Think&Cite: Improving Attributed Text Generation with Self-Guided Tree Search and Progress Reward Modeling

Junyi Li and **Hwee Tou Ng**
Department of Computer Science, National University of Singapore
junyi_cs@nus.edu.sg, nght@comp.nus.edu.sg

Abstract

Despite their outstanding capabilities, large language models (LLMs) are prone to hallucination and producing factually incorrect information. This challenge has spurred efforts in attributed text generation, which prompts LLMs to generate content with supporting evidence. In this paper, we propose a novel framework, called **Think&Cite**, and formulate attributed text generation as a multi-step reasoning problem integrated with search. Specifically, we propose Self-Guided Monte Carlo Tree Search (SG-MCTS), which capitalizes on the self-reflection capability of LLMs to reason about the intermediate states of MCTS for guiding the tree expansion process. To provide reliable and comprehensive feedback, we introduce Progress Reward Modeling to measure the progress of tree search from the root to the current state from two aspects, i.e., generation and attribution progress. We conduct extensive experiments on three datasets and the results show that our approach significantly outperforms baseline approaches.[1]

1 Introduction

Large language models (LLMs) (Zhao et al., 2023) have achieved outstanding performance on many natural language processing tasks. Despite the advances, LLMs often generate responses that contain hallucinations and inaccurate information (Ji et al., 2023; Huang et al., 2023; Zhang et al., 2023). This issue undermines their reliability, and more importantly, hurts users' trust in LLMs. To improve the reliability of LLMs, a new paradigm for generation, *attributed text generation*, is proposed, such that LLMs generate responses with in-text citations that provide evidence for any statement (Gao et al., 2023b), as shown in Figure 1.

Most existing work (Slobodkin et al., 2024; Sun et al., 2024; Fierro et al., 2024) simply prompts LLMs to provide citations while generating texts. Besides, other work (Li et al., 2024; Huang et al., 2024) attempts to fine-tune LLMs on massive supervised training data that contains texts with annotated citations. Despite these recent efforts, it remains an open challenge to develop LLMs capable of learning to generate faithful content with reliable references. First, existing approaches adopt an auto-regressive generation paradigm that can be characterized as "System 1", a mode of thinking which is fast and instinctive, but less accurate (Kahneman, 2011). Thus, any intermediate generation errors (e.g., false statements or erroneous citations) can potentially lead to incorrect final responses. Inspired by research on complex reasoning (Zhang et al., 2024; Wang et al., 2024), we aim to develop models in the "System 2" mode for attribution to external evidence, requiring more in-depth, deliberative, and logical thinking (Kahneman, 2011). Second, attributed text generation often involves long text generation. Liu et al. (2023) find that long-form responses from existing LLMs usually contain unsupported statements and inaccurate citations. We argue that the absence of explicit generation planning in previous work hinders advances in such systems.

In this paper, we propose **Think&Cite**, a novel framework integrating search algorithms into attributed text generation. We formulate the task as

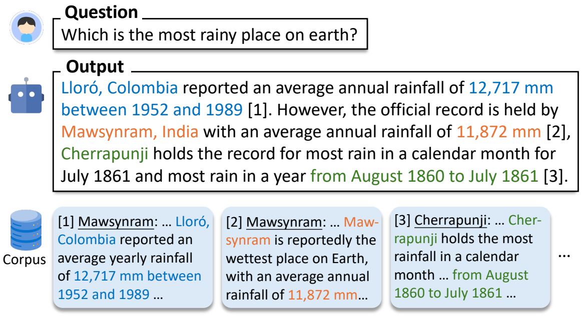

Figure 1: Given a question, the model generates texts by citing passages from a corpus as supporting evidence.

[1]Our dataset and source code are available at https://github.com/nusnlp/Think-Cite.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9928–9942
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

a multi-step reasoning problem, where the model generates one sentence in each step through an iterative *think-verbalize-cite* paradigm. To enhance this generation process, we propose Self-Guided Monte Carlo Tree Search (SG-MCTS), which extends the classic MCTS with two innovations. First, our approach leverages the self-reflection capability of LLMs to deliberate on the *intermediate states of MCTS* in real time, so as to guide the tree expansion process and proactively avoid inadequate reasoning paths. This is different from prior work which mainly reflected on the final outcome or complete trajectory. Second, we propose Progress Reward Modeling (PRM) to measure *the progress of tree search* from the root to the current state from two aspects, i.e., generation progress and attribution progress. In contrast to only evaluating single steps, progress-based reward modeling can provide reliable and comprehensive evaluation to guide the MCTS search process.

To the best of our knowledge, we are the first to apply tree search algorithms to the task of attributed text generation. We conduct extensive experiments on three datasets to verify the effectiveness of our approach. The results show that our model significantly outperforms previous prompting-based and fine-tuning baselines.

2 Related Work

Attributed Text Generation. Large language models (LLMs) have been used in attributed text generation due to their outstanding language generation capabilities (Gao et al., 2023b; Huang et al., 2024; Sun et al., 2024; Li et al., 2024; Slobodkin et al., 2024). The work on LLMs for attributed text generation can be broadly categorized into two types. The first type involves fine-tuning LLMs with preference learning (Li et al., 2024) and reinforcement learning (Huang et al., 2024), which teach LLMs to generate supportive and relevant citations to achieve higher rewards. However, this approach depends on human labor to curate high-quality datasets with annotated in-text citations. Another line of work directly instructs LLMs to generate attributed texts with appropriate prompts by attribute-then-generate planning (Slobodkin et al., 2024), or employing external verifiers to guide generation (Sun et al., 2024). However, this approach generates texts and citations in an auto-regressive manner, where any inaccurate intermediate generation can easily lead to failure in the subsequent process. In contrast, our approach proposes self-guided tree search with progressive reward to consider multiple paths.

LLMs with Tree Search. Integrating tree search algorithms with LLMs has attracted significant attention. Recent studies have investigated the use of tree search methods to enhance the performance of LLMs during inference (Zhang et al., 2024; Wang et al., 2024; Ye and Ng, 2024). Sutton (2019) highlights the superiority of scaling in both learning and search, over other approaches. Empirical evidence further demonstrates that scaling inference-time computation can significantly improve LLM performance without requiring additional training (Brown et al., 2024; Snell et al., 2024). A* search (Hart et al., 1968) and Monte Carlo Tree Search (MCTS) (Browne et al., 2012) are employed as planning techniques to improve the performance of LLMs in solving complex reasoning problems. Lewis et al. (2020) introduces the retrieval score in sequence likelihood to facilitate tree search and Self-RAG (Asai et al., 2024) explicitly supports tree-decoding with critique tokens. Our work is the first to apply tree search algorithms (i.e., Monte Carlo Tree Search) to solve the task of attributed text generation. Moreover, we propose self-guided MCTS that relies on the reflection capability of LLMs to improve tree expansion.

3 Problem Formulation

Our proposed framework aims to have a pre-trained LLM $\mathcal{M}_\theta$ generate responses with in-text citations that serve as evidence for the output content, referred to as *attributed text generation* (Slobodkin et al., 2024; Gao et al., 2023a).

