STOC-TOT: Stochastic Tree-of-Thought with Constrained Decoding for Complex Reasoning in Multi-Hop Question Answering

Anonymous EMNLP submission

Abstract

Multi-hop question answering (MHQA) requires a model to retrieve and integrate information from multiple passages to answer a complex question. Recent systems leverage the power of large language models and integrate evidence retrieval with reasoning prompts (e.g., chain-of-thought reasoning) for the MHQA task. However, the complexities in the question types (bridge v.s. comparison questions) and the reasoning types (sequential v.s. parallel reasonings) require more novel and fine-grained prompting methods to enhance the performance of MHQA under the zero-shot setting. In this paper, we propose STOC-TOT, a stochastic tree-of-thought reasoning prompting method with constrained decoding for MHQA and conduct a detailed comparison with other reasoning prompts on different question types and reasoning types. Specifically, we construct a tree-like reasoning structure by prompting the model to break down the original question into smaller sub-questions to form different reasoning paths. In addition, we prompt the model to provide a probability estimation for each reasoning path at each reasoning step. At answer time, we conduct constrained decoding on the model to generate more grounded answers and reduce hallucination. Experiments comparing STOC-TOT with on two MHQA datasets and five large language models showed that STOC-TOT outperforms other reasoning prompts by a significant margin.

1 Introduction

Question answering (QA) is a fundamental task in natural language processing (NLP) that involves designing systems capable of understanding human language questions and providing accurate and relevant answers. With the recent advancement of large language models (LLMs) that demonstrated superior reasoning ability (Brown et al., 2020), researchers have been focusing more on complex QA tasks, such as multi-hop question answering

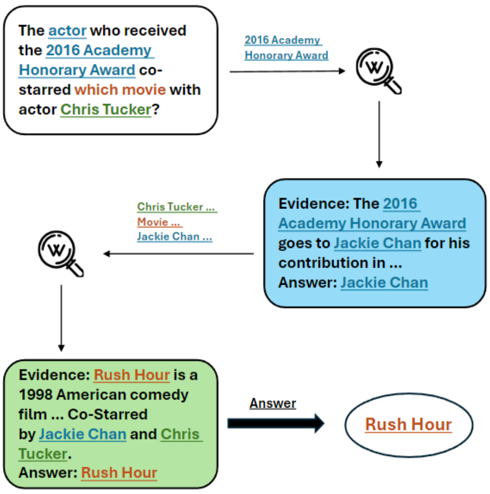


Figure 1: An example of the MHQA question. This question has two hops that require the model to reason about before answering the final question.

(MHQA). MHQA is more challenging as it requires models to understand complicated questions, perform multiple reasoning steps, and gather evidence across documents. Figure 1 shows an example of a two-hop MHQA question. To answer that question in Figure 1, the QA model needs to first figure out who is the actor that received the 2016 Academy Honorary Award. Then based on the answer to the previous question, the QA model needs to further answer a second question about which movie the actor co-starred with Chris Tucker.

State-of-the-art methods for MHQA are fully-supervised methods that often follow a retrieve-and-read framework, including a passage retrieving module that gathers relative evidence from documents and a reading comprehension module to reason about the evidence (Zhu et al., 2021; Li et al., 2022). Other methods include beam-search (Zhang et al., 2023) and label-smoothing (Yin et al., 2023). However, these methods often require extensive pre-training or fine-tuning and do not generalize well to other datasets.

Large language models (LLMs), on the other hand, show remarkable reasoning ability and rich knowledge of general-domain questions. Many LLMs can answer simple and straightforward questions that do not require complex reasoning without any supervision involved but often fail to deal with complex questions requiring multiple reasoning steps. To tackle the problem, researchers have developed many prompting techniques to improve LLM's reasoning ability, such as chain-of-thought (CoT) (Wei et al., 2022), self-consistency CoT (Sc-CoT) (Wang et al., 2023), and tree-of-thought (ToT) prompting (Yao et al., 2023a).

CoT has been shown effective across tasks requiring extensive, step-by-step reasoning, such as math calculation and reading comprehension. However, there could be various possible reasoning paths for many complex multi-hop questions, and CoT models cannot "turn back" when they have made a mistake along their reasoning paths. Sc-CoT further improves on CoT by proposing different chains of thought, thus expanding the reasoning space. However, there is no local reasoning expansion within each chain, and the "majority voting" strategy often fails in open-domain tasks where the output space is unlimited. ToT, designed to maintain different reasoning paths along its reasoning process, is more suitable for dealing with complex question types. However, the intermediate reasoning steps in NLP generation tasks are much less constrained and require more than a simple rule-based evaluation. The complexities in the question types (bridge v.s. comparison questions in Table 1), as well as the reasoning types (sequential v.s. parallel reasonings in Table 2), require more novel and fine-grained prompting methods to enhance the reasoning ability of LLMs.

