Faithful Question Answering with Monte-Carlo Planning

Ruixin Hong[1], Hongming Zhang[2], Hong Zhao[1], Dong Yu[2], Changshui Zhang[1]
[1]Institute for Artificial Intelligence, Tsinghua University (THUAI);
[1]Beijing National Research Center for Information Science and Technology (BNRist);
[1]Department of Automation, Tsinghua University, Beijing, P.R.China
[2]Tencent AI Lab, Seattle
hrx20@mails.tsinghua.edu.cn, hongmzhang@tencent.com,
dyu@global.tencent.com, zcs@mail.tsinghua.edu.cn,

Abstract

Although large language models demonstrate remarkable question-answering performances, revealing the intermediate reasoning steps that the models *faithfully* follow remains challenging. In this paper, we propose FAME (FAithful question answering with MontE-carlo planning) to answer questions based on faithful reasoning steps. The reasoning steps are organized as a structured entailment tree, which shows how premises are used to produce intermediate conclusions that can prove the correctness of the answer. We formulate the task as a discrete decision-making problem and solve it through the interaction of a reasoning environment and a controller. The environment is modular and contains several basic task-oriented modules, while the controller proposes actions to assemble the modules. Since the search space could be large, we introduce a Monte-Carlo planning algorithm to do a look-ahead search and select actions that will eventually lead to high-quality steps. FAME achieves advanced performance on the standard benchmark. It can produce valid and faithful reasoning steps compared with large language models with a much smaller model size.

1 Introduction

Enabling machines to reason and answer questions is a long-term pursuit in the AI community (McCarthy et al., 1960). In the field of question-answering (QA), large language models (LLMs) have achieved strong performances (Bommasani et al., 2021; Brown et al., 2020; Kojima et al., 2022). However, the intermediate reasoning steps from the known premises to the answer are often implicit and invisible. While some approaches encourage LLMs to produce the reasoning steps explicitly before generating the answer (Wei et al., 2022b), the answer may not *faithfully* follow the intermediate steps, i.e., the model could generate irrelevant or invalid steps while still resulting in the correct answer (Wei et al., 2022b; Zelikman et al., 2022; Creswell et al., 2022). Such a lack of faithfulness makes it difficult for users to trust the answers and debug the wrong answers, diminishing the overall trustworthiness of the QA systems.

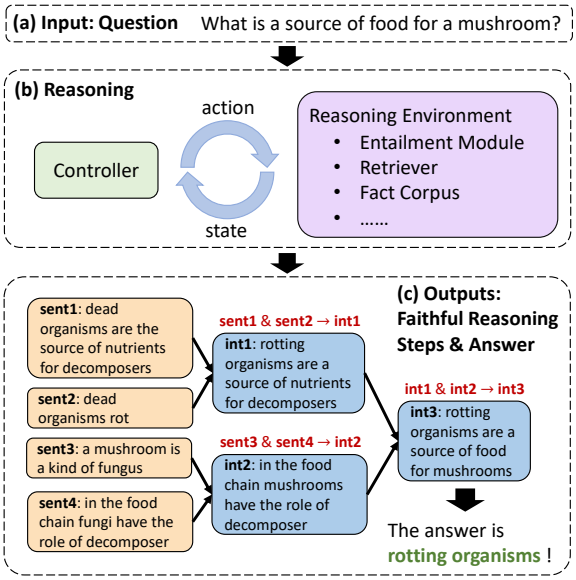


Figure 1: Given a question, FAME performs reasoning through the iterative interaction of a controller with a reasoning environment. It produces the reasoning steps (in the form of an entailment tree) and the answer faithfully following from the steps. The entailment tree contains the basic fact (sent_*) and novel intermediate conclusions (int_*) connected by entailment steps.

To tackle this issue, the recently proposed faithful question-answering (FQA) task (Tafjord et al., 2022; Creswell and Shanahan, 2022) asks the system to provide the reasoning steps that the answer faithfully follows, as demonstrated in Figure 1(a) and (c). The reasoning steps are organized as an *entailment tree* (Dalvi et al., 2021), where each non-leaf node indicates an intermediate entailment step. The provision of the faithful steps allows users to inspect and debug the system's reasoning process, potentially enabling the construction of interactive and teachable QA systems (Dalvi et al., 2022).

Existing FQA methods typically adopt a step-

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 3944–3965

wise approach to generate an entailment step at a time (Tafjord et al., 2022; Creswell and Shanahan, 2022). When determining which step to generate next, they produce several candidate steps and select one based on the validity of the step. However, they do not explicitly consider whether the selected step will ultimately result in a high-quality tree that supports the answer. For complex questions that require multiple reasoning steps, such a lack of foresight might lead to the irreversible miss of the optimal step in a huge search space. Furthermore, existing methods are based on either the model's internal beliefs or a small number of given facts, which limits their ability to ground to the external world and update their known facts.

In this paper, we propose FAME, a novel FQA method integrating Monte-Carlo planning. We formulate the task as a discrete decision-making problem and solve it through the interaction of a reasoning environment and a controller, as shown in Figure 1. The reasoning environment is modular. We decompose the reasoning into several basic task-oriented modules, such as a retriever for updating known facts and a single-step entailment module for combining several premises to obtain a novel intermediate conclusion. To assemble these modules, we leverage a controller (implemented with a generative language model) to observe the state of the environment and propose the next actions. The final answer is derived based on the validity and faithfulness of the generated entailment tree.