Formally, given an input question $\boldsymbol{x}$ and a corpus of text passages $\mathcal{D}$, the model $\mathcal{M}_\theta$ is required to generate a response $\boldsymbol{y} = \langle y_1, ..., y_T \rangle$ consisting of T sentences where each sentence y_t cites a list of passages from $\mathcal{D}$, denoted by $\mathcal{C}_t = \{c_{t,1}, ..., c_{t,m}\}$. Due to the marginal benefit of incorporating more citations (Gao et al., 2023b), in this paper, we allow at most three citations for each sentence ($m \leq 3$), and these citations are enclosed in square brackets, such as `[1][2]`. We also mainly focus on knowledge-intensive scenarios where the question concerns world knowledge and most sentences from LLMs contain multiple facts and require supporting citations as evidence. Following prior work (Gao et al., 2023b; Piktus et al., 2021), we divide the corpus $\mathcal{D}$ into 100-word passages

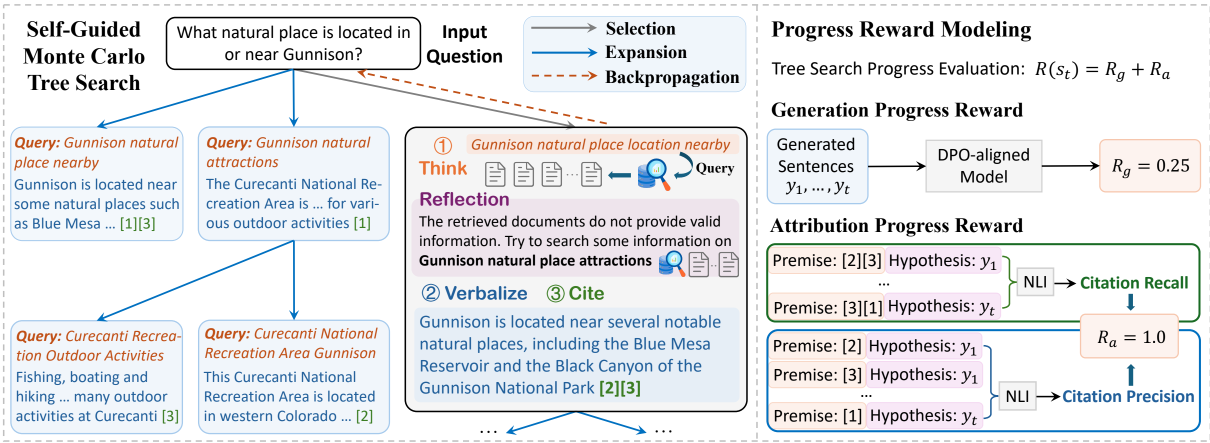

Figure 2: Overall framework of our proposed Think&Cite approach.

for fine-grained retrieval, which makes it easier for humans to verify and does not introduce too much irrelevant information.

4 Approach

The proposed THINK&CITE framework builds on a language agent for attributed text generation, using Self-Guided Monte Carlo Tree Search (SG-MCTS) to plan and search over multiple generation paths, and Progress Reward Modeling to provide progressive signals for the search process. Figure 2 depicts the overall framework of our approach.

4.1 Attributed Text Generation Agent

Inspired by prior work (Yao et al., 2022; Chen et al., 2023), we develop a language agent to address the task of attributed text generation, which performs an iterative *think-verbalize-cite* process, leveraging the reasoning and planning capabilities of LLMs.

Iterative Think-Verbalize-Cite. In our approach, the language agent generates one sentence in each step. To generate the t-th sentence, the agent first proactively thinks about the blueprint, such as topic or abstract (Narayan et al., 2023), for the next sentence that serves as a retrieval query q_t. Then the agent uses the search tool to retrieve the most relevant top-K passages $\mathcal{D}_t$ from the given corpus $\mathcal{D}$ through the Search action, i.e., "`Search: {query}`". Based on the retrieved passages, the agent verbalizes a sentence y_t by citing a list of passages $\mathcal{C}_t$ from $\mathcal{D}_t$ through the `Generate` action, i.e., "`Generate: {sentence}`". The historical queries, retrieved passages, generated sentences, and citations, denoted as $\mathcal{H} = \{\langle q_i, \mathcal{D}_i, y_i, \mathcal{C}_i \rangle\}_{i=1}^{t}$ will be combined as the context for thinking and verbalizing in the next step. If the agent believes that the task has been solved, it can output "End" to terminate this process. In this way, the agent deliberately plans and retrieves diverse information, which can dynamically consider shift in content focus as the generation progresses, in contrast to prior work relying on a static reference corpus (Slobodkin et al., 2024; Huang et al., 2024; Li et al., 2024; Fierro et al., 2024). Besides, this paradigm is similar to recent work on iterative retrieval-augmented generation (Jiang et al., 2023; Shao et al., 2023), but differs in that our work requires the model to anticipate a content blueprint for the next generation to retrieve relevant information and carefully select appropriate references for incorporating them at suitable positions within the generated text.

4.2 Self-Guided Monte Carlo Tree Search

We formulate attributed text generation as a multi-step reasoning problem where the model deliberates on the attribution for text. Monte Carlo Tree Search has become an effective search algorithm for many decision-making tasks (Silver et al., 2016; Ye et al., 2021). In this work, we propose *Self-Guided Monte Carlo Tree Search (SG-MCTS)*, which capitalizes on the self-reflection capability of LLMs to guide the search process of MCTS. Previous work (Shinn et al., 2023; Zhou et al., 2024; Yu et al., 2024) often reflects on the final outcome or the complete trajectory, which is inefficient and sparse. In contrast, our method aims to criticize and deliberate on *intermediate states* of MCTS to guide tree expansion in real time and proactively ignore erroneous generation paths.

Generally, MCTS builds a search tree $\mathcal{T}$ based on a policy model π_θ, which is often the LLM $\mathcal{M}_\theta$. In this tree, node $s_t = [q_t, \mathcal{D}_t, y_t, \mathcal{C}_t]$ denotes a state at the t-th tree level, including search query

q_t, retrieved passages $\mathcal{D}_t$, generated sentence y_t, and cited passages $\mathcal{C}_t$. The root node $s_0 = [\boldsymbol{x}]$ denotes the input question. Each node contains one sentence and the final output is the concatenation of sentences $\langle y_1, ..., y_T \rangle$, where each sentence y_t comes from a node on the path from the root node to the leaf node. In each iteration, SG-MCTS follows four steps, i.e., selection, reflection-guided expansion, evaluation, and backpropagation.

Selection Phase. The selection stage aims to identify a node s_t from the search tree $\mathcal{T}$ for subsequent expansion. The Upper Confidence Bound applied to Trees (UCT) algorithm (Kocsis and Szepesvári, 2006) is employed to select the optimal node with the highest UCT score:

$$UCT(s_t) = V(s_t) + w\sqrt{\frac{\ln N(p)}{N(s_t)}}, \quad (1)$$

where $V(s_t)$ is the value function (expected reward) of s_t estimated at the evaluation stage, $N(s_t)$ is the visit count of s_t, w is a weight controlling exploration, and p is the parent node of s_t.