To tackle the challenges and design a more reliable reasoning method for open-domain NLP tasks, we propose STOC-TOT, a stochastic ToT-based framework that instructs the model to generate different reasoning paths from the same question and assign probability scores to reasoning paths to effectively avoid reasoning dead-ends. To the best of our knowledge, our work is the first to adapt the tree-of-thought reasoning prompting to natural language tasks that require complex reasoning, such as MHQA. We provide an example overview of our framework in Figure 2. Specifically, we construct a tree-like reasoning structure by prompting the model to break down the original question into smaller sub-questions to form different reasoning paths. We evaluate the validity of each reasoning path on three levels of aspects and arrive at a model-given probability score. At answer time, we innovatively propose to use constrained decoding in the answering process to reduce hallucination by forcing the model to generate grounded answers from evidence and letting models give concise and exact answers. Ultimately, we arrive at the best answer by choosing the path with the highest aggregated probability score. Experiments on two benchmarking MHQA datasets demonstrate that STOC-TOT significantly improves the reasoning ability of LLMs in complex reasoning scenarios, especially with GPT-4, improving Exact Match accuracy by 7%, and F1 score by 7.8 points on the HotpotQA dataset over the original tree-of-thought prompting. Our contributions are as follows:

- We propose STOC-TOT, which constructs a stochastic reasoning tree in complex reasoning scenarios. We introduce stochastic estimations on different reasoning paths, which helps the model have a more reliable reasoning process than previous reasoning prompting methods.

- We innovatively propose to use constrained decoding in the answering process. This step reduces model hallucination by forcing the model to generate grounded answers from evidence and letting models give concise and exact answers.

- We evaluate the effectiveness of STOC-TOT by conducting experiments on two MHQA datasets. We observe substantial improvements over other reasoning prompting methods, with STOC-TOT surpassing all other selected reasoning prompting baselines on 5 tested models.

2 Related Work

Multi-Hop Question Answering Multi-hop Question Answering (MHQA) is a challenging task requiring models to reason over different evidence across documents to answer a complex multi-hop question. Many high-quality MHQA datasets have been developed, including HotpotQA (Yang et al., 2018), WikiHop (Welbl et al., 2018), MuSiQue (Trivedi et al., 2022), and others. Among these, HotpotQA is the task's most representative and widely used dataset. Previous state-of-the-art MHQA models often follow a two-stage pipeline: a retriever that extracts evidence from the documents, and a reader that reasons about the evidence

