computational resources and may reduce their ability to understand short texts (Liu et al., 2024b).

2.2 Model Modification

To reduce the computational cost of training specialized models, several methods based on positional encoding have been proposed to enhance long dependency capture. Such methods include constraining the covariance of Q/K vectors (Zhu et al., 2024) or dynamically adjusting the RoPE angle (Lin et al., 2024). Additionally, SelfExtend expands the context window during the inference phase with only minor modifications to the model, without the need for fine-tuning (Jin et al., 2024). While effective in reducing training costs, these approaches often struggle with high complexity and computational limitations in practice.

2.3 RAG and Input Reduction

LongRAG combines global and local retrieval to enhance the accuracy of long-context question answering tasks (Zhao et al., 2024b). The applicability of RAG with long-context models for downstream tasks has been analyzed by (Xu et al., 2024a). These methods require additional information, which is not always available. Input reduction is another research direction that avoids modifying the model structure. In LongLLMLingua, the author compresses redundant information in the prompts to improve the density of key information (Jiang et al., 2024). However, input reduction has the risk of discarding key information.

2.4 Hybrid Approaches

Hybrid methods aim to dynamically select between retrieval-based and long-context models based on task complexity. Self-Route, for instance, adjusts the model's approach depending on the task at hand (Li et al., 2024). InfLLM stores long context externally and dynamically correlates related fragments during inference (Xiao et al., 2024). These methods can be flexible and efficient, but the complexity of dynamically choosing between methods may introduce additional overhead.

2.5 Multi-agent Reasoning

Recently, approaches based on multi-agent reasoning have gained attention as a promising solution to handle long contexts. HOMER reduces memory usage through chunking and token reduction (Song et al., 2024), while COA allows agents to process text chunks sequentially and communicate to integrate results (Zhang et al., 2024b). LONGAGENT coordinates agents through a leader agent to answer queries over 128K documents (Zhao et al., 2024a). COA and LONGAGENT avoid token reduction like HOMER. However, COA's unidirectional messaging and lack of dynamic interactions can lead to broken inference chains. LONGAGENT's performance depends on the leader's ability to decompose tasks, and fixed chunk sizes may disrupt paragraph integrity. In contrast, our TOA enables agents to probe different reasoning orders of chunks, effectively promoting multi-perspective understanding to mitigate position bias and reduce hallucinations.

3 Methodology

We first formulate the problem that TOA aims to solve. Given a query q and a document $\mathcal{D} = \{t_1, t_2, ..., t_M\}$ with M tokens, where M is much longer than the context window length of the LLM. Our goal is to build a system that can respond to the query correctly according to the document. TOA follows a three-phase reasoning process, as shown in Figure 2.

3.1 Phase 1: Chunk Perception

Due to the excessive context length, we first break down the document $\mathcal{D}$ into N short text chunks $(c_0, \cdots, c_{N-1})$. For the ith chunk c_i, it consists of tokens from position $i \cdot \frac{M}{N}$ to $(i+1) \cdot \frac{M}{N} - 1$, where $\frac{M}{N}$ is set to be below the agent's maximum context window. By adjusting the number of agents, TOA can handle varying input lengths, where one agent is for one chunk. For simplicity, the agent for the ith chunk is named agent $\mathcal{A}_i$.

In phase 1, $\mathcal{A}_i$ generates the initial cognitive state based on the ith chunk. The input of $\mathcal{A}_i$ is (q, c_i, p), where q represents the user's query,

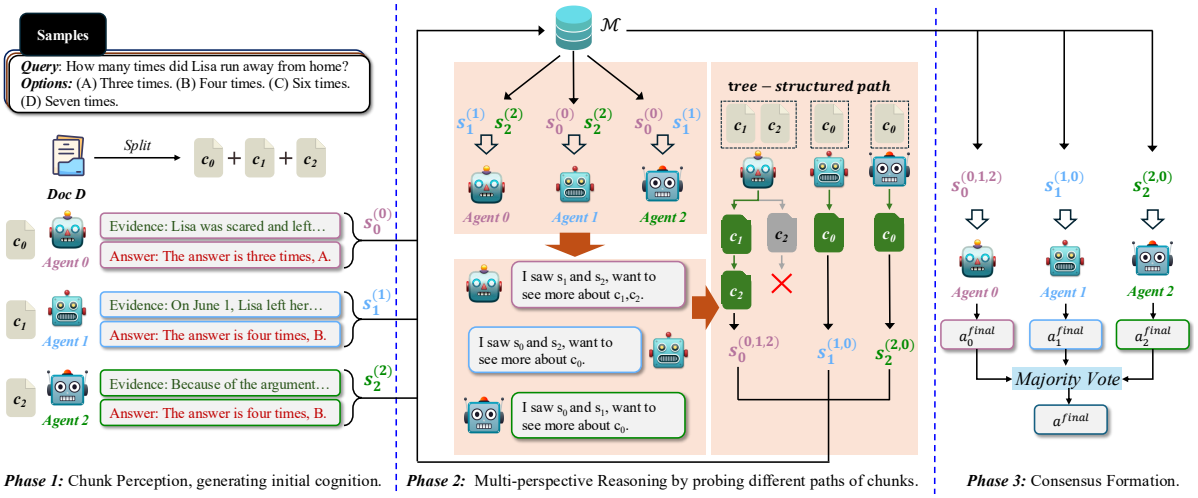

Figure 2: An overview of TOA. In Phase 1, the document is split into chunks, with each agent processing a chunk and providing cognition, which are stored in $\mathcal{M}$. In Phase 2, agents exchange cognition, express interest in reading additional chunks, and probe additional chunks in different orders. In Phase 3, each agent generates a local answer, and the final answer is determined by majority voting.

c_i represents the ith chunk, and p is the prompt. $\mathcal{A}_i$ outputs $\langle e_i, a_i \rangle$, where e_i is the evidence from chunk c_i that support to answer query q, and a_i is the answer returned by $\mathcal{A}_i$. We can express $\mathcal{A}_i$ as a function:

$$\langle e_i, a_i \rangle = \mathcal{A}_i(q, c_i, p), \forall i \in [0, N-1].$$

The initial cognitive state of agent $\mathcal{A}_i$ is denoted as $s_i^{(i)} = \langle e_i^{(i)}, a_i^{(i)} \rangle$ and stored in a buffer $\mathcal{M}_i$. In Figure 2, $\mathcal{M}$ denotes the buffers for all agents. To answer the question "How many times did Lisa run away from home?", each agent relies solely on its chunk. However, such multi-hop questions require a comprehensive understanding of the entire document. Lacking a global view, the answer of each agent is typically biased.