Reflection-Guided Expansion Phase. In the expansion phase, the selected node s_t is expanded by generating successor node s_{t+1} through the think-verbalize-cite process. The **Think** step first generates an initial search query $\hat{q}_{t+1}$, abstracting the topic or content of the next sentence, which will be used to retrieve passages $\widehat{\mathcal{D}}_{t+1}$. However, the initial query might be obscure or inaccurate, which can hinder subsequent evidence retrieval and ultimately result in incorrect sentence generation. Moreover, some questions do not have straightforward answers, necessitating iterative refinement to retrieve and generate accurate results. Therefore, we introduce the **Reflection** step, where the model reflects on the initial query $\hat{q}_{t+1}$ to identify errors based on the question $\boldsymbol{x}$ and retrieved passages $\widehat{\mathcal{D}}_{t+1}$ as:

$$u = \mathcal{M}_\theta(\hat{q}_{t+1}, \widehat{\mathcal{D}}_{t+1}, \boldsymbol{x}), \quad (2)$$

where the reflection text u includes retrieval advice on certain aspects, e.g., the query should be more focused in search topics. Based on the reflection, the policy model reformulates a new query q_{t+1} to retrieve more relevant passages $\mathcal{D}_{t+1}$:

$$q_{t+1}, \mathcal{D}_{t+1} = \mathcal{M}_\theta(u, \hat{q}_{t+1}, \widehat{\mathcal{D}}_{t+1}). \quad (3)$$

Note that the reflection step can be iterated until the model determines that the retrieved evidence is supportive or the maximum number of reflection steps is reached. Finally, the **Verbalize** and **Cite** steps generate the next sentence y_{t+1} with accurate citations $\mathcal{C}_{t+1}$ from $\mathcal{D}_{t+1}$ as:

$$y_{t+1}, \mathcal{C}_{t+1} = \mathcal{M}_\theta(q_{t+1}, \mathcal{D}_{t+1}, \mathcal{H}), \quad (4)$$

where $\mathcal{H}$ is the historical context. The new node includes search query, retrieved corpus, generated sentence, and cited passages, denoted as $s_{t+1} = [q_{t+1}, \mathcal{D}_{t+1}, y_{t+1}, \mathcal{C}_{t+1}]$. Compared to simple expansion in typical MCTS, our approach can refine flawed expanded nodes to avoid low-quality generation. Since the MCTS tree is built step by step, improving the quality of the next action allows the model to navigate more favorable pathways in the vast search space, thereby enhancing the overall search quality and efficiency of the tree.

Evaluation Phase. The evaluation stage aims to compute the expected reward $R(s_{t+1})$ of the newly expanded node s_{t+1} using Progress Reward Modeling (see Section 4.3). The progress evaluation involves two aspects: generation and attribution. The generation progress reward R_g measures the *textual quality* of generated sentences so far $y_1, ..., y_{t+1}$. The attribution progress reward R_a evaluates the *attribution consistency* between generated sentences $y_1, ..., y_{t+1}$ and cited passages $\mathcal{C}_1, ..., \mathcal{C}_{t+1}$. Finally, the total reward is computed as the sum of both: $R(s_{t+1}) = R_g + R_a$.

Backpropagation Phase. In the backpropagation phase, the reward $R(s_{t+1})$ of the new node is propagated back to its parent node s_t, updating the value function of each node $s_0, s_1, ..., s_t$ along the path from the root node to its parent node:

$$N_{\text{new}}(s_i) = N_{\text{old}}(s_i) + 1, \quad 0 \le i \le t \quad (5)$$

$$V_{\text{new}}(s_i) = \frac{V_{\text{old}}(s_i)N_{\text{old}}(s_i) + R(s_{t+1})}{N_{\text{new}}(s_i)}, \quad (6)$$

where $N_{\text{old}}(s_i)$ and $V_{\text{old}}(s_i)$ are the prior visit count and value function of node s_i, respectively.

The above four steps are performed iteratively until the policy model outputs "`End`", indicating the task has been solved or the maximum number of MCTS iterations is reached.

4.3 Progress Reward Modeling

Previous outcome reward models (Hosseini et al., 2024) and process reward models (Lightman et al., 2024) mainly evaluate the final result or intermediate steps. In this work, we propose to measure the progress of tree search from the root s_0 to state s_{t+1} after taking the next step. Since an attributed text includes the text and its citations, we design two aspects for progress reward modeling, *Generation Progress Reward* and *Attribution Progress Reward*, to evaluate the quality of the generated textual content and the relevance of citations, respectively.

4.3.1 Generation Progress Reward

In direct preference optimization (DPO) (Rafailov et al., 2023), the token-level log-ratio can be explained as an implicit token-level reward under a max-entropy reinforcement learning (RL) formulation. Thus, we propose to leverage existing DPO-aligned models to measure the quality score R_g of the generated sentences $\boldsymbol{y}_{1:t+1} = y_1, ..., y_{t+1}$ after generating the next sentence y_{t+1}.

Specifically, we define a sentence-level Markov Decision Process (MDP) where the state $\boldsymbol{s}_t = \langle \boldsymbol{x}, y_1, ..., y_t \rangle$ denotes the input and sentences generated so far and the initial state $\boldsymbol{s}_0 = \boldsymbol{x}$ is the input question. The action $\boldsymbol{a}_t = y_{t+1}$ denotes the next sentence to be generated. Hence, the RLHF optimization objective can be rewritten as a max-entropy RL problem at the sentence level:

$$\mathbb{E}_{\boldsymbol{a}_t \sim \pi_\theta(\cdot|\boldsymbol{s}_t)}[\sum_{t=1}^{T} r'(\boldsymbol{s}_t, \boldsymbol{a}_t)] + \beta \mathbb{E}_{\boldsymbol{s}_0 \sim \mathcal{X}}[\mathcal{H}(\pi_\theta(\cdot|\boldsymbol{s}_0))],$$

where the sentence-level reward function r' can be calculated as:

$$r'(\boldsymbol{s}_t, \boldsymbol{a}_t) = \begin{cases} \beta \log \pi_{\text{ref}}(\boldsymbol{a}_t|\boldsymbol{s}_t), \text{ if } \boldsymbol{s}_{t+1} \text{ is not terminal}, \\ r(\boldsymbol{y}|\boldsymbol{x}) + \beta \log \pi_{\text{ref}}(\boldsymbol{a}_t|\boldsymbol{s}_t) \text{ if } \boldsymbol{s}_{t+1} \text{ is terminal}. \end{cases}$$

The max-entropy RL formulation derives the optimal value function V^* and Q-function Q^* as:

$$Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t) = r'(\boldsymbol{s}_t, \boldsymbol{a}_t) + V^*(\boldsymbol{s}_{t+1}),$$
$$V^*(\boldsymbol{s}_t) = \log \sum_{\boldsymbol{a}} \exp(Q^*(\boldsymbol{s}_t, \boldsymbol{a})), \text{when } t \leq T.$$

Thus, the optimal policy π^* is derived as:

$$\beta \log \pi^*(\boldsymbol{a}_t|\boldsymbol{s}_t) = Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t) - V^*(\boldsymbol{s}_t),$$
$$\Rightarrow \quad \beta \log \frac{\pi^*(\boldsymbol{a}_t|\boldsymbol{s}_t)}{\pi_{\text{ref}}(\boldsymbol{a}_t|\boldsymbol{s}_t)} = V^*(\boldsymbol{s}_{t+1}) - V^*(\boldsymbol{s}_t).$$