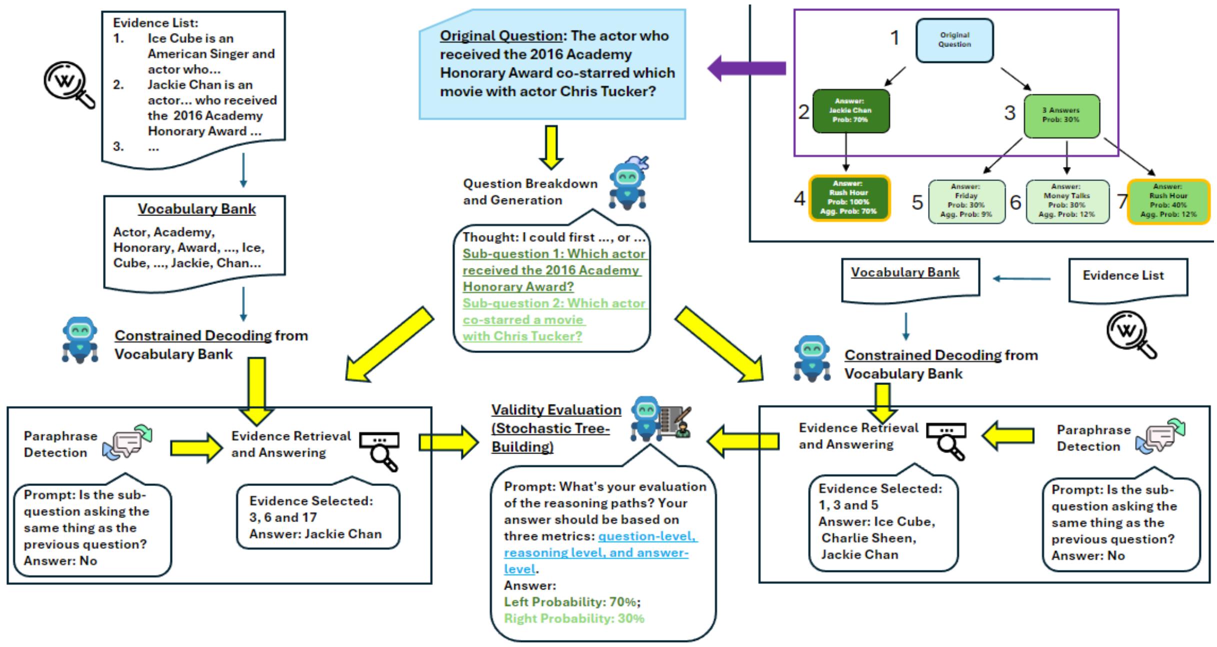


Figure 2: Overview of our framework, with the example in Figure 1. The top-right Corner shows the overall structure of the constructed tree, with each node's label on the left. **Darker green** in the nodes means a **higher evaluated probability of the reasoning path**. The original Question is colored in blue. We chose the first round of our tree-building process as an example in the purple block.

to arrive at an answer (Zhu et al., 2021; Li et al., 2022). Other methods include beam-search (Zhang et al., 2023) and label-smoothing (Yin et al., 2023). Some LLM-based frameworks (Yao et al., 2023b; Gou et al., 2024) were also evaluated on the task of MHQA, but their performance fell short compared with supervised methods.

Reasoning Prompting of LLMs Various prompt engineering methods have been developed (Wei et al., 2022; Wang et al., 2023; Yao et al., 2023a; Besta et al., 2024; Sel et al., 2024; Chen et al., 2023), aiming to improve large language models' reasoning ability across various tasks and domains. Chain-of-thought (CoT) prompting (Wei et al., 2022) prompts the large language models (LLMs) to divide their reasoning process into smaller steps when solving a question, forming a chain of thoughts. Chain-of-thought self-consistency prompting (Wang et al., 2023) improves on the CoT method by proposing different reasoning chains and ensembles on the final result. Tree-of-thought (ToT) prompting method (Yao et al., 2023a) actively maintains a tree of thoughts, where each thought is a coherent language sequence that serves as an intermediate step toward problem-solving. Graph-of-thought (Besta et al., 2024) further improves ToT by constructing a Directed Graph instead of a tree. LLMs can loop over a thought to refine it and aggregate thoughts or chains.

Constrained Decoding Constrained decoding is the technique that asks the models to generate outputs following a given set of rules. The most common way of conducting constrained generation uses beam search (Och and Ney, 2004) in decoding time. Before the LLM era, works on constrained decoding focused on task-specific sequence-to-sequence models that span across many fields, such as machine translation (Hokamp and Liu, 2017; Post and Vilar, 2018), named entity recognition (Lester et al., 2020), and dialogue generation (Balakrishnan et al., 2019). Recently, Microsoft introduced Guidance [1], which allows users of various large language models to control their outputs given a human-defined vocabulary or rules.

3 Method

3.1 Task Formation

Given a multi-hop question Q and background corpus of evidence P, the goal of our framework is to output the answer A to question Q, drawing its reasoning with the support of multiple evidence passages $p_1, p_2, \ldots$ retrieved from corpus P.

[1] https://github.com/guidance-ai/guidance

3.2 StoC-ToT Framework

For each of the questions Q, multiple reasoning lines and, thus, multiple ways of breaking down the question could exist. However, not every reasoning line would lead us to the right answer, and they take us to dead ends. To avoid such reasoning dead-ends, we build a stochastic reasoning tree to represent the possible reasoning lines and the probability of each reasoning line taking us to the right answer. We achieve this by proposing a self-interactive framework that automatically builds the reasoning tree given a multi-hop question. Figure 2 shows our framework with an example question.

In our reasoning process, we first prompt the model to propose different possible sub-questions to solve at each reasoning step. Each sub-question corresponds to one possible reasoning path and is presented as a node in the tree. We then ask the model to answer the generated sub-questions. To prevent hallucination and make the model more focused on the given question and evidence, we build a vocabulary bank using words from the evidence list and the original question and instruct the model to do constrained decoding from the vocabulary bank when generating its answers. After answering every sub-question generated from the same question in the previous reasoning level, we prompt the model to evaluate each reasoning path and estimate how likely the reasoning path would lead us to the right answer. This probability estimation would be assigned to the corresponding node in the tree. After the reasoning process finishes, each reasoning path would have an aggregated probability calculated from nodes along the path.