3.2 Phase 2: Multi-Perspective Understanding

In phase 2, each agent reads the initial cognitive states of other agents stored in $\mathcal{M}$. As shown in Figure 2 as an example, $\mathcal{A}_0$ reads $s_1^{(1)}$ and $s_2^{(2)}$, the initial cognitive states of $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. $\mathcal{A}_1$ and $\mathcal{A}_2$ do the same. The initial cognitive state of $\mathcal{A}_i$ can be viewed as a summary of the ith chunk by $\mathcal{A}_i$. By reading the initial cognitive states of other agents, an agent, already having an understanding of its chunk, can decide which other chunks can help it answer the query. For $\mathcal{A}_i$, the identities of these additional chunks are stored in $\mathcal{G}_i$. In Figure 2, $\mathcal{A}_0$ wants to read additional chunk c_1 and c_2, $\mathcal{A}_1$ and $\mathcal{A}_2$ wants to see c_0, respectively.

Then, TOA probes different paths following a tree structure for a comprehensive multi-perspective understanding of the document $\mathcal{D}$. For agent $\mathcal{A}_i$, the additional chunks to read are indexed in $\mathcal{G}_i$. Suppose $\mathcal{G}_i = \{j_0, j_1, ..., j_{k-1}\}$. We define Perm_i as all different paths visiting the chunks in $\mathcal{G}_i$. For example, if $\mathcal{G}_i = \{0, 1, 2\}$, then $\text{Perm}_i = \{(0,1,2), (0,2,1), (1,0,2), (1,2,0), (2,1,0), (2,0,1)\}$. If connected by a root, these paths are the paths form the root to leaves in a tree structure. Each path represents a unique way to read these chunks in $\mathcal{G}_i$. The purpose of probing all possible paths is to explore different cognitive orders, helping to avoid biases caused by the fixed reading order and ensuring a thorough understanding. If one path misses a key clue, other paths may fill in the gap. This redundancy mechanism greatly enhances robustness for complex tasks. *We illustrate an example in Appendix C to illustrate that the different reading orders lead to different answers by the same agent.*

Suppose the current cognitive state of $\mathcal{A}_i$ is $s_i^{(i)}$. After reading a chunk c_j indexed in $\mathcal{G}_i$, the cognitive state of $\mathcal{A}_i$ is updated as:

$$s_i^{(i,j)} \leftarrow \langle e_i^{(i,j)}, a_i^{(i,j)} \rangle = \mathcal{A}_i(s_i^{(i)}, c_j).$$

As shown in Figure 2, $\mathcal{A}_0$ reads chunk c_1 and c_2 so that the cognitive state of $\mathcal{A}_0$ is updated from $s_0^{(0)}$ to $s_0^{(0,1,2)}$. Similarly, the cognitive state of $\mathcal{A}_1$ and $\mathcal{A}_2$ are updated from $s_1^{(1)}$ to $s_1^{(1,0)}$ and from $s_2^{(2)}$ to $s_2^{(2,0)}$, respectively.

While each agent $\mathcal{A}_i$ to have a multi-perspective understanding of the document to answer queries correctly, the number of paths grows with the number of chunks in $\mathcal{G}_i$. To mitigate such a combinatorial explosion, we propose two optimization strategies.

State Caching in Tree Structure

As discussed above, all paths related to an agent form a tree structure, i.e., a path of chunks corresponds to a path from the root to a leaf in a tree structure. Given two paths of chunks, for example (0,1,2,4) and (0,1,2,3), they share a chunk sequence (0,1,2) and then fork to chunk 4 and chunk 3. Once the cognitive state for the chunk sequence (0,1,2) has been generated for the path (0,1,2,4), there is no need to generate it again for the path (0,1,2,3).

For this purpose, we store and reuse previously generated intermediate states by $\mathcal{A}_i$ in $\mathcal{M}_i$ with prefix hashing. Recall that c_i is the initial chunk of $\mathcal{A}_i$ in phase 1. The cache $\mathcal{M}_i$ is initialized as follows:

$$\mathcal{M}_i(\langle c_i \rangle) \leftarrow s_i^{(i)}.$$

For the jth path of $\mathcal{A}_i$ from Perm_i, $p_{ij} = \langle c_i, c_{j_0}, c_{j_1}, ..., c_{j_{k-1}} \rangle$, generating the cognitive state for p_{ij} starts by searching $\mathcal{M}_i$ for p_{ij}. If it exists, we retrieve the cognitive state directly; otherwise, we search $\mathcal{M}_i$ for p_{ij}^{-1}, i.e., p_{ij} after dropping the last chunk, if exists, we retrieve $s_i^{(i,j_0,\cdots,j_{k-2})}$ from $\mathcal{M}_i$ and use it to generate $s_i^{(i,j_0,\cdots,j_{k-1})}$ as follows:

$$s_i^{(i,j_0,\cdots,j_{k-1})} = \mathcal{A}_i(s_i^{(i,j_0,\cdots,j_{k-2})}, c_{j_{k-1}}).$$

The newly generated $s_i^{(i,j_0,\cdots,j_{k-1})}$ is stored in $\mathcal{M}_i$ and indexed by p_{ij}. This strategy uses a prefix sharing reuse mechanism to convert repeated calculations into lookup operations.

Adaptive Pruning

Information related to answering a query can be sparsely distributed in a long context. Adaptive pruning dynamically evaluates the value of segmented information and immediately terminates invalid paths. Specifically, if the chunk currently read along a path is deemed useless, the remaining portion of the path is immediately pruned. The interpretation of this pruning strategy is that the order of the chunks in the pruned path makes the information irrational or unenlightened to help answer the question. In this way, the pruning strategy effectively reduces useless path traversal and improves computational efficiency. In Figure 2, the red cross under c_2 for $\mathcal{A}_0$ illustrates a path pruning. In practice, all cognitive states are still cached normally before the path pruning occurs.

3.3 Phase 3: Consensus Formation

The consensus formation phase synthesizes distributed reasoning results through a two-tier hierarchical voting mechanism.

Intra-Agent Aggregation

For $\mathcal{A}_i$ with multiple paths, the answers at the end of each path can be different. In this situation, we prioritize the longest chunk sequence from $\mathcal{M}_i$:

$$\sigma_i^* = \underset{\sigma \in \mathcal{M}_i}{\text{argmax}} |\sigma|,$$

where σ is a chunk sequence in the cache $\mathcal{M}_i$. We believe that the longer sequences imply a broader context integration, reducing local bias (Koh et al., 2022). Each agent then generates its final answer by reprocessing the query with its optimal context:

$$a_i^{\text{final}} = \mathcal{A}_i\left(q, \mathcal{M}_i(\sigma_i^*), p\right),$$

where $\mathcal{M}_i(\sigma_i^*)$ retrieves the cached state for σ_i^*.