This motivates us to use a DPO policy to derive the partial sum of the reward to formulate the progress reward R_g for a partial response $\boldsymbol{y}_{1:t+1}$:

$$\sum_{k=0}^{t} \beta \log \frac{\pi^*(\boldsymbol{a}_k|\boldsymbol{s}_k)}{\pi_{\text{ref}}(\boldsymbol{a}_k|\boldsymbol{s}_k)} = V^*(\boldsymbol{s}_{t+1}) - V^*(\boldsymbol{s}_0),$$
$$\Rightarrow \quad R_g(\boldsymbol{y}_{1:t+1}) = \sum_{k=0}^{t} w_k \log \frac{\pi^*(y_{k+1}|\boldsymbol{x}, \boldsymbol{y}_{1:k})}{\pi_{\text{ref}}(y_{k+1}|\boldsymbol{x}, \boldsymbol{y}_{1:k})},$$

where $\boldsymbol{y}_{1:k} = y_1, ..., y_k$, $w_k = \frac{1}{|\boldsymbol{y}_{1:k}|}$ is the weight for each sentence-level log-likelihood ratio, and $|\boldsymbol{y}_{1:k}|$ is the number of tokens in $\boldsymbol{y}_{1:k}$.

4.3.2 Attribution Progress Reward

We employ two citation metrics used in prior work (Gao et al., 2023b), i.e., citation recall and precision, for attribution progress reward R_a.

Specifically, citation recall measures the percentage of sentences in the partial response $\boldsymbol{y}_{1:t+1}$ that can be supported by the corresponding cited passages. We employ an NLI model (Honovich et al., 2022) to examine whether the cited passages can entail the model response. For each sentence y_i ($1 \leq i \leq t+1$), we concatenate the cited passages in $\mathcal{C}_i$ as premise and regard the generated sentence y_i as hypothesis for the NLI model. If the premise entails the hypothesis, we set the citation recall as 1, and 0 otherwise. Citation precision evaluates the percentage of citations that support the corresponding sentence. We use the same NLI model above to calculate the precision score. For each citation $c_{i,j}$, its precision score is set to 1 if (1) all citations in $\mathcal{C}_i$ entail the generated sentence y_i and (2) $\mathcal{C}_i \setminus \{c_{i,j}\}$ does not entail the sentence y_i. Otherwise, the precision score is set to 0. We compute the precision score for each citation (0 or 1) and average over all citations. Finally, we calculate F1 score as the attribution progress reward $R_a(\boldsymbol{y}_{1:t+1}, \mathcal{C}_1, ..., \mathcal{C}_{t+1})$ to provide a balanced attribution quality measure between the generated sentences and cited passages.

5 Experiments

5.1 Experimental Setup

Datasets. For evaluation, we use the ALCE benchmark (Gao et al., 2023b), which consists of three datasets: (1) **ASQA** (Stelmakh et al., 2022), a long-form QA dataset containing ambiguous questions that require multiple answers to cover different aspects; (2) **QAMPARI** (Amouyal et al., 2022), a factoid QA dataset where the answer to each question is a list of entities drawn from different passages; (3) **ELI5** (Fan et al., 2019), a long-form QA dataset containing how/why/what questions. For ASQA and QAMPARI, most questions can be answered by Wikipedia, thus we adopt the 2018/12/20 Wikipedia snapshot as the corpus. For ELI5, since its questions are diverse in topics, we use Sphere (Piktus et al., 2021), a filtered version of Common Crawl, as the corpus. Following Gao et al. (2023b), we adopt GTR (Ni et al., 2022) for Wikipedia and BM25 (Robertson et al., 2009) for Sphere to retrieve the top 100 passages as the corpus for each question. See Appendix A for more details.

Evaluation Metrics. We use the evaluation metrics in the original ALCE benchmark. To evaluate the correctness of the output, we use **Exact Match (EM) Recall** for ASQA, **Recall-5** for QAMPARI, and **Claim Recall** for ELI5, for measuring the percentage of gold answers (key information pieces) in the output. We further compute **Precision** as a correctness metric for the QAMPARI dataset, measuring the percentage of generated answers that are correct. To evaluate the citation quality of the output, we compute **Citation Recall**, which measures the percentage of sentences in the output that can be entailed from their cited passages, and **Citation Precision**, which measures the percentage of citations that can help support the output sentences.

Baselines. We compare our approach to the following baselines based on ChatGPT and GPT-4o:

• **Vanilla RAG** directly instructs the model to generate responses and cite accordingly based on the given top 5 passages. We use in-context learning (Brown et al., 2020) with two demonstrations.

• **Summary/Snippet RAG** provides summaries or snippets of passages instead of the full text. The model will generate responses with citations based on the top 10 passage summaries or snippets.

• **Interact** allows the model to further access the full text of certain passages for the Summary/Snippet RAG method. The model can propose an action "`Check: Document [1][2]`" to obtain the full text of the corresponding documents.

• **Inline Search** allows the model to request an action "`Search: {query}`" to retrieve the most relevant passage from the top 100 passages. This method is similar to our approach which serves as a direct comparison.

• **ReRank** randomly samples four responses for each question and select the best one based on the citation recall metric.

The above baselines have been employed and evaluated on the original ALCE benchmark, as reported in (Gao et al., 2023b). Besides, we compare our approach to existing work on attributed text generation. **FG-Reward** (Huang et al., 2024) proposes to use fine-grained rewards as training signals to fine-tune LLaMA-2-7B (Touvron et al., 2023) to generate attributed responses. **VTG** (Sun et al., 2024) guides the LLM (i.e., `text-davinci-003`) using an evolving memory and a two-tier verifier. **APO** (Li et al., 2024) curates a preference dataset and uses preference optimization to fine-tune LLaMA-2-13B. Note that our model directly performs inference on the test sets of the three evaluation datasets without performing any fine-tuning.

Implementation Details. We use LLaMA-3.1-8B-Instruct and GPT-4o as our policy models to assess the performance of our approach. For the reward models, we adopt a DPO model, i.e., Llama-3-8B-SFR-Iterative-DPO-R[2], to compute the generation progress reward and an NLI model, i.e., T5-XXL-TRUE-NLI-Mixture (Honovich et al., 2022), to compute the attribution progress reward. For each search query, we retrieve top 3 passages from the corpus as the candidate references $\mathcal{D}_t$. In UCT algorithm (Eq. 1), the weight w is set to 0.2. For SG-MCTS, we expand three child nodes for each parent node and set the maximum number of reflection steps to 10, the maximum tree layer to 6, and the maximum iteration of MCTS to 30.

5.2 Main Results

Table 1 shows the results of our method and baselines across three datasets.

Firstly, it can be observed that the three retrieval-augmented generation (RAG) methods fall short compared to more recent models, while using summaries or snippets improves correctness. This improvement comes at the expense of citation quality as the passage information is highly compressed. ReRank leads to consistent improvements in citation quality across three datasets. As a direct comparison, Inline Search exhibits worse performance compared to other prompting-based baselines. This is due to simply proposing search queries without considering evidence quality.