To select actions foresightedly, we introduce a Monte-Carlo planning algorithm (Kocsis and Szepesvári, 2006) to do the look-ahead search. Specifically, we assign an action value to each candidate action, which is iteratively updated based on the quality of its successor states after the action is explicitly executed. With these values, we could make more informed decisions and select the actions that will eventually lead to a high-quality tree. In addition, we design a verifier-guided iterative training technique to train the controller.

Experiments on the standard benchmark EntailmentBankQA (Creswell and Shanahan, 2022) show that FAME outperforms previous best FQA methods by a large margin. Manual evaluation results demonstrate that FAME could produce valid and faithful reasoning steps compared with LLMs (i.e., GPT-3 and ChatGPT). Further ablation results illustrate the advantages of Monte-Carlo planning compared to other planning methods.

2 Related Work

Explicit Reasoning with Language Models. LLMs can achieve strong QA performances, even in few-shot and zero-shot settings (Bommasani et al., 2021; Brown et al., 2020; Kojima et al., 2022). Recent approaches that explicitly generate the reasoning steps to derive the final answer (Zelikman et al., 2022; Wang et al., 2022; Zhou et al., 2022a; Lampinen et al., 2022) have shown significant improvement for many tasks (Suzgun et al., 2022). For example, Chain-of-Thought (Wei et al., 2022b) encourages LLMs to generate several steps via few-shot prompting. However, the generated steps could be unfaithful (Wei et al., 2022b; Zelikman et al., 2022; Creswell et al., 2022). To tackle this issue, faithful question-answering (FQA) proposes to answer questions based on the faithful reasoning steps (Tafjord et al., 2022; Creswell and Shanahan, 2022; Weir and Durme, 2022; Bostrom et al., 2022), where each step is a valid entailment and the intermediate conclusions support the answer. Existing FQA methods typically adopt a step-wise approach. To decide the next step, their strategy (e.g., overgenerate-and-filter of Entailer (Tafjord et al., 2022) and beam search of Selection-Inference (SI) (Creswell and Shanahan, 2022)) might lack foresight and could be inadequate for complex questions. Our work explicitly does the look-ahead search to make more foresighted decisions. In addition, Entailer generates facts using the model's internal representation, which may hallucinate possibly incorrect facts to support its answer (Tafjord et al., 2022). SI requires a complete set of supporting facts for the question and can not update facts. Our method is based on a deterministic corpus to prevent hallucination and could adaptively update facts using a retriever.

Our modular design is related to the approaches that break down the questions into smaller modules (Khot et al., 2022; Zhou et al., 2022b; Sanyal et al., 2022; Kazemi et al., 2022). In this paper, we compare against SI (Creswell and Shanahan, 2022), which is the representative modular FQA approach.

Explanation for Question Answering. Great efforts have been devoted to improving the explainability of QA systems (Wiegreffe and Marasovic, 2021; Thayaparan et al., 2020; Lamm et al., 2021; Rosenthal et al., 2021; Huang et al., 2022). Recently, EntailmentBank (Dalvi et al., 2021) proposes to formulate the reasoning steps of QA systems as multi-step textual entailments, which pro-

Action	Effect	Example
`Retrieve`: <query>	Call a retriever to retrieve facts by <query> from the fact corpus C and update candidate premises X. The query can be selected from $\{H\} \cup X$ adaptively.	`Retrieve`: int_1
`Entail`: <premises>	Call an entailment module to generate a conclusion given the selected <premises> as input. Add a new step formed by the <premises> and the generated conclusion to T_p.	`Entail`: sent_1 & sent_2
`End`: <is_proved>	End reasoning and return whether the controller considers H is proved. <is_proved> $\in$ {"proved", "unproved"}.	`End`: proved

Table 1: Action space for the controller. If the controller generates other text, it is treated as an invalid action.

vides the most detailed and informative explanation. A series of methods are developed to reconstruct such a tree-structured explanation for the correct answer (Dalvi et al., 2021; Hong et al., 2022; Yang et al., 2022; Ribeiro et al., 2022; Liu et al., 2022). Our work is based on EntailmentBank but focuses on QA instead of post-hoc explanations.

Monte-Carlo Planning for Language. Despite remarkable successes in games (Silver et al., 2017; Schrittwieser et al., 2020), few works attempt to apply Monte-Carlo planning (MCP) to language. Previous works use MCP for decoding during language generation (Kumagai et al., 2016, 2018; Leblond et al., 2021; Chaffin et al., 2022), but how to use MCP for QA is still underexplored. Our work formulates QA as a decision-making problem and explores the utilization of MCP.

3 Task Definition

Faithful question-answering (Tafjord et al., 2022; Creswell and Shanahan, 2022) requires to answer the question and provide the valid reasoning steps that the answer faithfully follows. The inputs include a question Q and candidate options $O = \{o_1, o_2, \ldots, o_{|O|})\}$.[1] The desired outputs are valid reasoning steps T in the form of the entailment tree and an answer $o_a \in O$, which follows T. The entailment tree T consists of multi-premise entailment steps, whose leaf nodes (sent_*) are facts selected from a fact corpus C, intermediate nodes (int_*) are novel intermediate conclusions. A tree is considered *valid* if each non-leaf node is a valid entailment of its immediate children, and is considered *faithful* if its conclusion of the root node can support the option o_a.

4 Our Approach: FAME

Given a question Q and options O, following previous works (Tafjord et al., 2022; Weir and Durme, 2022), we first convert them into declarative hypotheses $\{H_1, \ldots, H_{|O|}\}$.[2] We then try to generate an entailment tree for each hypothesis in a forward chaining manner and select the most plausible option based on the validity and faithfulness of trees.