Formally, given a question Q, we instruct the model to generate sub-questions $q_1, q_2, ..., q_n$, and build a tree structure with the original question Q as the root node and each question q_i as subsequent nodes. The tree would expand as each sub-question q_i has its sub-question q_j, and the reasoning paths are thus represented as branches in the tree structure. From the original question Q and the evidence list $E = e_1, e_2, ..., e_n$, we build a vocabulary bank $V = [w_1, w_2, ..., w_n], w_i \in Q, w_j \in E$. We then prompt the model to generate their answer $a_1, a_2, ..., a_n$ using only $w_i \in V$. We describe the details of our framework below.

Example-Based Sub-Question Generation Our framework starts with the sub-question generation module, which generates sub-questions $q_1, q_2, ..., q_n$ using the question Q_g from the previous reasoning level. The sub-questions are generated based on both the model's reasoning ability and the model's semantic understanding of the question Q_g. An example is given in Figure 2, where the sub-questions from nodes 2 and 3 were generated using the question from node 1. However, we cannot guarantee that each sub-question asked is a good sub-question, and sometimes, the generated sub-question merely repeats the previous question. We introduce the paraphrase detection module and pass on the generated sub-questions to reduce redundancy and improve question quality.

Paraphrase Detection Answering repetitive questions often leads to low-quality answers and time-consuming steps. Following the sub-question generation module, we introduce the paraphrase detection module to reduce redundancy and improve question quality. In this module, we prompt the model and ask it to distinguish informative questions from questions that merely repeat what is already stated at the previous reasoning level. If a sub-question is a paraphrase, we instruct the model to stop generating sub-questions from the current question. In other words, we prune the low-quality sub-branch of the tree that could otherwise be generated. By pruning these branches, we effectively improve the efficiency of our framework.

Evidence Retrieval and Answering We then move on to answering the question after our paraphrase detection module. Our evidence retrieval and answering module focuses on retrieving evidence and generating answers to the given sub-question. We also pass in the full evidence list provided and prompt the model to give out an answer to the given sub-question. The evidence retrieval and answering module selects relative evidence from an evidence pool for each sub-question and uses words only from the vocabulary bank to generate its final answer. We will discuss details of constrained decoding in Section 3.3. The generated sub-answer and the answered sub-question are then passed on to the sub-question generation module at the next level to continue the reasoning process.

Validity Estimation Not each sub-question asked is a good sub-question, and not each reasoning path is reasonable. After every sub-question q_i generated from the same question Q_g has been answered, we prompt the model to provide a probability estimation p_i for each (q_i, a_i) pair. This probability is the model's evaluation of going down the correct reasoning path. Specifically, this probability is obtained by prompting the model to consider the following three aspects:

- Question Level: Is the question semantically clear and answerable?
- Reasoning Level: Is the reasoning line coherent when considering previous levels?
- Answer Level: Does the evidence fully support the answer to the question?

As shown in Figure 2, we conduct validity estimation for sub-questions and sub-answers in nodes 2 and 3 since the sub-questions were generated from the same question in node 1.

At the leaf node of our tree, we would have a final question q_f. along with a final answer A to the original question Q, and also an aggregated probability $p_{final} = \prod_i p_i$, with each p_i being the probability of the nodes along the reasoning path. We assign p_{final} to the leaf node, representing the aggregated probability of answer A being the correct answer to Q.

3.3 Constrained Decoding

One challenge for generative LLMs in the task of question answering is hallucination. LLMs often fail to pay attention to the golden evidence and hallucinate their own reference even when large amounts of evidence exist. To alleviate the problem of LLM halluscination during evidence selection and answer generation, we innovatively propose to use constrained decoding in the answering process to reduce hallucination by forcing the model to generate grounded answers from evidence and let models give concise and exact answers. As shown in Figure 2, we conduct constrained decoding by asking the model to generate words from the vocabulary bank, consisting of words taken only from the original question and the evidence list provided. More formally, we construct a vocabulary bank $V = w_1, w_2, ..., w_i$ from all words in the provided evidence sentences. We conduct a simple filtering by removing common English stop words. We then instruct the model's evidence retrieval and answering module to construct its answers using words only from the given vocabulary V.

Code-based Constrained Decoding For open-source LLMs (e.g., Llama), we build our logit processor at the decoding time. Specifically, for every word $w_j \notin V$, we manually set the score to negative infinity to prevent the model from generating them. Thus, every answer generated will only use words from the evidence list.

Prompt-based Constrained Decoding For closed-source LLMs (e.g., GPT models), since we do not have access to their decoding function, we had to instruct the GPT models using prompts to do constrained decoding. We provide our prompt template used in Appendix A.