Secondly, by fine-tuning an LLM on supervised training data with annotated citations, FG-Reward and APO show increased citation quality in both

[2]https://huggingface.co/Salesforce/LLaMA-3-8B-SFR-Iterative-DPO-R

	ASQA			QAMPARI				ELI5		
	Correctness	Citation		Correctness		Citation		Correctness	Citation	
	EM Rec.	**Rec.**	**Prec.**	**Recall-5**	**Prec.**	**Rec.**	**Prec.**	**Claim Rec.**	**Rec.**	**Prec.**
ChatGPT										
Vanilla RAG	40.4	73.6	72.5	20.8	20.8	20.5	20.9	12.0	51.1	50.0
w/ ReRank	40.2	84.8	81.6	22.8	21.4	21.2	21.4	11.4	69.3	67.8
Summary RAG	43.3	68.9	61.8	23.6	21.2	23.6	25.7	12.5	51.5	48.2
w/ Interact	39.1	73.4	66.5	23.2	20.9	22.1	24.3	13.7	50.1	49.2
Snippet RAG	41.4	65.3	57.4	24.5	21.5	22.9	24.9	14.3	50.4	45.1
w/ Interact	41.2	64.5	57.7	22.4	20.8	21.6	23.1	13.3	47.8	45.2
Inline Search	32.4	58.3	58.2	17.2	20.4	14.9	14.9	13.4	45.6	43.7
GPT-4o										
Vanilla RAG	41.3	68.5	75.6	22.2	25.0	25.9	27.0	<u>20.3</u>	53.1	55.2
w/ ReRank	42.1	83.4	<u>82.3</u>	32.6	30.9	33.1	32.8	19.4	68.2	65.7
Summary RAG	46.5	70.2	67.2	36.2	34.0	36.2	39.8	18.7	64.4	63.9
w/ Interact	<u>48.1</u>	73.1	72.8	<u>38.2</u>	<u>37.1</u>	39.2	40.6	20.1	69.2	66.2
Snippet RAG	45.1	68.9	66.5	37.1	35.2	37.2	38.3	19.8	64.8	60.1
w/ Interact	45.2	67.8	66.7	37.2	34.5	38.7	39.5	19.9	68.1	65.1
Inline Search	40.3	65.7	66.9	27.8	27.2	19.4	23.8	12.5	50.2	53.3
FG-Reward	40.1	77.8	76.3	16.7	19.5	19.5	20.1	11.5	60.9	60.2
VTG	41.5	<u>86.7</u>	80.0	20.3	22.4	<u>43.5</u>	<u>46.9</u>	16.7	<u>82.6</u>	<u>71.6</u>
APO	40.5	72.8	69.6	15.4	20.6	17.5	19.3	13.5	26.0	24.5
Ours (LLaMA)	45.2	82.3	80.6	25.7	28.1	40.5	43.1	17.4	77.5	75.3
Ours (GPT-4o)	**50.1**	**89.5**	**87.1**	**45.2**	**41.9**	**50.6**	**52.8**	**25.9**	**85.6**	**80.2**

Table 1: Evaluation results on three datasets on attributed text generation. "Rec." and "Prec." are short for recall and precision. The **bold** and <u>underline</u> fonts denote the best and second best results in each dataset, respectively.

ASQA and ELI5 datasets. Besides, VTG employs a generation verifier and a memory verifier to assess the logical support of evidence, leading to strong citation quality (e.g., 86.7% citation recall in ASQA). However, fine-tuning the LLMs is constrained by the quality and quantity of the supervised training data where the supporting evidence requires substantial costs to link to correct sources. Moreover, these approaches still rely on auto-regressive generation, where any intermediate generation errors (e.g., false statements or inadequate citations) may result in problematic final responses.

Finally, our approach outperforms all baselines significantly across all three datasets. Think&Cite formulates attributed text generation as a multi-step reasoning problem and introduces a slow and deliberative thinking mode to search for the optimal solutions. Think&Cite leverages the self-reflection capability of LLMs to guide the tree expansion process. Besides, the proposed progress reward modeling can further provide comprehensive and reliable feedback to help the model explore better generated responses.

5.3 Further Analysis

We report further analysis of our method using GPT-4o on ASQA, as we have similar findings in the other datasets.

Method	Correctness	Citation	
	EM Rec.	**Rec.**	**Prec.**
Think&Cite	50.1	89.5	87.1
w/o SG-MCTS	42.1	78.2	75.0
w/o Reflection	46.5	83.6	80.3
w/o GP Reward	47.1	86.2	84.9
w/o AP Reward	46.7	81.3	80.4

Table 2: Ablation study in ASQA.

Ablation Study. To validate the effectiveness of our proposed framework, we conduct an ablation analysis of its key design elements. We design four variants: (1) *w/o SG-MCTS* removes self-guided MCTS and directly generates answers step by step; (2) *w/o Reflection* removes the reflection step and adopts the vanilla MCTS algorithm; (3) *w/o GP Reward* removes the generation progress reward R_g; and (4) *w/o AP Reward* removes the attribution progress reward R_a. We show the results in Table 2. All the variants perform worse than the original method, indicating the effectiveness of each component. Specifically, the performance of *w/o SG-MCTS* drops significantly, indicating that integrating search algorithms in attributed text generation is highly beneficial. Using vanilla MCTS (*w/o Reflection*) results in worse citation quality, due to the introduction of erroneous references without reflection on the retrieved results. Similarly, both

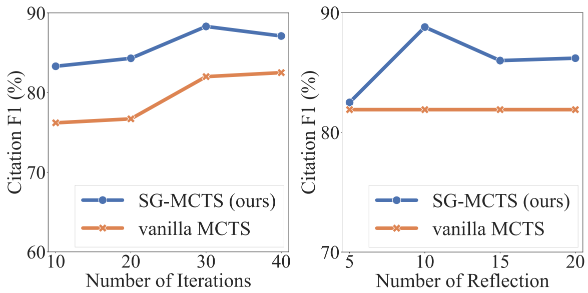

Figure 3: Results on ASQA *w.r.t.* the number of iterations (**left**) or the number of reflection steps (**right**).

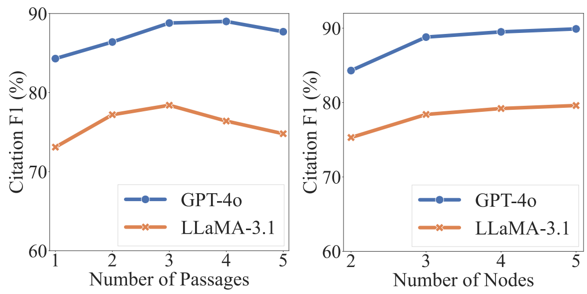

Figure 4: Results on ASQA *w.r.t.* the number of passages (**left**) or the number of expanded nodes (**right**).

w/o GP Reward and *w/o AP Reward* lead to worse performance, indicating that both generation and citation quality check are critical.