Note that an agent may return **none** after reading a path of chunks. It implies that the agent does not find sufficient information to answer the query.

Cross-Agent Majority Voting

To resolve agent disagreements, we aggregate all candidates via majority voting:

$$a^{\text{final}} = \text{MajorityVote}(\{a_i^{\text{final}}\}_{i=0}^{N-1}).$$

The detailed pseudocode of TOA is provided in Appendix E.

4 Experiment Setup

In this section, we describe the preliminary preparations for our experiments (see Appendix A and B for complete setup for datasets and baselines).

4.1 Evaluation Datasets

Experiments are conducted on two long-context reasoning datasets and one benchmark.

DetectiveQA (Xu et al., 2024b): DetectiveQA is a bilingual dataset with an average question length of 100K, containing a series of detective novel questions and answers.

NovelQA (Wang et al., 2025): NovelQA is a dataset with the average input length over 200K for testing the long-text ability of LLMs. It comprises the texts of 89 novels and 2305 question-answer pairs on the details of these novels.
Needle-in-a-Haystack (Kamradt, 2023): *"Needle-in-a-Haystack"* is a widely adopted benchmark for assessing LLMs' ability to process and retrieve critical information from long documents. The method involves inserting one or more short, specific text segments (the needles) into a much longer, semantically unrelated document (the haystack). Adjusting the position of the needle allows us to evaluate the model's ability to extract key information from various locations in the input. The needle's depth percentage refers to its insertion point within the haystack. By evaluating the model's performance at different needle positions, we gain insights into its ability to retrieve information throughout long-context documents.

4.2 Evaluation Metrics

For QA tasks, choosing one from four optional answers, we evaluate performance using two metrics, **accuracy** and **none-rate**. *accuracy* refers to the percentage of correct answers. *none* means that the model indicates it cannot retrieve relevant information to answer the question, and *none-rate* is the percentage of *none* in the total test samples, which is an effective reflection of the model's robustness and reliability in open-domain QA tasks.

For *Needle-in-a-Haystack* tasks, we use GPT-4o (version 2024-05-13) to score (from 1-10) the answers given by our TOA and baselines. A higher score means that the LLM finds more relevant information.

4.3 Baselines

LONGAGENT (Zhao et al., 2024a). An approach that uses multi-agent collaboration where the leader breaks down the problem and hands it over to agents for multiple rounds of discussion.
COA (Zhang et al., 2024b). CoA consists of multiple worker agents that sequentially communicate to handle different segmented chunks of the text, followed by a manager agent who synthesizes these contributions into a coherent final output.
LongLLMLingua (Jiang et al., 2024). A prompt compression method to enhance the perception of LLM for critical information.
LongRAG (Zhao et al., 2024b). A Retrieval Augmented Generation approach to Enhance LLM's understanding of complex contextual knowledge.
Gemini 1.5-pro (Team et al., 2024). The Gemini 1.5-pro released by Google supports up to 2M context window lengths.
GPT-4o (Hurst et al., 2024). The GPT-4o model from OpenAI offers a context window of 128K.
Sequential. The original document is split into multiple chunks, and a single agent reads them sequentially to directly output the final answer.
Vote. Split the original document into document chunks, then use multiple agents to read the content separately and directly determine the final answer through majority voting.

Note that LongRAG is excluded from the *Needle-in-a-Haystack* test due to the preprocessing overhead brought by building the vector database. LONGAGENT is excluded from the DeepSeek-V3 based experiments because fine-tuning the model is infeasible.

4.4 LLMs

We select LLaMA3.1-8B-instruct (Grattafiori et al., 2024) and DeepSeek-V3-Chat (Liu et al., 2024a) as the base models for TOA and all baselines. The former is deployed locally, while the latter is accessed via API calls.

5 Results and discussion

5.1 Overall Performance

Through multi-agent collaboration, TOA may alleviate the lost in the middle problem.

Figures 3 and 4 show that TOA achieves average scores of 9.38 and 7.77 on the single and multi needle tasks, respectively. Notably, its performance remains stable in the middle paragraph range (40%, 70%), unlike baselines such as Sequential and Vote, which exhibit significant drops. In the more demanding multi-needle setting, TOA consistently extracts key information, demonstrating superior global alignment across dispersed content.

TOA achieves state-of-the-art performance with lower none-rate.

Table 2 compares the performance on two long-context reasoning datasets: DetectiveQA and NovelQA, using both LLaMA3.1-8B and DeepSeek-V3 as base models. We observed that COA performed poorly, and we speculate that the problem is that its unidirectional chain suffers from information decay and error amplification issues, which may lead to incomplete or wrong answers.

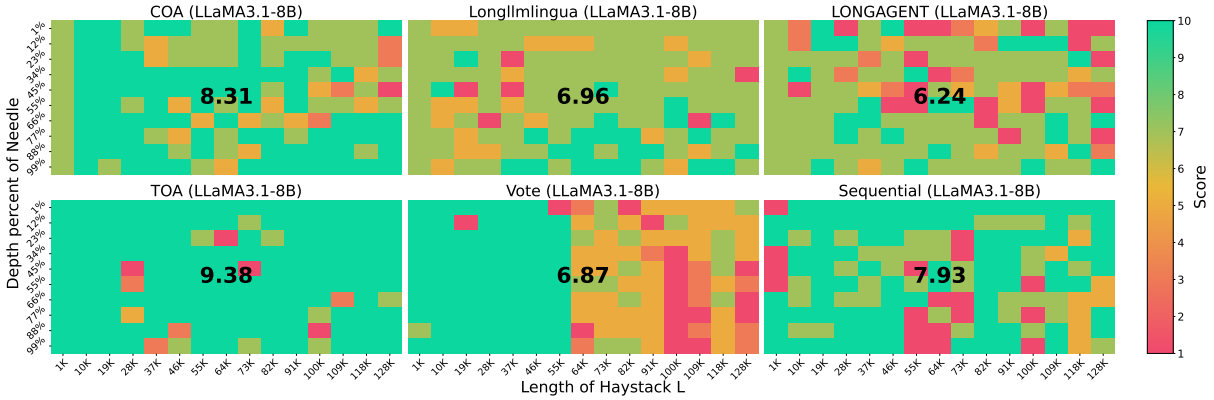

Figure 3: Needle-in-a-Haystack Single-Needle QA results. With TOA, we achieve up to more than 50% performance improvement compared to the baselines, when the length of haystack changes from 1k to 128k, using the same base model LLama3.1-8B. The percentage value on the y-axis represents the depth percentages of Needle. The bold black numbers in each subfigure indicate the average score.