4 Experimental Setup

Dataset We compare STOC-TOT with baseline methods on the HotpotQA dataset (Yang et al., 2018) and the MuSiQue dataset (Trivedi et al., 2022), both of which are widely used MHQA datasets across state-of-the-art MHQA baselines. The experiments are conducted under the distractor setting, where we provide the model with an evidence pool containing both golden and irrelevant evidence. The model needs to find the golden evidence to answer the question correctly. We randomly selected 200 examples from each dataset as our evaluation set.

Baselines We included three baselines:

- Vanilla Prompting with no examples provided. We only provide the model with questions and evidence and instruct it to output the answer.
- Chain-of-Thought (CoT) prompting (Wei et al., 2022) with a standard input-output (IO) prompt. We design the prompt with one in-context example, which presents the whole reasoning chain, including all intermediate steps.
- Tree-of-Thought prompting (Yao et al., 2023a) with slight modifications to adapt to the MHQA task. We largely followed the original framework and used majority voting on the reasoning lines to decide the final answer.

We recognize that there are LLM-based retrieval augmented generation frameworks (Yao et al., 2023b; Gou et al., 2024) that were also evaluated on HotpotQA. However, we excluded them from our baselines as they used outside knowledge bases, which are under a different testing scenario.

4.1 Implementation

We experiment with the baselines and our model utilizing five LLMs: GPT-3.5-turbo (Brown et al.,

Table 1: Performance comparison of StoC-ToT and baseline methods on the HotpotQA dataset.

Prompting Method	GPT3.5		GPT4		LLaMa2(13B)		LLaMa2(70B)		LLaMa3(8B)	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Zero-Shot Vanilla	34.0	45.0	51.0	65.0	25.5	36.5	30.5	41.0	27.5	40.7
Chain-of-Thought	35.5	47.3	52.0	66.8	30.5	42.5	33.5	45.0	32.5	44.6
Tree-of-Thought	36.5	49.5	55.0	68.5	29.5	41.3	35.5	47.3	30.5	37.5
StoC-ToT	**45.5**	**56.2**	**62.0**	**76.3**	**31.0**	**43.0**	**43.0**	**56.3**	**33.0**	**44.5**
w/o constrained decoding	40.5	53.5	59.5	73.0	31.0	43.0	40.5	53.5	32.0	44.3

Table 2: Performance comparison of StoC-ToT and baseline methods on the MusiQue dataset.

Prompting Method	GPT3.5		GPT4		LLaMa2(13B)		LLaMa3(8B)	
	EM	F1	EM	F1	EM	F1	EM	F1
Zero-Shot Vanilla	17.0	28.8	31.5	41.2	9.5	16.0	12.0	19.2
Chain-of-Thought	18.0	29.7	32.5	44.2	11.0	17.5	12.5	21.6
Tree-of-Thought	20.5	32.0	35.0	47.3	11.0	17.2	12.0	20.6
StoC-ToT	**26.5**	**38.0**	**42.0**	**55.3**	**11.5**	**18.0**	**14.5**	**22.0**
w/o constrained decoding	24.0	35.5	38.5	51.0	11.5	18.0	14.0	22.0

2020) and GPT-4(OpenAI et al., 2024) from OpenAI, LLaMa 2-13B (Touvron et al., 2023), LLaMa 2-70B, and LLaMa 3-8B from MetaAI. Due to the lengthy running time, LLaMa 2-70B was not tested on the MusiQue dataset. For all models, We set the temperature to 0.5, top_k to 1.0, and maximum number of iterations to 5.

4.2 Evaluation Metric

Following the metrics in (Yang et al., 2018), we use Exact Match and F1 score as two evaluation metric. For an answer a given by our framework, the Exact Match score equals 1 if the answer span matches the golden answer exactly and 0 otherwise. The F1 metric measures the average overlap between the prediction and ground truth answers.

5 Results

5.1 Overall Results

We compare StoC-ToT with LLM baselines on the HotpotQA dataset and the MusiQue dataset and present our results in Tables 1 and 2. The backbone LLMs in our experiments include GPT3.5, GPT4, Llama2-13B, Llama2-70B, and Llama3-8B. Due to time constraints, we only tested with Llama2-70B on the HotpotQA dataset. On the HotpotQA dataset, StoC-ToT attains an on-average increase in performance of over 6 % compared with vanilla prompting on GPT models, and the improvement goes up to 11% when we further implement StoC-ToT with constrained decoding. On the more challenging MusiQue dataset, we still see an increase in performance of StoC-ToT compared with the other baselines, most notably on GPT4, where we observe an 11.5% EM improvement (from 31.50 to 42.0).