Reflection vs MCTS Iteration. In each iteration, SG-MCTS employs four key steps and reflection to improve the quality of intermediate states in the expansion phase by criticizing and refining error queries. To examine the effectiveness of reflection, we compare the performance between increasing the maximum number of MCTS iterations and reflection steps. We first vary the maximum number of MCTS iterations in $\{10, 20, 30, 40\}$ and fix the maximum number of reflection steps as 10. Similarly, we also vary the maximum number of reflection steps in $\{5, 10, 15, 20\}$ and fix the maximum number of iterations as 30. We present the F1 score based on the citation recall and precision in Figure 3. The figure shows that increasing the number of iterations and reflection steps enhance the task performance. This is expected, as more extensive exploration raises the probability of finding the correct generation. However, more reflection steps make the model "overthinks", introducing noise and resulting in performance degradation. SG-MCTS outperforms vanilla MCTS without reflection, since incorrect retrieval is likely to exist in parent nodes, causing the reasoning process of child nodes to continue along the wrong path. The reflection step refines flawed retrieval resulting from inadequate queries, allowing subsequent exploration to proceed more accurately.

Hyper-parameter Analysis. There are two critical hyper-parameters for correctness and citation quality: the number of retrieved passages $\mathcal{D}_t$ for each query q_t, and the number of expanded child nodes s_{t+1} in tree search. As illustrated in Figure 4, citation quality can be improved initially with increasing number of retrieved passages. However, further increase beyond a certain threshold results in worse performance, mainly because incorporating more passages introduces noise, negatively impacting the credibility of the generated content. Besides, we observe a consistent improvement when increasing the number of expanded nodes, although the improvement stabilizes later. Since expanding more nodes leads to higher computational costs, we sample three child nodes per parent node.

5.4 Case Study

To facilitate understanding of the entire workflow of our approach, we conduct a qualitative analysis in ASQA. We present an example in Figure 5 in Appendix D. Throughout the search process, the LLM considers the input question as the root node and incrementally expands the search tree to reach the terminal state. As shown in the example, the model first generates the query (i.e., "*Gunnison natural place location nearby*") to retrieve passages. Since the passages do not contain valid information to answer the question, the model reflects and proposes a new query (i.e., "*Gunnison natural place attractions*") for retrieval. Based on the retrieved passages, the model generates the sentence and cites the second and third passages (i.e., "`[2][3]`"). By following the multi-step generation process, the model can deliberate on the topic and output reliable content with accurate citations.

6 Conclusion

In this work, we proposed Think&Cite, a novel framework integrating tree search for attributed text generation. Think&Cite built upon an iterative think-verbalize-cite generation paradigm. To enhance the generation process, we proposed self-guided Monte Carlo Tree Search, which leveraged the self-reflection capability of LLMs to criticize and refine the intermediate states of MCTS to guide tree expansion. Moreover, we proposed progress reward modeling to measure the progress of tree search and to provide reliable feedback. Extensive experiments on three datasets showed that our proposed Think&Cite outperforms traditional prompting and fine-tuning methods.

Limitations

The scope of our experiments is constrained by the substantial computational cost of tree-based search methods. Future work can explore a broader range of attributed text generation datasets. In our model, Monte Carlo tree search is employed for self-guided generation. Future work can explore additional search algorithms to evaluate the generality and robustness of our proposed framework.

Acknowledgments

This work is fully supported by the Advanced Research and Technology Innovation Centre (ARTIC), the National University of Singapore under Grant (project number: ELDT-RP1).

References

Samuel Joseph Amouyal, Ohad Rubin, Ori Yoran, Tomer Wolfson, Jonathan Herzig, and Jonathan Berant. 2022. QAMPARI: An open-domain question answering benchmark for questions with many answers from multiple paragraphs. *CoRR*, abs/2205.12665.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Bradley C. A. Brown, Jordan Juravsky, Ryan Saul Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *CoRR*, abs/2407.21787.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Cameron Browne, Edward Jack Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez Liebana, Spyridon Samothrakis, and Simon Colton. 2012. A survey of Monte Carlo tree search methods. *IEEE Trans. Comput. Intell. AI Games*, 4(1):1–43.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. Fireact: Toward language agent fine-tuning. *CoRR*, abs/2310.05915.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: long form question answering. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3558–3567. Association for Computational Linguistics.

Constanza Fierro, Reinald Kim Amplayo, Fantine Huot, Nicola De Cao, Joshua Maynez, Shashi Narayan, and Mirella Lapata. 2024. Learning to plan and generate text with citations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 11397–11417. Association for Computational Linguistics.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Y. Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. 2023a. RARR: researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 16477–16508. Association for Computational Linguistics.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023b. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 6465–6488. Association for Computational Linguistics.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.*, 4(2):100–107.

Or Honovich, Roee Aharoni, Jonathan Herzig, Hagai Taitelbaum, Doron Kukliansky, Vered Cohen, Thomas Scialom, Idan Szpektor, Avinatan Hassidim, and Yossi Matias. 2022. TRUE: re-evaluating factual consistency evaluation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 3905–3920. Association for Computational Linguistics.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron C. Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-STaR: Training verifiers for self-taught reasoners. *CoRR*, abs/2402.06457.

Chengyu Huang, Zeqiu Wu, Yushi Hu, and Wenya Wang. 2024. Training language models to generate text with citations via fine-grained rewards. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 2926–2949. Association for Computational Linguistics.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *CoRR*, abs/2311.05232.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12):248:1–248:38.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 7969–7992. Association for Computational Linguistics.

Daniel Kahneman. 2011. Thinking, fast and slow. *Farrar, Straus and Giroux*.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based Monte-Carlo planning. In *Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings*, volume 4212 of *Lecture Notes in Computer Science*, pages 282–293. Springer.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Dongfang Li, Zetian Sun, Baotian Hu, Zhenyu Liu, Xinshuo Hu, Xuebo Liu, and Min Zhang. 2024. Improving attributed text generation of large language models via preference learning. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 5079–5101. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Nelson F. Liu, Tianyi Zhang, and Percy Liang. 2023. Evaluating verifiability in generative search engines. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 7001–7025. Association for Computational Linguistics.

Shashi Narayan, Joshua Maynez, Reinald Kim Amplayo, Kuzman Ganchev, Annie Louis, Fantine Huot, Anders Sandholm, Dipanjan Das, and Mirella Lapata. 2023. Conditional generation with a question-answering blueprint. *Trans. Assoc. Comput. Linguistics*, 11:974–996.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y. Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. 2022. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 9844–9855. Association for Computational Linguistics.

Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Dmytro Okhonko, Samuel Broscheit, Gautier Izacard, Patrick S. H. Lewis, Barlas Oguz, Edouard Grave, Wen-tau Yih, and Sebastian Riedel. 2021. The web is your oyster - knowledge-intensive NLP against a very large web corpus. *CoRR*, abs/2112.09924.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 9248–9274. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nat.*, 529(7587):484–489.

Aviv Slobodkin, Eran Hirsch, Arie Cattan, Tal Schuster, and Ido Dagan. 2024. Attribute first, then generate: Locally-attributable grounded text generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3309–3344. Association for Computational Linguistics.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, abs/2408.03314.

Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. 2022. ASQA: factoid questions meet long-form answers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 8273–8288. Association for Computational Linguistics.

Hao Sun, Hengyi Cai, Bo Wang, Yingyan Hou, Xiaochi Wei, Shuaiqiang Wang, Yan Zhang, and Dawei Yin. 2024. Towards verifiable text generation with evolving memory and self-reflection. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 8211–8227. Association for Computational Linguistics.