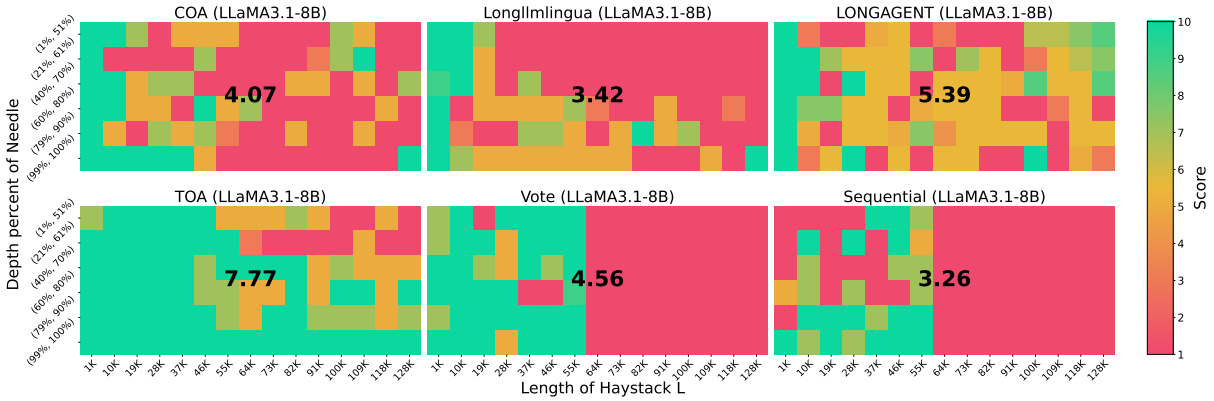

Figure 4: Needle-in-a-Haystack Multi-Needle QA results. With TOA, we achieve up to more than 100% performance improvement compared to the baselines, when the length of haystack changes from 1k to 128k, using the same base model LLama3.1-8B. The two percentages on the y-axis represent the depth of Needle 1 and Needle 2, respectively. The bold black numbers in each subfigure indicate the average score.

LongLLMLingua greatly compresses the input, but it is easy to lose key information when the query is not directly related to the document, resulting in a high none-rate. LONGAGENT performs relatively well, but it depends on the leader's ability to decompose the problem, and performance may decline when the leader makes a wrong judgment. TOA consistently achieves the highest accuracy across all tasks and models, reaching 54.3% and 45.0% on DetectiveQA and NovelQA respectively with LLaMA3.1-8B, and 57.3% and 47.3% with DeepSeek-V3. This demonstrates TOA's strong adaptability across different models.

More notably, TOA maintains a remarkably low none-rate—only 1.7% on DetectiveQA and 4.3% on NovelQA using LLaMA3.1-8B, suggesting its ability to avoid hallucinated answers in cases of uncertainty. These results support the claim that TOA refrains from producing overconfident but incorrect answers. Interestingly, TOA has a comparable performance against the latest and much larger commercial models, such as Gemini 1.5-pro, by using a much smaller model (LLaMA3.1-8B). This shows that the improvements stem from architectural innovation rather than scaling alone, and highlights TOA's parameter efficiency and practical deployment potential. We also investigate the impact of caching and pruning on efficiency, as shown in Appendix D.

5.2 Chunk Size Effects

We conducted a controlled experiment by varying COA's chunk size from 4K to 32K—larger chunks imply fewer agents. As shown in Table 3, COA's performance improves with larger chunks from 4K to 16K but drops at 32K. It reflects the limitation of COA's linear agent chain: deeper information paths increase the risk of context degradation and errors,

LLMs	Baselines	Datasets			
		DetectiveQA		NovelQA	
		Acc↑	None↓	Acc↑	None↓
Llama3.1-8B	COA	0.253 ± 0.012	0.370	0.263 ± 0.017	0.160
	LONGAGENT	0.487 ± 0.031	0.157	0.373 ± 0.026	0.243
	LongLLMLingua	0.307 ± 0.005	0.260	0.170 ± 0.000	0.500
	LongRAG	0.370 ± 0.000	0.217	0.440 ± 0.000	0.153
	TOA	**0.543 ± 0.009**	**0.017**	**0.450 ± 0.028**	0.043
	Sequential	0.400 ± 0.028	0.143	0.257 ± 0.009	0.143
	Vote	0.330 ± 0.022	0.023	0.343 ± 0.017	**0.003**
DeepSeek-V3	COA	0.310 ± 0.022	0.467	**0.480 ± 0.014**	**0.110**
	LongLLMLingua	0.440 ± 0.043	0.257	0.410 ± 0.008	0.313
	LongRAG	0.420 ± 0.008	0.293	0.347 ± 0.026	0.402
	TOA	**0.573 ± 0.005**	**0.140**	0.473 ± 0.012	0.120
	Sequential	0.397 ± 0.025	0.360	0.423 ± 0.017	0.303
	Vote	0.467 ± 0.019	0.220	0.413 ± 0.012	0.130
GPT-4o	/	0.560 ± 0.008	0.140	0.487 ± 0.012	0.270
Gemini1.5-pro	/	0.557 ± 0.005	0.090	0.457 ± 0.019	0.070

Table 2: Performance comparison on long-context reasoning tasks. We select 100 samples for evaluating each task and repeat it three times to reduce errors. Bold text indicates the best result for the same base model, while an underscore denotes the second-best result. TOA achieves the best performance on most tasks and reaches or even exceeds the performance of much larger commercial models on some tasks with a much smaller base model.

Tasks	DetectiveQA		NovelQA	
	Acc↑	None↓	Acc↑	None↓
COA-4K	0.253	0.370	0.263	0.160
COA-8K	0.340	0.290	0.350	0.030
COA-16K	0.450	0.180	0.410	**0.020**
COA-32K	0.400	0.210	0.390	0.060
TOA-32K	**0.543**	**0.017**	**0.450**	0.043

Table 3: More comparisons with COA method on long-context reasoning tasks where 4K represents the length of the text chunk assigned to each agent is 4096.