Comparison with Tree-of-Thought StoC-ToT surpasses the original Tree-of-Thought prompting by 7% with the GPT4 model on both tested datasets. For LLMs with inferior reasoning ability, such as LLaMa2-8B, we still observe a performance improvement, even on the harder MusiQue dataset. These results suggest that StoC-ToT is more effective at forming and selecting reliable reasoning paths under complex reasoning scenarios.

Constrained Decoding Even though the LLM's reasoning ability can be improved by reasoning prompting, such techniques have little help in preventing hallucination. However, StoC-ToT implements constrained decoding, which makes the model much more grounded to evidence when answering the question, effectively addressing hallucination issues and improving the overall performance of our framework.

5.2 Ablation Study

Sensitivity to Demonstration Question Type We study the effect on StoC-ToT performance when different types of demonstration questions are provided in the prompt template. The HotPotQA dataset specified two types of questions. The "Bridge" question contains a "bridge entity" that connects the question and the final answer. In

Table 3: Performance of StoC-ToT with different prompt types on the HotpotQA dataset in terms of EM score. “Com" represents comparison questions, and “Bri" represents bridge questions.

Model Variant	**GPT3.5**		**GPT4**		**LLaMa2(13B)**		**LLaMa2(70B)**		**LLaMa3(8B)**	
Prompt/Question Type	**Com**	**Bri**	**Com**	**Bri**	**Com**	**Bri**	**Com**	**Bri**	**Com**	**Bri**
Prompt: Comparison	58.8	41.0	76.5	57.2	38.2	31.9	58.8	41.0	44.1	33.7
Prompt: Bridge	55.9	43.4	73.5	59.0	35.3	32.5	55.9	42.2	41.2	34.9

Table 4: Question Type Examples. On the left side, the bridging entity is highlighted in red, and the final question is highlighted in orange. On the right side, entities that are being compared are highlighted in blue.

Bridge Question	**Comparison Question**
What distinction is held by the former NBA player who was a member of the Charlotte Hornets during their 1992-93 season and was head coach for the WNBA team Charlotte Sting?	Were Scott Derrickson and Ed Wood of the same nationality?

Table 5: Reasoning Type Examples. On the left side, the entity in red needs to be found before solving the question in orange. On the right side, questions with parallel reasoning contain parts (highlighted in blue) that can be solved in arbitrary order.

Sequential Reasoning	**Parallel Reasoning**
The football manager who recruited David Beckham managed Manchester United during what timeframe?	What distinction is held by the former NBA player who was a member of the Charlotte Hornets during their 1992-93 season and was head coach for the WNBA team Charlotte Sting?

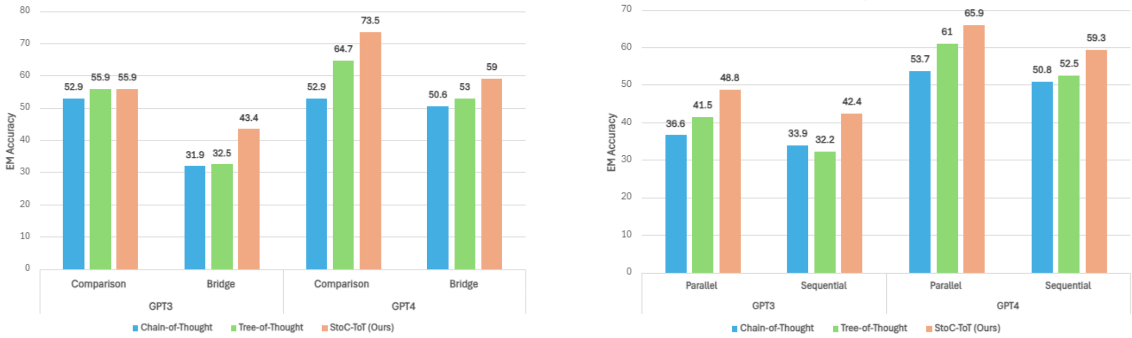
contrast, the "Comparison" question requires the model to compare two entities of the same type. Of the 200 questions in our evaluation set, 34 are comparison questions, and 166 are bridge questions. Examples of bridge and comparison questions are in Table 4.

We examined StoC-ToT performance under the two different question types, each with a different prompt template: one containing only a comparison question as an example and the other containing only a bridge question as an example. We provide the content of our templates in Appendix A. Results are shown in Table 3. We observe that the difference in prompt templates influences the performance of our framework under different question types by a small margin. The comparison questions are generally easier to solve, and StoC-ToT performs better on comparison questions than on bridge questions. StoC-ToT will handle comparison questions better if the prompt template contains comparison questions and vice versa.