Richard Sutton. 2019. The bitter lesson. *Incomplete Ideas (blog)*, 13(1):38.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Chaojie Wang, Yanchen Deng, Zhiyi Lv, Zeng Liang, Jujie He, Shuicheng Yan, and Bo An. 2024. Q*: Improving multi-step reasoning for LLMs with deliberative planning. *CoRR*, abs/2406.14283.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Hai Ye and Hwee Tou Ng. 2024. Preference-guided reflective sampling for aligning language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21646–21668, Miami, Florida, USA. Association for Computational Linguistics.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. 2021. Mastering Atari games with limited data. In *NeurIPS*, pages 25476–25488.

Xiao Yu, Baolin Peng, Vineeth Vajipey, Hao Cheng, Michel Galley, Jianfeng Gao, and Zhou Yu. 2024. Improving autonomous AI agents with reflective tree search and self-learning. *arXiv preprint arXiv:2410.02052*.

Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, et al. 2024. LLaMA-Berry: Pairwise optimization for O1-like Olympiad-Level mathematical reasoning. *arXiv preprint arXiv:2410.02884*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023. Siren's song in the AI ocean: A survey on hallucination in large language models. *CoRR*, abs/2309.01219.

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

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024. Language agent tree search unifies reasoning, acting, and planning in language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

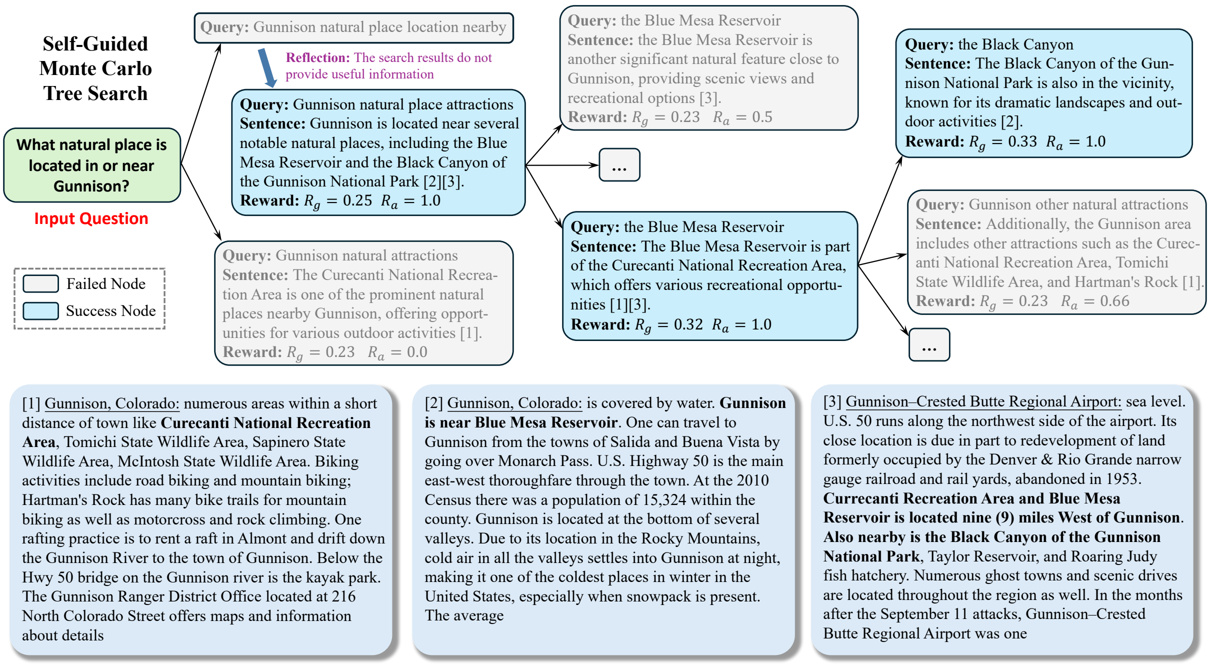

Figure 5: A qualitative example from ASQA showing the attributed generation process of Think&Cite.

Appendix

A Datasets

We evaluate our approach on the ALCE benchmark (Gao et al., 2023b) consisting of three datasets. Specifically, the ASQA dataset (Stelmakh et al., 2022) contains 948 questions, where the answers can be found from Wikipedia; the QAMPARI dataset (Amouyal et al., 2022) contains 1,000 questions based on Wikipedia; and the ELI5 dataset (Fan et al., 2019) includes 1,000 questions, where the answers can be found from Sphere (Piktus et al., 2021). The details of these three datasets are given in Table 3. Following the setting of the ALCE benchmark (Gao et al., 2023b), we divide the retrieval corpus into 100-word passages to enable a fair and consistent comparison to baselines. Besides, 100-word chunks provide more precise evidence for the generated content, make it easier for humans to verify, and do not introduce too much irrelevant information.

Dataset	Corpus (#Passages)	Question Type
ASQA	Wikipedia (21M)	Factoid
QAMPARI	Wikipedia (21M)	Factoid (list)
ELI5	Sphere (899M)	Why/How/What

Table 3: Details of the three evaluation datasets.

Method	NI	TT	NT	Recall	Precision
FG-Reward	-	60	1229	77.8	76.3
VTG	-	76	1057	86.7	80.0
Think&Cite (Ours)	10	67	1149	83.5	81.6
	20	78	1238	85.7	81.8
	30	96	1547	89.5	87.1
	40	103	1720	87.0	85.4

Table 4: Comparison of time costs per question and the citation recall and precision. Here, **NI** represents the maximum number of MCTS iterations per question, **TT** and **NT** represent the total inference time (second) and number of generated tokens per question, respectively.

B Analysis of Computational Expenses

We conduct further analysis of computational costs using our model (incorporating GPT-4o) concerning different numbers of MCTS iterations on the ASQA dataset. First, we observe that using our model, the average time taken and the average number of tokens increase when the number of MCTS iterations increases. Citation performance also improves (except when number of iterations = 40). These results are in line with recent results reported in the literature on test-time scaling. Second, because FG-Reward adopts rejection sampling (which involves sampling multiple times) to generate text and VTG uses the whole corpus to verify citation correctness, their total time costs are also high to some extent.