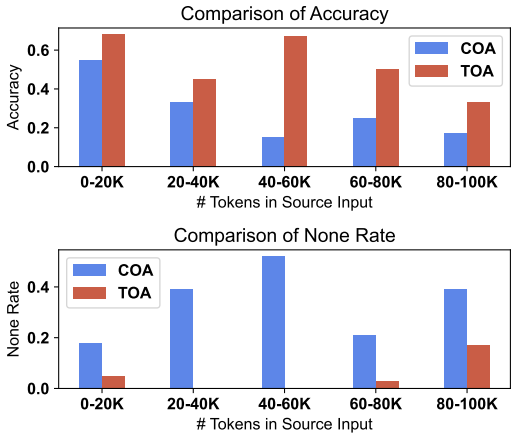

Figure 5: Performance of COA and TOA on DetectiveQA dataset. TOA is more robust to longer inputs.

especially in reasoning tasks. In contrast, TOA allows multi-perspective understanding to support more accurate and stable outputs.

5.3 Robustness to Input Length

We further examine how input length affects the performance of TOA. Figure 5 shows that TOA remains stable up to 100K tokens, with only a slight accuracy drop and consistently low none-rate. In contrast, COA's performance declines sharply beyond 64K, with rising none-rate. This demonstrates TOA's robustness to long input.

5.4 Impact of Agent Number on Performance

To explore the impact of agent quantity N on the performance of TOA, we conducted experiments using various agents for QA tasks. The results are shown in Table 4.

The results show that the performance is optimal when using 5 agents. With fewer agents (3), each agent is tasked with processing larger chunks, which can lead to the "lost in the middle" problem, where key information in the middle of the document is overlooked. On the other hand, when more

Num	DetectiveQA		NovelQA	
	Acc↑	None↓	Acc↑	None↓
Agent=3	0.460	0.170	0.250	0.320
Agent=5	**0.543**	**0.017**	**0.450**	**0.040**
Agent=7	0.510	0.040	0.280	0.190

Table 4: Performance of TOA with different numbers of agents on DetectiveQA and NovelQA tasks. The accuracy (Acc) should be maximized, and the none rate should be minimized.

than 5 agents are used (7 in this case), the text is excessively fragmented, making it harder to synthesize information effectively. Therefore, the use of 5 agents strikes an ideal balance between chunk size and effective information processing, resulting in the best overall performance.

6 Conclusion

This study tackles the challenge of long-context reasoning in LLMs. We proposed TOA to prompt multi-perspective understanding of long text by using multiple agents to probe the segmented text chunks in different orders. To enhance computational efficiency, we introduce prefix-hash caching and adaptive pruning techniques. Empirical evaluations show that TOA, despite being built on the lightweight LLaMA3.1-8B, achieves performance on par with significantly larger commercial models on various long-context reasoning tasks.

Limitations

One limitation of TOA is the additional computational overhead due to probing various orders of text chunks, in particular, as the number of text chunks increases. While caching and pruning techniques are applied, inference remains slower than that of simpler baselines. This poses a challenge for deployment in large-scale settings. Addressing the scalability will be the focus of our future study.

References

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku.

Longze Chen, Ziqiang Liu, Wanwei He, Yinhe Zheng, Hao Sun, Yunshui Li, Run Luo, and Min Yang. 2024a. Long context is not long at all: A prospector of long-dependency data for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8222–8234.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024b. Longlora: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations*.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. Longrope: Extending llm context window beyond 2 million tokens. In *Forty-first International Conference on Machine Learning*.

Ahmed El-Kishky, Alexander Wei, Andre Saraiva, Borys Minaev, Daniel Selsam, David Dohan, Francis Song, Hunter Lightman, Ignasi Clavera, Jakub Pachocki, and 1 others. 2025. Competitive programming with large reasoning models. *arXiv preprint arXiv:2502.06807*.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. 2024. How to train long-context language models (effectively). *arXiv preprint arXiv:2410.02660*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? In *First Conference on Language Modeling*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677.

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen,

and Xia Hu. 2024. Llm maybe longlm: Selfextend llm context window without tuning. In *Proceedings of the 41st International Conference on Machine Learning*, pages 22099–22114.

Daniel Kahneman. 2011. *Thinking, fast and slow*. macmillan.

Greg Kamradt. 2023. Needle in a haystack—pressure testing llms. https://github.com/gkamradt/LLMTest_NeedleInAHaystack.

Huan Yee Koh, Jiaxin Ju, Ming Liu, and Shirui Pan. 2022. An empirical survey on long document summarization: Datasets, models, and metrics. *ACM computing surveys*, 55(8):1–35.

Zhuowan Li, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Retrieval augmented generation or long-context llms? a comprehensive study and hybrid approach. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 881–893.

Hongzhan Lin, Ang Lv, Yang Song, Hengshu Zhu, Rui Yan, and 1 others. 2024. Mixture of in-context experts enhance llms' long context awareness. *Advances in Neural Information Processing Systems*, 37:79573–79596.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 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.

A NEWELL. 1959. Report on a general problem-solving program. In *Proceedings of the International Conference on Information Processing, 1959*, pages 256–264.

Varshini Reddy, Rik Koncel-Kedziorski, Viet Dac Lai, Michael Krumdick, Charles Lovering, and Chris Tanner. 2024. Docfinqa: A long-context financial reasoning dataset. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 445–458.

Zejiang Shen, Kyle Lo, Lauren Yu, Nathan Dahlberg, Margo Schlanger, and Doug Downey. 2022. Multilexsum: Real-world summaries of civil rights lawsuits at multiple granularities. *Advances in Neural Information Processing Systems*, 35:13158–13173.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Dong Shu, Haoran Zhao, Xukun Liu, David Demeter, Mengnan Du, and Yongfeng Zhang. 2024. Lawllm: Law large language model for the us legal system. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 4882–4889.

Woomin Song, Seunghyuk Oh, Sangwoo Mo, Jaehyung Kim, Sukmin Yun, Jung-Woo Ha, and Jinwoo Shin. 2024. Hierarchical context merging: Better long context understanding for pre-trained llms. In *The Twelfth International Conference on Learning Representations*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Cunxiang Wang, Ruoxi Ning, Boqi Pan, Tonghui Wu, Qipeng Guo, Cheng Deng, Guangsheng Bao, Xiangkun Hu, Zheng Zhang, Qian Wang, and 1 others. 2025. Novelqa: Benchmarking question answering on documents exceeding 200k tokens. In *The Thirteenth International Conference on Learning Representations*.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. 2024. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. 2024a. Retrieval meets long context large language models. In *The Twelfth International Conference on Learning Representations*.