We propose to formulate the task as a discrete decision-making problem. The reasoning is done through several interactions between a reasoning environment and a controller. In each interaction, the controller observes a state from the environment and predicts the next action. Then the environment executes the action and updates its state.[3] Since the search space could be large, we introduce a Monte-Carlo planning algorithm to select the optimal action. We introduce details about the environment, controller, and Monte-Carlo planning in Sec 4.1, 4.2, and 4.3, respectively.

4.1 Reasoning Environment

State. A reasoning state $s = \{H, T_p, X\}$ consists of three parts: a target hypothesis H, a partial tree T_p, and candidate premises X. T_p contains the entailment steps so far. X is the set of sentences that can be selected as premises for the next entailment step. Sentences in X are either facts retrieved from the corpus (sent_*) or conclusions generated by previous steps (int_*). The maximum size of X is restricted to 25 to fit the input length limitation of common language models (Dalvi et al., 2021).
Action. We consider three types of action $a \in \mathcal{A}(s)$ for a state s, as shown in Table 1. The entailment module is a seq2seq generation model to perform single-step entailment reasoning. Implementation details of the environment can be found in Sec 5.2.

4.2 Reasoning Controller

The controller is a sequence generation model whose input is a linearized state and whose outputs

[1] For open-ended questions, we follow Tafjord et al. (2022) to collect candidate options using an external source (e.g., Macaw (Tafjord and Clark, 2021))

[2] We follow Tafjord et al. (2022) to use a generation model whose input is $q + o_i$ and output is H_i. Specifically, we use a T5-large (Raffel et al., 2020) trained on EntailmentBank.

[3] Illustrations of the reasoning process are in Appendix B.

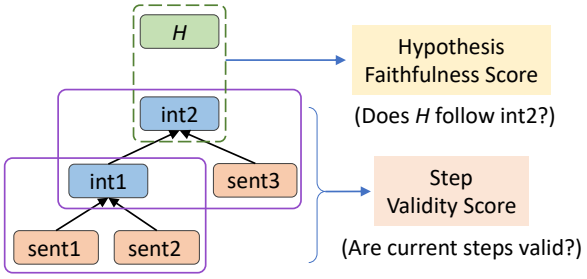


Figure 2: The step verifier V scores a state based on its step validity and hypothesis faithfulness. The hypothesis H is the declarative form of the question and option.

are actions. The input sequence is the concatenation of H, T_p, and X. The linearized T_p is a series of steps, where the step premises are connected with "&" and the step conclusion comes after "→." For each state, the controller predicts multiple candidate actions and their likelihood scores.

4.3 Monte-Carlo Planning

To select the optimal actions, solely using the controller's likelihood scores could be insufficient. We propose to select actions based on the qualities of the successor states after the execution of actions. Specifically, we first introduce a state verifier to estimate the scores of states (Sec. 4.3.1). Each candidate action is assigned an action value Q, which is updated iteratively by the planning algorithm. Our algorithm is based on the Monte-Carlo tree search with an upper confidence bound. The upper confidence bound helps to balance between exploiting the currently best actions to obtain precise estimates and exploring sub-optimal actions to ensure that no good actions are missed. Given the sufficient computational cost, this method can converge asymptotically to the optimal actions in the single agent problem (Kocsis and Szepesvári, 2006; Rosin, 2011; Schrittwieser et al., 2020).

4.3.1 State Verifier

A state is considered valuable if its T_p is valid and T_p can support the hypothesis H. Thus, we introduce a state verifier V to estimate the score of a state from two perspectives, as illustrated in Figure 2. (1) **Step Validity**. Is T_p valid? We use a step verifier V_s to check the validity of a single step. The step verifier takes the premises and the conclusion as input and predicts a continuous score in $[0, 1]$ indicating how likely the conclusion is to be entailed by the premises. Then the step scores are aggregated to produce the validity score of T_p,

$$\texttt{Valid}(T_p) = \frac{1}{|T_p|} \sum_{\text{step} \in T_p} V_s(\text{step}). \quad (1)$$

(2) **Hypothesis Faithfulness**. Does H faithfully follow T_p? To check this, we extract the highest conclusion int_h in T_p (e.g., int_2 in Figure 2) and verify if int_h can support H. Following Dalvi et al. (2021), we use `BLEURT` (Sellam et al., 2020) as the scorer V_h to estimate the sentence similarity between int_h and H. In addition, we check whether H is entailed by int_h with the step verifier V_s.

$$\texttt{Faithful}(T_p, H) = (V_h(\text{int}_h, H) + V_s(\text{int}_h \rightarrow H))/2. \quad (2)$$

If more than one int_h exists, we take their maximum faithfulness score. The overall state score is composed of its validity and faithfulness scores,

$$V(s) = (\texttt{Valid}(T_p) + \texttt{Faithful}(T_p, H))/2. \quad (3)$$

If there is no step yet ($T_p = \emptyset$), then $V(s) = 0$.

4.3.2 Planning Algorithm

Figure 3 illustrates our planning algorithm. We construct a planning tree where each node in the planning tree is correlated with a state s. For each action a of s, we store a set of statistics $\{P(s,a), Q(s,a), N(s,a), \textit{Tra}(s,a)\}$, where P is the prior score (i.e., the likelihood score from the controller), Q is the action value, N is the visit count, and *Tra* is the deterministic state transition. $Q(s,a)$ can be viewed as the a posteriori score, which considers the future impact of a. The planning algorithm ends after a fixed budget of simulations. Each simulation consists of three stages.