Your objective is to write an accurate, engaging, and concise answer for the given question using only the provided search results and cite them properly. Use an unbiased and journalistic tone. Always cite for any factual claim. You can use four operations:
(1) """Search: key words""" which requests some key words to retrieve the most relevant documents based on the search and generation history.
(2) """Reflexion: thoughts""" which reflects on the search results if not obtaining valid documents for answering the question.
(3) """Output: sentences""" which outputs a long-form answer based on retrieved documents. In the answer, cite properly by using [1][2][3]. Cite at least one document and at most three documents in each sentence. If multiple documents support the sentence, only cite a minimum sufficient subset of the documents.
(4) """End""" which ends the task of question answering.
Here are some example:

Question: Who set the record for longest field goal?
Search: record for longest field goal
Document [1] (Title: David Akers) record for the longest field goal by kicking a 63-yard field goal off the crossbar ...
Document [2] (Title: Dirk Borgognone) Dirk Borgognone Dirk Ronald Borgognone (born January 9, 1968) is a former National ...
Document [3] (Title: Russell Erxleben) against the University of Colorado. Despite early troubles in the game, including ...
Reflexion: The search results do not provide valid information to answer the question. It might be better to search the longest field goal in NFL.
Search: record for longest field goal NFL
Document [4] (Title: Field goal) toward its own end. The longest field goal kick in NFL history is 64 yards, a record set ...
Document [5] (Title: Field goal) both end zones) is only 66 yards. Scaccia, while playing indoor football, attempted a 64-yard ...
Document [6] (Title: Field goal) this accomplishment is not the official record. All of the above kicks were successful ...
Output: The record for the longest field goal in an NFL game was set by Matt Prater at 64 yards [4].
Search: record for longest field goal at any level college
Document [7] (Title: Field goal) toward its own end. The longest field goal kick in NFL history is 64 yards, a record set by ...
Document [8] (Title: Field goal range) 35 and 40 yard lines (closer in a crosswind) often will go for the more risky ...
Document [9] (Title: Field goal range) NFL and have been banned from NCAA since 1989) is 68 yards held by Fabrizio Scaccia ...
Output: But the record for the longest field goal at any level was 69 yards, kicked by collegiate kicker Ove Johansson in a 1976 Abilene Christian University football game against East Texas State University [8].
End

<Remaining Three Demonstrations>

Now, following the example's output format to solve this new question.

Question: <insert the input question>

Table 5: Prompt of attributed text generation in ASQA. The blue text denotes the intention description, and the green text denotes the generation demonstration.

C Prompts

We instruct the LLM to perform self-guided tree search for attributed text generation in a few-shot manner. The in-context learning prompts for the ASQA, QAMPARI, and ELI5 datasets are presented in Table 5, Table 6, and Table 7 respectively. In these prompts, we first define four operations for the LLM in the iterative think-verbalize-cite paradigm. Then, we construct four in-context examples, which are selected to be consistent with the ALCE benchmark (Gao et al., 2023b).

D Case Study

We present an example from ASQA in Figure 5.

Your objective is to write an accurate, engaging, and concise answer for the given question using only the provided search results and cite them properly. Use an unbiased and journalistic tone. Always cite for any factual claim. You can use four operations:
(1) """Search: key words""" which requests some key words to retrieve the most relevant documents based on the search and generation history.
(2) """Reflexion: thoughts""" which reflects on the search results if not obtaining valid documents for answering the question.
(3) """Output: sentences""" which outputs a long-form answer based on retrieved documents. In the answer, cite properly by using [1][2][3]. Cite at least one document and at most three documents in each sentence. If multiple documents support the sentence, only cite a minimum sufficient subset of the documents.
(4) """End""" which ends the task of question answering.
Here are some example:

Question: Which books were written by Nevil Shute?
Search: Nevil Shute books theme social barriers
Document [1] (Title: Nevil Shute) early stages. My congratulations." His celebrity as a writer caused the Ministry ...
Document [2] (Title: Nevil Shute) theme is the bridging of social barriers such as class ("Lonely Road" and "Landfall"), race ...
Document [3] (Title: Nevil Shute) construction company, Airspeed Ltd. His popularity grew slowly with each novel, but he became much more famous ...
Output: Beyond the Black Stump [2], Lonely Road [2], The Chequer Board [2], In the Wet [2], Trustee from the Toolroom [2], Round the Bend [2].
Search: Nevil Shute books simple readable style
Document [4] (Nevil Shute) construction company, Airspeed Ltd. His popularity grew slowly with each novel, but he became much more famous ...
Document [5] (The Chequer Board) the Burmese people", both of which are central to the book's story. Shute was concerned that sales of the book ...
Document [6] (In the Wet) had used the idea of multiple votes for merit in his short story "The Curious Republic of Gondour". ...
Reflexion: The search results do not provide any useful information to answer the question. It might be better to search Nevil Shute books in 1950s.
Search: Nevil Shute books 1950s
Document [7] (Nevil Shute) early stages. My congratulations." His celebrity as a writer caused the Ministry of Information to send him to the ...
Document [8] (Nevil Shute) theme is the bridging of social barriers such as class ("Lonely Road" and "Landfall"), race ("The Chequer Board") ...
Document [9] (Nevil Shute) construction company, Airspeed Ltd. His popularity grew slowly with each novel, but he became much more famous ...
Output: Marazan [7], Stephen Morris [7].
End

<Remaining Three Demonstrations>

Now, following the example's output format to solve this new question.

Question: <insert the input question>

Table 6: Prompt of attributed text generation in QAMPARI. The blue text denotes the intention description, and the green text denotes the generation demonstration.

Your objective is to write an accurate, engaging, and concise answer for the given question using only the provided search results and cite them properly. Use an unbiased and journalistic tone. Always cite for any factual claim. You can use four operations:
(1) """Search: key words""" which requests some key words to retrieve the most relevant documents based on the search and generation history.
(2) """Reflexion: thoughts""" which reflects on the search results if not obtaining valid documents for answering the question.
(3) """Output: sentences""" which outputs a long-form answer based on retrieved documents. In the answer, cite properly by using [1][2][3]. Cite at least one document and at most three documents in each sentence. If multiple documents support the sentence, only cite a minimum sufficient subset of the documents.
(4) """End""" which ends the task of question answering.
Here are some example:

Question: What causes Bi-polar disorder?
Search: What is Bi-polar disorder?
Document [1] (Title: Bipolar II disorder) setting boundaries for one's perceptions and behaviors. Studies indicate that the following ...
Document [2] (Title: Biology of bipolar disorder) located in these regions have been hypothesized as a mechanism for bipolar ...
Document [3] (Title: Biology of bipolar disorder) regardless of polarity. Inflammation has been consistently reported in bipolar ...
Reflexion: The search results do not provide valid information to answer the question. It might be better to search the symptoms of Bi-polar disorder.
Search: What are the symptoms of Bi-polar disorder and how long do they last?
Document [4] (Title: Bi-polar disorder | definition of Bi-polar disorder by Medical dictionary) bi-polar disorder | definition of bi-polar ...
Document [5] (Title: Mania and Bi-Polar) can go from depressed to ž01csuper happyž01d all in one day, or even in a few days, does ...
Document [6] (Title: For Individuals — Adam Schwartz) For Individuals ž014 Adam Schwartz The information is extensive and covers a ...
Output: Bipolar disorder is an emotional disorder that causes extreme mood swings between excitement and depression [4]. The spectrum of mood swing may span from days to months [5].
Search: What could cuase Bi-polar disorder?
Document [7] (Title: Bi-Polar disorder) Bi-Polar disorder Bi-polar is generally a cyclic disease where individuals display depressive ...
Document [8] (Title: Depression Bi-polar Disorder Symptoms 2019 | Win Over Depression) Depression Bi-polar Disorder Symptoms 2019 ...
Document [9] (Title: Mental disorder) may include parenting factors including parental rejection, lack of parental warmth ...
Output: We are still not certain of the exact factors that cause such disorder, but genetics is considered a major factor [7].
End

<Remaining Three Demonstrations>

Now, following the example's output format to solve this new question.

Question: <insert the input question>

Table 7: Prompt of attributed text generation in ELI5. The blue text denotes the intention description, and the green text denotes the generation demonstration.