Question and Reasoning Types We examine StoC-ToT, Tree-of-Thought prompting, and Chain-of-Thought prompting by comparing their performance under different question-type settings. Detailed results are shown in Figure 3(a). StoC-ToT performs better at both Bridge Questions and Sequential Questions, suggesting that StoC-ToT can avoid reasoning dead-ends and is better at forming intermediate reasoning lines.

We also conduct an in-depth analysis of the reasoning types in the existing MHQA datasets by randomly selecting 100 questions from our testing set. The questions are roughly divided into two categories: 1) tree-like parallel reasoning and 2) chain-like sequential reasoning. Questions with parallel reasoning contain two or more reasoning paths that can be solved arbitrarily. Questions with sequential reasoning follow a strict reasoning chain, and all the sub-questions must be solved to form the correct reasoning process. All comparison questions are parallel reasoning, but some bridge questions contain parallel reasoning. Examples of sequential and parallel reasoning questions are in Table 5. Out of the selected 100 questions, 59 questions were Sequential and 41 questions were Parallel. Results are shown in Figure 3(b). StoC-ToT performs better on both reasoning types, especially on questions containing parallel reasoning. This suggests that StoC-ToT's stochastic way of forming the tree is very effective when solving questions containing multiple reasoning paths.

Performance and Hops As the number of hops increases in a question, the reasoning line gets more complex and varied. Figure 4 shows the performances of different prompting techniques on questions in the MusiQue dataset with differ-



(a) Question Type

(b) Reasoning Type

Figure 3: Performace comparison of Chain-of-Thought, Tree-of-Thought, and SToC-ToT on questions of different question types (Left) and reasoning types (Right). Experiments were done on the HotpotQA dataset.

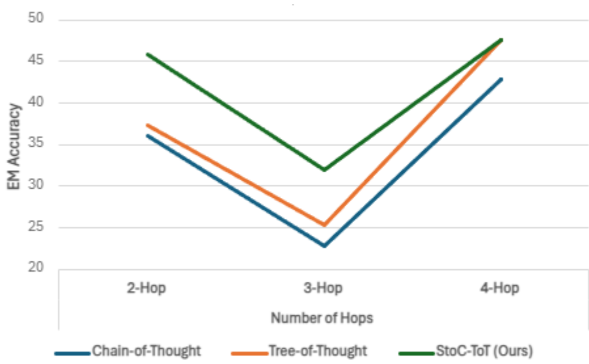


Figure 4: Performance comparison of CoT, ToT, and SToC-ToT on different number of hops in the question. Experiments done in the MusiQue dataset.

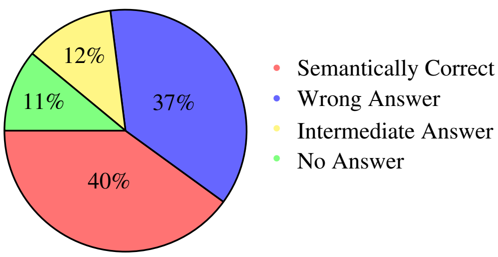


Figure 5: Ratio of different categories in error cases, on the HotpotQA dataset.

ent numbers of hops. SToC-ToT performs best in all categories, demonstrating our framework's superior ability to deal with complex reasoning scenarios. This ablation study was conducted only on GPT4, as other models performed poorly on 3-hop and 4-hop scenarios, regardless of the reasoning prompting technique used.

Error Analysis We conduct a detailed analysis of the errors made by our framework on GPT3 and GPT4, and present our results in Figure 5. We categorize the errors into four types: (1) **No Answer**: our framework did not come up with an answer for the question due to not finishing the reasoning process; (2) **Intermediate Answer**: our framework came up with an answer for one of the intermediate hops instead of for the final question; (3) **Wrong Answer**: our framework came up with an answer that is neither the final answer nor one of the intermediate answers; (4) **Semantically Correct**: our framework came up with the right answer, but did not have an exact match with the final answer. Appendix B shows examples of each error category. Large amounts of error cases were correct answers with extra wording or hallucination errors, signaling potential improvements over our constrained decoding scheme. Reasoning process errors, including no answer and intermediate answer, make up only 25% of the total error cases. This result shows that our framework is capable of building a robust reasoning process for complex questions.

6 Conclusion

This paper proposes SToC-ToT, a stochastic tree-of-thought reasoning framework with constrained generation for multi-hop question answering. SToC-ToT is specialized in dealing with complex reasoning scenarios in natural language tasks. Experiments on two benchmark datasets show that our framework outperforms previous reasoning prompting techniques with multiple Large Language Models. Detailed analysis shows that our framework is capable of building a robust reasoning process given different types of questions. Further research can aim to enhance the reliability of our framework by proposing better validity evaluation schemes and more effective methods for improving groundedness and preventing hallucination.