• **Selection.** The selection stage starts from the initial state s^0 (root node of the planning tree) and finishes when it reaches an unexpanded leaf node s^m. For each $k = 1, \ldots, m+1$, an action a^k is selected based on the statistics of s^{k-1} to maximize an upper confidence bound (Silver et al., 2017),

$$a^k = \arg\max_{a \in \mathcal{A}(s)} \left[Q(s,a) + c_p P(s,a) \frac{\sqrt{\sum N(s,\cdot)}}{1 + N(s,a)} \right], \quad (4)$$

where $\sum N(s,\cdot)$ is the total visit counts of s and c_p is a constant controlling the influence of the prior $P(s,a)$ over $Q(s,a)$. With the upper confidence bound, the selection strategy initially favors actions with high prior scores and low visit counts (the second part) and gradually prefers actions with high action values (the first part). The next states are looked up through the state transition *Tra*.

• **Expansion.** For the leaf node s^m and the selected action a^{m+1}, we execute a^{m+1} and get the next state s^{m+1}. A new node correlated with s^{m+1}

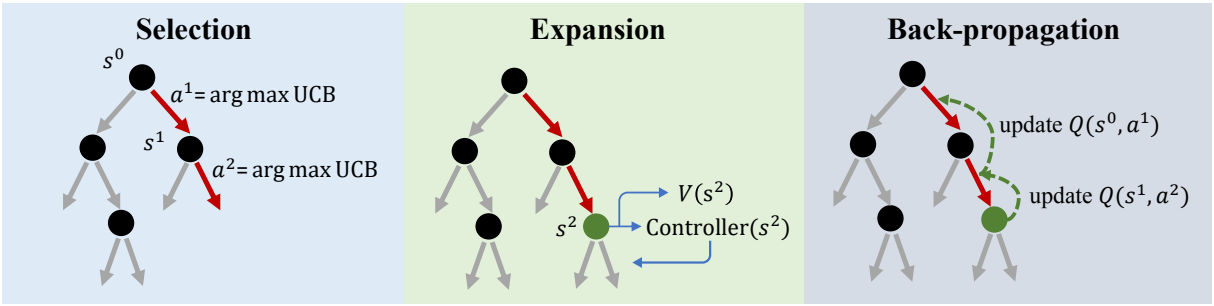


Figure 3: The illustration of Monte-Carlo planning. We construct a planning tree where each node is a state s. **Selection**: In each simulation, we traverse the planning tree from s^0 by selecting actions with the maximum upper confidence bound (UCB, equation (4)). **Expansion**: We expand a new node and evaluate the new state with the state verifier V. The candidate actions of the new state are predicted by the controller and stored. **Back-propagation**: Action values Q are updated based on the scores of successor states after the action is executed. For simplicity, here we assume that there are only two candidate actions for each state.

is added to the planning tree. We record the transition $Tra(s^m, a^{m+1}) = s^{m+1}$ and the state score $V(s^{m+1})$. We then use the controller to predict the candidate actions a and their prior scores p_a. Each action is initialized with $\{P(s,a)=p_a, Q(s,a)=0, N(s,a)=0\}$. Note that during each simulation, we only execute one action and call the controller, environment, and state verifier at most once.

• **Back-propagation.** The statistics along the trajectory $(s^0, a^1, \ldots, s^m, a^{m+1})$ are updated. For the final action, we set $Q(s^m, a^{m+1}) = V(s^{m+1})$. For $k = m, \ldots, 1$, we update $Q(s^{k-1}, a^k)$ by integrating information from its next state s^k. We consider $Q(s^{k-1}, a^k)$ as the probability that a^k can correctly solve the task. Given that a^k can solve the task only requires one action of s^k can solve the task, the value of a^k should be the disjunction of all actions of s^k. We use the maximization operator to soften the disjunction[4] and obtain the value estimation G in this simulation,

$$G(s^{k-1}, a^k) = \bigvee_{a \in \mathcal{A}(s^k)} Q(s^k, a) = \max_{a \in \mathcal{A}(s^k)} Q(s^k, a). \quad (5)$$

We update the statistics as follows,

$$Q(s^{k-1}, a^k) \leftarrow \frac{N(s^{k-1}, a^k) \cdot Q(s^{k-1}, a^k) + G(s^{k-1}, a^k)}{N(s^{k-1}, a^k) + 1},$$
$$N(s^{k-1}, a^k) \leftarrow N(s^{k-1}, a^k) + 1. \quad (6)$$

4.3.3 Option Selection

At the end of the planning, an extra selection stage is run to get the best state s_{best} using equation (4). We score the option o corresponding to H by aggregating the verifier's and controller's judgments,

$$\texttt{Score}(o) = (V(s_{\text{best}}) + P(s_{\text{best}}, \texttt{End: proved}))/2, \quad (7)$$

where $P(s_{\text{best}}, \texttt{End: proved})$ is the action likelihood score from the controller. With all options scored, we eventually select the option with the highest score as the answer.

4.4 Controller Training

Given a correct trajectory $(s^0, a^1, \ldots, s^m, a^{m+1})$, the controller is trained to maximize the likelihood of the correct actions $P(s^{k-1}, a^k)$. We first train the controller with behavior cloning (Nakano et al., 2021) to imitate an oracle strategy. Then, we iteratively use the controller to generate trajectories and fine-tune the controller with high-quality trajectories selected by the verifier.

Behavior Cloning. Given a state, the oracle strategy gives an action based on the gold entailment tree: (1) If the hypothesis H is already in X, return `End: proved`; (2) Otherwise, if X contains all premises of the next step, return `Entail:` <premises>; (3) Otherwise, select a query from $\{H\} \cup X$ such that the updated X contains as many leaf facts of the gold tree as possible, and return `Retrieve:` <query>.