Zhe Xu, Jiasheng Ye, Xiangyang Liu, Tianxiang Sun, Xiaoran Liu, Qipeng Guo, Linlin Li, Qun Liu, Xuanjing Huang, and Xipeng Qiu. 2024b. Detectiveqa: Evaluating long-context reasoning on detective novels. *CoRR*.

Howard Yen, Tianyu Gao, and Danqi Chen. 2024. Long-context language modeling with parallel context encoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2588–2610.

Jiajie Zhang, Zhongni Hou, Xin Lv, Shulin Cao, Zhenyu Hou, Yilin Niu, Lei Hou, Yuxiao Dong, Ling Feng, and Juanzi Li. 2024a. Longreward: Improving long-context large language models with ai feedback. *arXiv preprint arXiv:2410.21252*.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Arik. 2024b. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237.

Jun Zhao, Can Zu, Xu Hao, Yi Lu, Wei He, Yiwen Ding, Tao Gui, Qi Zhang, and Xuan-Jing Huang. 2024a. Longagent: Achieving question answering for 128k-token-long documents through multi-agent collaboration. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16310–16324.

Qingfei Zhao, Ruobing Wang, Yukuo Cen, Daren Zha, Shicheng Tan, Yuxiao Dong, and Jie Tang. 2024b. Longrag: A dual-perspective retrieval-augmented generation paradigm for long-context question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22600–22632.

Shiyi Zhu, Jing Ye, Wei Jiang, Siqiao Xue, Qi Zhang, Yifan Wu, and Jianguo Li. 2024. Coca: Fusing position embedding with collinear constrained attention in transformers for long context window extending. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4247–4262.

A Datasets and Benchmark

In our experiments, we utilized two datasets: DetectiveQA and NovelQA, selected for their relevance to multi-hop and complex reasoning tasks. For the DetectiveQA dataset, we exclusively used the human-annotated English corpus to ensure high annotation quality. Each question requires the model to select one correct answer from four provided options. For the NovelQA dataset, we focused on complex reasoning scenarios by specifically extracting multi-hop reasoning questions from the dataset. Due to data leakage concerns, the ground-truth answers for NovelQA are not publicly available. To obtain evaluation results, we submitted the predictions via Codabench [*]. Due to computational budget constraints, we randomly sampled 100 examples from each dataset for our experiments. To account for potential model variability, we conducted three independent runs for all experiments and reported the mean value and standard deviation as results.

We also use the Needle-in-a-Haystack benchmark to assess the ability to retrieve information in long-context documents. In the single-needle setting, we used the following sentence as the needle: "The production company for *The Year Without a Santa Claus* is best known for seasonal television specials, particularly its work in stop-motion animation." The corresponding question was: "For what type of work is the production company for *The Year Without a Santa Claus* best known?". In the multi-needle setting, we used two distinct needle sentences:

- "According to declassified Cold War documents, spies used a hollowed-out chess piece as a dead drop in 1970s Berlin."
- "According to declassified Cold War documents, a fake electrical fuse box was used as a dead drop by spies in 1970s Berlin."

The associated question was: "According to declassified Cold War documents, what were the two unusual objects that spies used as dead drops in 1970s Berlin?"

B Baselines

All experiments were conducted using a machine equipped with two NVIDIA GeForce RTX 3090 GPUs, an Intel Core i9-14900K CPU, and 128 GB of RAM. Implementations were based on PyTorch v2.6.0. To ensure consistency across all baselines, we set the decoding temperature to 0.01 and the maximum output length to 2048 tokens. For locally run models (e.g., LLaMA3.1-8B-instruct), we used FP16 precision and performed inference on the same hardware setup.

- For LONGAGENT, we manually reimplemented the method by closely following the steps described in the original paper. Specifically, we fine-tuned LLaMA3.1-8B on 10,000 positive and 15,000 negative samples (constructed from the same data distribution). The chunk size was set to the default value of 4096 tokens.
- For COA, we used its open-source implementation with the default chunk size of 4096. To evaluate the impact of different chunk sizes, apart from the default size of 4096, we also tested COA on various chunk sizes (see Section 5.2 for details).
- For LongLLMLingua, we used the open-source prompt compression implementation with all parameters at their default values. The compressed prompts were subsequently passed to our base model for evaluation.
- For LongRAG, we employed the official open-source implementation. The chunk size was set to the maximum supported value of 500, with all other parameters left at default. Prior to evaluation, all documents were preprocessed via chunking and vectorization for retrieval.
- For commercial models such as Gemini-1.5-Pro and GPT-4o, we directly fed the documents and questions into the models and prompted them to generate answers without further customization.
- For TOA, we set the number of agents to 5. In both TOA and the Vote baseline, if some agents returned 'None' during the voting process, we only considered valid responses. In cases of a tie, an additional independent agent was queried to make the final decision.

C The Impact of Reading Order

To better understand the importance of reading order in multi-hop reasoning, we present a simple

[*] https://www.codabench.org/competitions/2727/

illustrative example:

Imagine a short story:

1. "The room was messy."
2. "John sighed."
3. "He had just finished a big project."

If you read these sentences in order (1→2→3), you might infer that John sighed because the messy room stressed him out. However, if you reverse the order (3→2→1), it suggests that John sighed in relief after completing his project, and the messy room is merely a background detail. The same facts, presented in different orders, lead to entirely different interpretations. By probing various orders of text chunks, TOA realizes multi-perspective reasoning and avoids being constrained to a single narrative flow. This enables us to uncover how order shapes meaning, potentially help us reveal hidden biases or overlooked perspectives.

D Efficiency

To better measure the efficiency of our method, Table 5 presents the number of API calls by TOA and baseline COA. Both use DeepSeek-V3 as the base model on the NovelQA dataset. The main efficiency gains in TOA occur in phase 2, where caching and pruning are applied. Without caching and pruning, phase 2 requires an estimated 2103 API calls. Enabling caching alone reduces this number to 1830, saving 273 calls (a 13.0% reduction). When both caching and pruning are applied, the number of calls drops further to 1034 — saving at least 1069 calls (a 50.8% improvement).

We further compared the number of token requests in Table 6. Pruning and caching significantly reduced overhead. It's worth noting that this mechanism relies on the capabilities of the base model itself, so its effect is not significant on models with small parameters. Although the total cost in TOA is slightly higher than that of COA, this additional cost yields a notable improvement in output.

E Algorithm for TOA

The pseudocode of TOA is presented in Algorithm 1.