Limitations

Our framework relies on initiating multiple model instances and requires multiple prompts per round. The repetitive callings impose heavy time costs for our framework, even after implementing our paraphrase module. Another limitation comes from how we generated sub-questions. Currently, we directly prompt the model to generate sub-questions. A more complex standard can be used to increase the quality of the sub-questions generated. Also, more extensive experiments should be provided, including experimenting on other different datasets and case studies.

Ethics Statement

This research adhered to the ethical standards and best practices outlined in the ACL Code of Ethics. Language Models can sometimes produce illogical or inaccurate reasoning paths, so their outputs should be cautiously used. The outputs are only examined to understand how a model arrives at its answers and investigate why it makes certain errors. All experiments used publicly available datasets from previously published works and did not involve ethical or privacy issues.

References

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba. 2019. Constrained decoding for neural NLG from compositional representations in task-oriented dialogue. 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 831–844. Association for Computational Linguistics.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 17682–17690. AAAI Press.

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. *CoRR*, abs/2005.14165.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1535–1546. Association for Computational Linguistics.

Brian Lester, Daniel Pressel, Amy Hemmeter, Sagnik Ray Choudhury, and Srinivas Bangalore. 2020. Constrained decoding for computationally efficient named entity recognition taggers. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 1841–1848. Association for Computational Linguistics.

Xin-Yi Li, Weixian Lei, and Yubin Yang. 2022. From easy to hard: Two-stage selector and reader for multi-hop question answering. *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Comput. Linguistics*, 30(4):417–449.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1314–1324. Association for Computational Linguistics.

Bilgehan Sel, Ahmad Tawaha, Vanshaj Khattar, Ruoxi Jia, and Ming Jin. 2024. Algorithm of thoughts: Enhancing exploration of ideas in large language models. In *Forty-first International Conference on Machine Learning*.

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, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. 9835 musique: Multi-hop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35:*

Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Trans. Assoc. Comput. Linguistics*, 6:287–302.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. 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*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Zhangyue Yin, Yuxin Wang, Xiannian Hu, Yiguang Wu, Hang Yan, Xinyu Zhang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. 2023. Rethinking label smoothing on multi-hop question answering. In *Chinese Computational Linguistics - 22nd China National Conference, CCL 2023, Harbin, China, August 3-5, 2023, Proceedings*, volume 14232 of *Lecture Notes in Computer Science*, pages 72–87. Springer.

Jiahao Zhang, Haiyang Zhang, Dongmei Zhang, Yong Liu, and Shen Huang. 2023. Beam retrieval: General end-to-end retrieval for multi-hop question answering. *CoRR*, abs/2308.08973.

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

A Prompt Templates

Sub Question Generation Template The prompt template containing one comparison question and one bridge question is given below:

prompt: Break a question into high-quality sub-questions that are easier to answer. Here are two examples as guidelines:
"Question: Are Tokyo and Busan in the same country? Thought 1: I could either find which country Tokyo is located in, or which country Busan is located in. Sub Question 1-1: Which country is Tokyo located in? Sub Question 1-2: Which country is Busan located in?"
"Question: Tokyo is located in the country that has what colors present on its national flag? Thought 1: I need to first find out which country Tokyo is located in. Sub Question 1-1: Which country is Tokyo located in?"
Only give out your thought process and current-level sub-questions. Do not give out answers to your questions. Question: *Given Question*. Thought 1:

Prompt-based Constrained Generation Template The prompt template at answering time is given below:

prompt: Given a question and a list of evidence that may of help, give your answer directly, using words only from the vocabulary bank, without any explanations.
Question: *Given Question*. Evidence as reference: *Given Evidence*. Vocabulary Bank: *Given Vocabulary*. Answer:

B Examples of the Error Cases

•Type-2: Intermediate Answer

Question:
Where does the hotel and casino located in which Bill Cosby's third album was recorded?
Answer given by STOC-TOT on GPT4:
Las Vegas.
Golden Answer:
Las Vegas Strip in Paradise.

•Type-3: Wrong Answer

Question:
Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?
Answer given by STOC-TOT on GPT4:
siri remote and devices with netsupport manager software
Golden Answer:
keyboard function keys

•Type-4: Semantically Correct

Question:
Roger O. Egeberg was Assistant Secretary for Health and Scientific Affairs during the administration of a president that served during what years?
Answer given by STOC-TOT on GPT4:
1969 to 1974
Golden Answer:
1969 until 1974