Verifier-Guided Iterative Training. Training with behavior cloning alone may have two problems. First, since the entailment trees are annotated only for the correct options, we do not have training trajectories for the wrong options. Second, the trajectory for the correct option may not be unique. To tackle these problems, we propose an iterative training process. **Step 1:** We train a controller using behavior cloning on the correct options. **Step 2:** We use the controller to generate trajectories for all training options. For the correct option, we check the quality of the trajectory with the state verifier. If the state score of the final state is greater than a

[4] We follow the T-conorm of Gödel fuzzy logic (Caicedo and Rodríguez, 2010; Gupta and Qi, 1991).

threshold, we add the trajectory to the training data. In this way, we could find some potentially correct trajectories, which may not be the same as the annotated ones but can still produce valid and faithful entailment trees. For the wrong option, we modify each (s^{k-1}, a^k) to $(s^{k-1}, \texttt{End:unproved})$ and add them to the training data. **Step 3:** We fine-tune the controller with the updated training data and go to Step 2 until the performance is saturated.

5 Experiments

5.1 Dataset

We conduct experiments based on EntailmentBank (Dalvi et al., 2021). It consists of 1,840 expert-annotated entailment trees, each of which corresponds to a hypothesis derived from a question+*correct option* from the ARC (Clark et al., 2018) dataset. The leaf facts of trees are selected from the WorldTreeV2 corpus (Xie et al., 2020).

Since EntailmentBank is originally designed for post-hoc tree reconstruction instead of QA, we follow Creswell and Shanahan (2022) to convert it to EntailmentBankQA by adding back the 4-way multiple options from the ARC dataset. Summary statistics are shown in Table 2. Answering these questions requires retrieving from a large fact corpus and complex multi-hop reasoning (each tree contains 7.6 nodes and 3.2 steps on average).

5.2 Implementation Details

Retriever. The retriever is based on the Siamese Network encoder provided by Sentence Transformers (Reimers and Gurevych, 2019). We fine-tune the all-mpnet-base-v2 encoder via a contrastive loss (van den Oord et al., 2018) to maximize the cosine similarity between the hypothesis and its leaf facts of the tree from the EntailmentBank training split. We use the fact corpus provided by EntailmentBank following previous works.

Entailment Module. The entailment module is a T5-large model which takes the premises as input and generates the conclusion. Following MetGen (Hong et al., 2022), we divide the single-step entailment reasoning into a set of basic logical reasoning (i.e., substitution, conjunction, and if-then). Special prefixes are used to specify the reasoning type the model should perform. We train the module with entailment steps from the EntailmentBank training split. Given premises, we generate conclusions using all types of modules and select the conclusion with the highest step verifier score.

	Train	Dev	Test	All
Questions	1,313	187	340	1,840
Easy	920	128	234	1,282
Chal (Challenge)	393	59	106	558
Entailment Steps	4,175	597	1,109	5,881

Table 2: EntailmentBankQA Statistics.

Step Verifier V_s. We fine-tune a DeBERTa-large model (He et al., 2021) to classify if a step is valid. We use the human-labeled data provided by Tafjord et al. (2022), which contains 1,827 valid steps and 1,564 invalid steps. We also include 4,175 valid steps from the EntailmentBank training split.

Controller. The controller is implemented with a T5-large model. For each state, we use the beam search to generate five candidate actions and then exclude the ill-formed and invalid actions (e.g., $\texttt{Retrieve:}\ \text{int}_2$, but int_2 is not yet available in X). For iterative training, we use a threshold of 0.98 to select high-quality trajectories. Model performance is typically saturated after five iterations.

Planning Algorithm. We select the hyperparameters with dev split (Appendix C.1). c_p in equation (4) is 0.2. The simulation/action budget is 30. Note that we execute only one action in each simulation. We run our method three times and report the mean and standard deviation. More implementation details can be found in Appendix A.

5.3 Baselines

We compare with recent SOTA FQA methods. **Entailer** (Tafjord et al., 2022) uses the model's internal beliefs to produce trees and answers. For each hypothesis, it generates six candidate steps and selects the best one with a verifier. **NELLIE** (Weir and Durme, 2022) is a backward-chaining inference engine based on a semi-structured fact base and extra labeled inference templates. **Selection-Inference (SI)** (Creswell and Shanahan, 2022) iteratively generates trees by a selection model and an inference model, and produces answers with a halter model. For Entailer, we use its open-source T5-large model and default parameters.[5] Since it also needs hypothesis conversion, we use the same conversion results as ours for a fair comparison.

[5] https://allenai.org/data/entailer

Method	All	Easy	Chal
NELLIE †	43.7	45.5	39.6
Entailer	53.1	56.4	46.2
FAME (Ours)	**67.1**±0.6	**70.8**±0.4	**59.1**±1.2

Table 3: Answer accuracy (%) on the EntailmentBankQA test split. † indicates results from the published paper. All methods are based on T5-large.

Method	Task 1	Task 2
SI+Halter †	72.4	55.9
SI+Halter+Search †	83.2	72.9
FAME (Ours)	**91.5**±0.8	**78.2**±0.9

Table 4: Task 1 and Task 2 accuracy (%) on the EntailmentBankQA test split. † indicates results from SI. SI is based on Chinchilla-7B (Hoffmann et al., 2022).