F Prompt Templates Used in TOA

Table 7 - 11 present the prompt templates used in the three phases of TOA, respectively.

G Case Study

A case study is described in Table 12.

Phase 1: Chunk Perception

The document is split into five chunks and allocated to five agents, each for one. Each agent generates the first response based on their chunks. Because on different chunks, agents can respond vary differently. For instance, Agent 0 identifies the victims as hotel regulars, whereas Agent 1 views them as business associates.

Phase 2: Multi-Perspective Understanding

Once the agents generate their initial cognition, they proceed to the multi-perspective understanding. Agents exchange their cognition and request access to other agents' chunks to update their cognition. A key feature is that each agent probes the different orders of text chunks for multi-perspective understanding of the input document. For example, Agent 0 explores text chunks 2, 3, and 4. During this process, agents evaluates the utility of each chunk. If a chunk is deemed unhelpful (marked as "useless"), the path is pruned. The pruning strategy ensures that reasoning efforts are focused on informative content, reducing unnecessary computation. Another strategy applied is state caching with prefix hashing. For instance, Agent 0 reuses the cached state after evaluating Agent 3's chunk. This caching strategy is especially effective when certain sequences of chunks are revisited when probing paths.

Phase 3: Consensus Formation

Following multi-perspective understanding, agents synthesize the gathered information to form a final consensus. In this example, all five agents converge on the same answer **A**. The final decision is determined through majority vote among agents, targeting for a more robust and collectively reasoned decision.

Method	Phase	Strategy	API Calls	Saved Calls	Saving Rate
TOA	Phase 2	w/o Caching & Pruning	2103	–	–
		w/ Caching Only	1830	273	13.0%
		w/ Caching & Pruning	1034	1069	50.8%
	Phase 1 & 3	–	1500	–	–
	All Phases	w/ Caching & Pruning	**2534**	–	–
COA	–	–	**2287**	–	–

Table 5: The number of API calls by TOA and COA on the NovelQA dataset. The evaluation is conducted on the same 100 examples used in the experiments.

Dataset	Model	TOA Tokens			Token Savings	COA Tokens
		NO Cache&Prune	with Cache	with Cache+Prune		
NovelQA	DeepSeek-V3	31210K	25609K	12804K	59%	10229K
	Llama3.1-8B	24798K	23495K	23476K	5%	10229K
DetectiveQA	DeepSeek-V3	18438K	15522K	12292K	33%	5414K
	Llama3.1-8B	12575K	11933K	11932K	5%	5414K

Table 6: Token statistics across datasets and models with different caching/pruning settings. The evaluation is conducted on the same 100 examples used in the experiments.

Prompt Template for Chunk Perception (Phase 1)

Background:
You are a skilled agent tasked with answering a question based on a long context. Since the context is too long, it is divided into chunks, each assigned to a different agent.

Task:
You are in Phase 1. Given a document chunk and a multiple-choice question, your goal is to answer the question accurately. First, extract and summarize facts relevant to the question from your assigned segment. Then, draw your conclusion based solely on those facts. Do not rely on prior knowledge.

Output Format (JSON):

```
{
   "evidence": "Factual excerpts supporting your reasoning",
   "answer": "Your answer based on the evidence"
}
```

Table 7: Prompt template (Phase 1).

Algorithm 1 TOA algorithm.

Input: Document $\mathcal{D}$, chunk number N, query q and prompt p.
Output: Final answer a^{final}.

Split $\mathcal{D}$ into small chunks $\mathcal{C} = \{c_i\}_{i=0}^{N}$
for $i = 0$ to $N - 1$ **do**
  Initialize $\mathcal{M}_i$ to store all cognitions
  Initialize $\mathcal{U}_i$ to store all chunk sequence usefulness
  $s_i^{(i)} \leftarrow \langle e_i^{(i)}, a_i^{(i)} \rangle = \mathcal{A}_i(q, c_i, p)$
  $\mathcal{M}_i(\langle c_i \rangle) \leftarrow s_i^{(i)}$
end for
for $i = 0$ to $N - 1$ **do**
  Get the helpful chunks index set $\mathcal{G}_i$
  if $\mathcal{G}_i \neq \emptyset$ **then**
    $\text{Perm}_i \leftarrow \text{Permutations}(\mathcal{G}_i)$
    for $\pi \in \text{Perm}_i$ **do**
      $\mathcal{P}_\pi = \langle c_i \rangle \oplus \langle c_{\pi_1}, c_{\pi_2}, ..., c_{\pi_k} \rangle$
      for $r = 2$ to $|\mathcal{P}_\pi|$ **do**
        $\sigma_r \leftarrow \mathcal{P}_\pi[0:r]$
        if $\sigma_r \in \mathcal{U}_i$ **then**
          if $\mathcal{U}_i(\sigma_r) == 0$ **then**
            break
          else
            $s_i^{\sigma_r} = \mathcal{M}_i(\sigma_r)$
            continue
          end if
        end if
        if $\text{Utility}(s_i^{\sigma_{r-1}}, c_{\pi_r}) == 0$ **then**
          $\mathcal{U}_i(\sigma_r) \leftarrow 0$
          break
        else
          $s_i^{\sigma_r} \leftarrow \mathcal{A}_i(s_i^{\sigma_{r-1}}, c_{\pi_r})$
          $\mathcal{M}_i(\sigma_r) \leftarrow s_i^{\sigma_r}$
          $\mathcal{U}_i(\sigma_r) \leftarrow 1$
        end if
      end for
    end for
  end if
end for
for $i = 0$ to $N - 1$ **do**
  $\sigma_i^* \leftarrow \text{argmax}_{\sigma \in (\mathcal{M}_i)} |\sigma|$
  $a_i^{\text{final}} \leftarrow \mathcal{A}_i(q, \mathcal{M}_i(\sigma_i^*), p)$
end for
$a^{\text{final}} \leftarrow \text{MajorityVote}(\{a_i^{\text{final}}\}_{i=0}^{N-1})$
return a^{final}

Prompt Template for Multi-Perspective Understanding (Phase 2-1)

Background:
You are a skilled agent tasked with answering a question based on a document. Since the document is too long, it is divided into multiple chunks, each read by a different agent.

Task:
You are in Phase 2-1. You have already read your assigned chunk and proposed an evidence-based answer. However, your view may be incomplete or incorrect due to the limited context.

You will now be shown the evidence and answers provided by other agents who read different chunk of the document. The correct answer may appear in one or more of these responses.

Note:
If only a few agents report relevant evidence while most say there is none, you should focus on those few with relevant content.