6 Result Analysis

6.1 Faithful Question Answering

EntailmentBankQA. As shown in Table 3, FAME outperforms baseline methods by a large margin, improving the accuracy from 53.1% to 67.1%. We also experiment in the settings of Creswell and Shanahan (2022), where a small set of facts is provided. Task 1 provides exactly the leaf facts of the gold tree for the correct option, while Task 2 additionally includes several distractors. Since retrieval is prohibited and not required, we remove the retrieval from the action space. Table 4 shows the results. FAME could be adapted to these settings and achieve higher accuracy than SI, which is based on larger language models. The errors in FAME are traceable. The most frequent cause of errors is the mistakes in the intermediate steps (See the error analysis in Appendix C.4).

Cross-Dataset Performance. To evaluate the generalization capability of our method, we conduct experiments on WorldTreeQA (Xie et al., 2020) and OBQA (Mihaylov et al., 2018) following previous works (Weir and Durme, 2022; Tafjord et al., 2022). We evaluate on the test split of WorldTreeQA (1,177 easy and 487 challenge questions, no overlap with EntailmentBankQA) and OBQA (500 questions) without further fine-tuning on their training split. We use the fact corpus that is included with the dataset. As shown in Table 5, FAME achieves better cross-dataset performances.

Method	WorldTreeQA			OBQA
	All	Easy	Chal	
NELLIE †	38.3	40.8	32.3	-
Entailer	50.7	54.4	41.9	45.6
FAME (Ours)	**61.5**±0.4	**65.1**±0.4	**52.6**±0.3	**46.6**±0.4

Table 5: Cross-dataset results on the WorldTreeQA and OBQA test split.

6.2 Reasoning Step Quality

Automatic Evaluation on EntailmentBank. To investigate whether our method can generate a valid entailment tree, we first perform the automatic evaluation on EntailmentBank, where we generate trees for correct options. The validity of the generated tree is evaluated by comparing its leaves, step structures, and intermediate conclusions against the gold one. The F1 score is computed, and the AllCorrect score is 1 if F1=1, otherwise 0. The Overall AllCorrect metric measures whether the generated tree and the gold tree are identical.[6] Please refer to Appendix E for more evaluation details. We use the controller trained with behavior cloning on the correct options. We compare with the SOTA methods that are specifically designed for tree reconstruction. As shown in Table 6, FAME achieves competitive performances in all metrics, indicating that it could generate valid and high-quality trees.

Manual Evaluation on EntailmentBankQA. To make a more accurate investigation, we manually evaluate the quality of trees for the options *selected by models*. We evaluate along three dimensions. **Fact Validity (FV)**: Are the leaf facts correct in the real world? **Step Validity (SV)**: Are the intermediate steps valid entailments? A step is considered invalid if its conclusion does not follow from the premises or trivially repeats one of the premises. **Hypothesis Faithfulness (HF)**: Can the conclusion support the selected answer (even if the answer is not the correct option)? **Overall**: The overall score is 1 if all the facts and steps are valid and the conclusion supports the answer. We invite three students as experts, and the inter-annotation agreement (Cohen's κ) for FV/SV/HF is 0.726/0.704/0.811.

We also investigate whether FQA can be solved by very large language models, such as GPT-3 (Brown et al., 2020) and ChatGPT(Schulman et al., 2022). For GPT-3, we use Chain-of-Thought

[6] As discussed by Yang et al. (2022) and Hong et al. (2022), these automatic metrics do not account for the existence of multiple valid trees. And the Overall AllCorrect score is a very harsh metric, but it is the fairest metric we could use.

Method	Leaves		Step Structures		Intermediates		Overall
	F1	AllCorrect	F1	AllCorrect	F1	AllCorrect	AllCorrect
EntailmentWriter (Dalvi et al., 2021) †	35.7	2.9	6.1	2.4	33.4	7.7	2.4
MetGen (Hong et al., 2022) †	34.8	8.7	9.8	8.6	36.6	20.4	8.6
NLProofs (Yang et al., 2022) †	43.2	8.2	11.2	6.9	**42.9**	17.3	6.9
RLET (Liu et al., 2022) †	38.3	9.1	11.5	7.1	34.2	12.1	6.9
IRGR (Ribeiro et al., 2022) †	**45.6**	11.8	16.1	11.4	38.8	**20.9**	11.5
FAME (Ours)	43.4±0.3	**13.8**±0.6	**16.6**±0.1	**12.4**±0.4	40.6±0.3	19.9±1.2	**11.9**±0.4

Table 6: The results of entailment tree generation on EntailmentBank test split. † indicates results from the published paper. RLET is based on DeBERTa-large, and all other methods are based on T5-large.

Method	FV	SV	HF	Overall
GPT-3 w/ CoT	**100.0**	23.2	72.0	4.0
ChatGPT	**100.0**	38.3	52.0	14.0
Entailer	72.8	48.0	**82.0**	24.0
FAME (Ours)	**100.0**	**66.0**	**82.0**	**46.0**

Table 7: Manual evaluation results on 50 questions randomly sampled from the test split. FV/SV/HF denotes fact validity/step validity/hypothesis faithfulness. GPT-3 and ChatGPT have 175B parameters. Entailer and FAME are based on T5-large (770M).

Algorithm	All	Easy	Chal
Greedy	59.1	62.9	50.8
Overgenerate-and-filter	62.0	66.4	52.5
Beam search	64.2	67.4	57.1
Monte-Carlo planning	**67.7**	**69.5**	**63.8**

Table 8: Ablation results of the planning algorithm on the dev split. All algorithms use the same action budget.

prompting (CoT) by treating the entailment tree as the thought. For ChatGPT, we describe the task in detail. Discussion and examples of prompts are in Appendix D. GPT-3/ChatGPT achieves an answer accuracy of 86%/92%, showing that our prompts could appropriately elicit the model's reasoning ability.[7]

Table 7 shows the results. We can make the following observations. (1) While achieving high accuracy rates, LLMs struggle to produce valid and faithful steps, e.g., only 38.3% of ChatGPT's steps are valid. (2) Entailer achieves a high HF score but a low FV score since it is not grounded in a fact corpus and may hallucinate possibly incorrect facts to support its answer. (3) FAME can answer questions faithfully (HF=82.0%) based on valid reasoning steps (SV=66.0%) and human-confirmed facts, achieving the highest overall score.

[7]We use the `text-davinci-003` model for GPT-3 and `Dec 15 Version` for ChatGPT.

	State Verifier		Answer Accuracy		
	Valid	Faithful	All	Easy	Chal
(a)	V_s	V_s+V_h	**67.7**	**69.5**	**63.8**
(b)	V_s	×	52.9	55.1	48.3
(c)	×	V_s+V_h	61.0	62.5	57.6
(d)	V_s	V_h	62.6	65.2	56.8
(e)	V_s	V_s	64.2	67.2	57.6

Table 9: Ablation results of the state verifier (equation (3)) on the EntailmentBankQA dev split. V_s is the step verifier and V_h is the sentence similarity scorer.

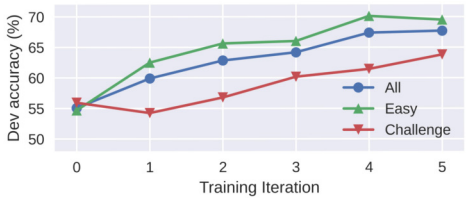


Figure 4: Ablation results of the iterative training.

6.3 Ablation Study

Planning Algorithm. To investigate the effectiveness of MCP, we design three comparison algorithms. **Greedy** algorithm selects the action with the maximum prior likelihood score for each state. **Overgenerate-and-filter** (Tafjord et al., 2022) generates $K = 5$ candidate actions, executes the actions to get the next states, and selects the one with the highest state score. **Beam search** (Creswell and Shanahan, 2022) maintains the best B states. At each time step, it generates K actions for each state, executes the actions, and keeps the top-B best states for further reasoning.[8] We set an action budget of 30 for all algorithms. The algorithms stop when the action budget is used up. Results in Table 8 show that MCP improves performance over the comparison algorithms, especially for the challenge questions (from 57.1% to 63.8%).[9] MCP

[8]We set the beam size B to 3, which achieves the best result. Appendix C.2 includes more discussion of the parameters.

[9]We also discuss the performance broken down by the length of gold steps in Appendix C.3.

could do the look-ahead search efficiently and select actions foresightedly, so as to perform better.

State Verifier. As shown in Table 9, verifying both the step validity and hypothesis faithfulness is necessary (comparing (a), (b), and (c)). We also find that the ensemble of V_s and V_h yields a more accurate faithfulness score and thus improves performance (comparing (a), (d), and (e)).

Iterative training. Figure 4 shows that the performance improves with the increasing number of verifier-guided training iterations. We randomly sample 100 reasoning trajectories selected by the verifier and find that 61% of them obtain the same trees as the gold ones, 30% obtain different but still valid trees, and 9% obtain invalid trees, demonstrating the effectiveness of iterative training.

7 Conclusion

We propose FAME for faithful question answering. It formulates the task as a discrete decision-making problem and tackles it with a modular reasoning environment and a controller. FAME uses the Monte-Carlo planning to make more informed and foresighted decisions. Experiments show that FAME can answer questions based on valid and faithful steps and achieve advanced performance.

8 Acknowledgements

We appreciate the anonymous reviewers for their insightful comments. We would like to thank Qihan Liu for the helpful discussions. This work is supported by National Key Research and Development Program (2020AAA0107800) and the Guoqiang Institute of Tsinghua University (2020GQG0005).

Limitations

Despite outperforming previous best methods, our method still has several limitations and substantial room for future improvement.

First, the variety of modules is limited in the current reasoning environment. It would be interesting to introduce a wider variety of modules (e.g., a numerical calculator) to make our method more general.

Second, our method currently retrieves facts from a fixed corpus. While this is efficient for the specific domain, it may not be sufficient for questions not covered by the fact corpus. It would be more powerful if we retrieve up-to-date information using a modern search engine as our retriever.

Third, in our experiments, we try our best to select the appropriate prompts to motivate GPT-3 and ChatGPT to generate reasoning steps and answers. With our prompts, GPT-3 and ChatGPT can achieve high answer accuracy. But it is hard to guarantee that our prompts are the best ones to elicit the model's capabilities completely.

Finally, although scaling up the size of the language models may lead to emergent abilities (Wei et al., 2022a), in this paper, we do not experiment with larger language models (e.g., T5-11B) due to the computational constraints.

To the best of our knowledge, our work is foundational research, and we do not find obvious risks related to malicious harmful effects, environmental impact, fairness considerations, or privacy considerations.

References

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. 2021. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258.

Kaj Bostrom, Zayne Sprague, Swarat Chaudhuri, and Greg Durrett. 2022. Natural language deduction through search over statement compositions. *CoRR*, abs/2201.06028.

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*.

Xavier Caicedo and Ricardo Oscar Rodríguez. 2010. Standard gödel modal logics. *Stud Logica*, 94(2):189–214.

Antoine Chaffin, Vincent Claveau, and Ewa Kijak. 2022. PPL-MCTS: constrained textual generation through discriminator-guided MCTS decoding. 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 2953–2967. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Antonia Creswell and Murray Shanahan. 2022. Faithful reasoning using large language models. *CoRR*, abs/2208.14271.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2022. Selection-inference: Exploiting large language models for interpretable logical reasoning. *CoRR*, abs/2205.09712.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7358–7370. Association for Computational Linguistics.