Decision:
Select which agent(s)' responses may help refine your understanding without introducing irrelevant information. You may choose one or more agent IDs, or `"None"` if no agent adds value.

Use only the information shown. Do not use external knowledge.

Valid choices: `{agent_list}`

Output Format (JSON):

```
{
  "explanation": "Justify your selection.",
  "id": "Selected agent ID(s), e.g., '0', '0,1', or 'None'"
}
```

Table 8: Prompt template (Phase 2-1).

Prompt Template for Multi-Perspective Understanding (Phase 2-2)

Background:
You are a skilled agent tasked with answering a question based on a document. Since the document is too long, it is divided into chunks, each read by a different agent.

Task:
You are in Phase 2-2. Based on your earlier reasoning, you requested to view additional text chunks from other agents to refine your understanding.

You will now be shown one of these chunks. Carefully evaluate its relevance. If the chunk only repeats known information or introduces irrelevant content, mark it as `"useless"`. Otherwise, mark it as `"useful"` and update your facts and conclusion accordingly.

If the chunk is `"useless"`, repeat your original facts and conclusion. Limit your output length—abbreviate if necessary.

You must judge only based on the content provided, not using external knowledge.

Output Format (JSON):

```
{
  "utility": "useless" or "useful",
  "fact": "Updated factual summary.",
  "conclusion": "Updated answer based on new information."
}
```

Table 9: Prompt template (Phase 2-2).

Prompt Template for Consensus Formation (Phase 3-1)

Background:
You are a skilled agent tasked with answering a question based on a document. Since the document is too long, it is divided into chunks, each read by a different agent.

Task:
You are in the final phase. All agents have now exchanged their opinions. Based solely on the question, answer options, and your aggregated opinion, provide the final answer.

If you are still uncertain and unable to choose a valid option, respond with `"None"`.

Note:
All relevant information has been condensed into your own opinions. Do not consider external content or reprocess the original document. Make your decision based only on your internal conclusion.

Output Format (JSON):

```
{
  "explanation": "Brief reasoning for your choice.",
  "result": "One of A, B, C, D, or None (no punctuation)"
}
```

Table 10: Prompt template (Phase 3-1).

Prompt Template for Tie-Breaking Decision (Phase 3-2)

Background:
You are the final decision maker. You are presented with a long document and a multiple-choice question.

There are `{len(Agent_list)}` decision makers. A majority vote was attempted, but a tie occurred.

Task:
Please examine each agent's factual conclusions and opinions carefully. Based on this information, select the best final answer.

Rules:
1. You **MUST** choose from the following options: `{result}`
2. **DO NOT** generate any answer outside this list.
3. Output your decision strictly in the following JSON format.

Tie Information:
Answers with the same number of votes: `{result}`

Output Format (JSON):

```
{
  "explanation": "Justify your choice.",
  "result": "Final answer choice, e.g., A or B"
}
```

Table 11: Prompt template (Phase 3-2).

Query	*"What is the relationship between the three victims?"*
Assign Documents	`Assigned to 5 agents, length 5*4428.0 = 22140`
Chunk Perception	**Agent 0:** D. *They were all regular customers at the same hotel.* **Agent 1:** C. *The victims were known to frequently exchange business proposals.* **Agent 2:** C. *All three victims had booked rooms under the same group reservation.* **Agent 3:** C. *Each victim was connected to a similar project involving a large sum of money.* **Agent 4:** A. *The victims had a shared history of working together.*
Multi-Perspective Understanding	**Agent 0:** I saw Agent [1, 2, 3, 4]'s cognition, want to see Agent [2, 3, 4]'s chunk. **Agent 1:** I saw Agent [0, 2, 3, 4]'s cognition, want to see Agent [4]'s chunk. **Agent 2:** I saw Agent [0, 1, 3, 4]'s cognition, want to see Agent [4]'s chunk. **Agent 3:** I saw Agent [0, 1, 2, 4]'s cognition, want to see Agent [4]'s chunk. **Agent 4:** I saw Agent [0, 1, 2, 3]'s cognition, want to see Agent [0]'s chunk.
Multi-Perspective Understanding	**Agent 0:** Begin sequence [2, 3, 4]. —Saw Agent [2]'s chunk — `useless`, marked (0, 2) as useless. ——Because [2] is useless, skip the rest of the sequence. Begin sequence [2, 4, 3]. —(0, 2) is already proved useless, skip the rest of the sequence. Begin sequence [3, 2, 4]. —Saw Agent [3]'s chunk - `useful`. ——New cognition added: (0, 3). —Saw Agent [2]'s chunk - `useless`, marked (0, 3, 2) as useless. ——Because [2] is useless, skip the rest of the sequence. Begin sequence [3, 4, 2]. —Load cache (0, 3), Saw Agent [4]'s chunk - `useful`. ——New cognition added: (0, 3, 4). —Saw Agent [2]'s chunk - `useless`, marked (0, 3, 4, 2) as useless. Begin sequence [4, 2, 3]. —Saw Agent [4]'s chunk - `useful`. ——New cognition added: (0, 4). —Saw Agent [2]'s chunk - `useless`, marked (0, 4, 2) as useless. ——Because [2] is useless, skip the rest of the sequence. Begin sequence [4, 3, 2]. —Load cache (0, 4), Saw Agent [3]'s chunk - `useful`. ——New cognition added: (0, 4, 3). —Saw Agent [2]'s chunk - `useful`. ——New cognition added: (0, 4, 3, 2). **Agent 1:** Begin Sequence [4]. —Saw Agent [4]'s chunk — `useful` ——New cognition added: (1, 4) **Agent 2:** Begin Sequence [4]. —Saw Agent [4]'s chunk — `useful` ——New cognition added: (2, 4) **Agent 3:** Begin Sequence [4]. —Saw Agent [4]'s chunk — `useful` ——New cognition added: (3, 4) **Agent 4:** Begin Sequence [0]. —Saw Agent [0]'s chunk — `useful` ——New cognition added: (4, 0)
Cognition Summary	(0), (0, 3), (0, 4), (0, 3, 4), (0, 4, 3), (0, 4, 3, 2) (1), (1, 4) (2), (2, 4) (3), (3, 4) (4), (4, 0)
Longest Cognitions	(0, 4, 3, 2), (1, 4), (2, 4), (3, 4), (4, 0)
Consensus Formation	**Agent 0:** A **Agent 1:** A **Agent 2:** A **Agent 3:** A **Agent 4:** A **Majority Answer:** `A`

Table 12: A case study