Bhavana Dalvi, Oyvind Tafjord, and Peter Clark. 2022. Towards teachable reasoning systems: Using a dynamic memory of user feedback for continual system improvement. *CoRR*, abs/2204.13074.

Madan M Gupta and J11043360726 Qi. 1991. Theory of t-norms and fuzzy inference methods. *Fuzzy sets and systems*, 40(3):431–450.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *CoRR*, abs/2111.09543.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models. *CoRR*, abs/2203.15556.

Ruixin Hong, Hongming Zhang, Xintong Yu, and Changshui Zhang. 2022. MetGen: A module-based entailment tree generation framework for answer explanation. In *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 1887–1905. Association for Computational Linguistics.

Yinya Huang, Hongming Zhang, Ruixin Hong, Xiaodan Liang, Changshui Zhang, and Dong Yu. 2022. Metalogic: Logical reasoning explanations with fine-grained structure. *EMNLP*.

Seyed Mehran Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. 2022. LAMBADA: backward chaining for automated reasoning in natural language. *CoRR*, abs/2212.13894.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *CoRR*, abs/2210.02406.

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.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *CoRR*, abs/2205.11916.

Kaori Kumagai, Ichiro Kobayashi, Daichi Mochihashi, Hideki Asoh, Tomoaki Nakamura, and Takayuki Nagai. 2016. Human-like natural language generation using monte carlo tree search. In *Proceedings of the INLG 2016 Workshop on Computational Creativity in Natural Language Generation, CC-NLG 2016, Edinburgh, UK, September 2016*, pages 11–18. Association for Computational Linguistics.

Kaori Kumagai, Ichiro Kobayashi, Daichi Mochihashi, Hideki Asoh, Tomoaki Nakamura, and Takayuki Nagai. 2018. Natural language generation using monte carlo tree search. *J. Adv. Comput. Intell. Intell. Informatics*, 22(5):777–785.

Matthew Lamm, Jennimaria Palomaki, Chris Alberti, Daniel Andor, Eunsol Choi, Livio Baldini Soares, and Michael Collins. 2021. QED: A framework and dataset for explanations in question answering. *Trans. Assoc. Comput. Linguistics*, 9:790–806.

Andrew K. Lampinen, Ishita Dasgupta, Stephanie C. Y. Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L. McClelland, Jane X. Wang, and Felix Hill. 2022. Can language models learn from explanations in context? *CoRR*, abs/2204.02329.

Rémi Leblond, Jean-Baptiste Alayrac, Laurent Sifre, Miruna Pislar, Jean-Baptiste Lespiau, Ioannis Antonoglou, Karen Simonyan, and Oriol Vinyals. 2021. Machine translation decoding beyond beam search. In *Proceedings of the 2021 Conference on*

Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021, pages 8410–8434. Association for Computational Linguistics.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2022. RLET: A reinforcement learning based approach for explainable QA with entailment trees. *CoRR*, abs/2210.17095.

John McCarthy et al. 1960. *Programs with common sense*. RLE and MIT computation center Cambridge, MA, USA.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2381–2391. Association for Computational Linguistics.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. Webgpt: Browser-assisted question-answering with human feedback. *CoRR*, abs/2112.09332.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Danilo Neves Ribeiro, Shen Wang, Xiaofei Ma, Rui Dong, Xiaokai Wei, Henghui Zhu, Xinchi Chen, Peng Xu, Zhiheng Huang, Andrew O. Arnold, and Dan Roth. 2022. Entailment tree explanations via iterative retrieval-generation reasoner. In *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 465–475. Association for Computational Linguistics.

Sara Rosenthal, Mihaela A. Bornea, Avirup Sil, Radu Florian, and J. Scott McCarley. 2021. Do answers to boolean questions need explanations? yes. *CoRR*, abs/2112.07772.

Christopher D. Rosin. 2011. Multi-armed bandits with episode context. *Ann. Math. Artif. Intell.*, 61(3):203–230.

Soumya Sanyal, Harman Singh, and Xiang Ren. 2022. Fairr: Faithful and robust deductive reasoning over natural language. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1075–1093. Association for Computational Linguistics.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nat.*, 588(7839):604–609.

J Schulman, B Zoph, C Kim, J Hilton, J Menick, J Weng, JFC Uribe, L Fedus, L Metz, M Pokorny, et al. 2022. Chatgpt: Optimizing language models for dialogue.

Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. 2020. BLEURT: learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7881–7892. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4603–4611. PMLR.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of go without human knowledge. *Nat.*, 550(7676):354–359.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *CoRR*, abs/2210.09261.

Oyvind Tafjord and Peter Clark. 2021. General-purpose question-answering with macaw. *CoRR*, abs/2109.02593.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2022. Entailer: Answering questions with faithful and truthful chains of reasoning. *EMNLP*.

Mokanarangan Thayaparan, Marco Valentino, and André Freitas. 2020. A survey on explainability in machine reading comprehension. *CoRR*, abs/2010.00389.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *CoRR*, abs/2203.11171.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *CoRR*, abs/2206.07682.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.

Nathaniel Weir and Benjamin Van Durme. 2022. Dynamic generation of interpretable inference rules in a neuro-symbolic expert system. *CoRR*, abs/2209.07662.

Sarah Wiegreffe and Ana Marasovic. 2021. Teach me to explain: A review of datasets for explainable natural language processing. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Zhengnan Xie, Sebastian Thiem, Jaycie Martin, Elizabeth Wainwright, Steven Marmorstein, and Peter A. Jansen. 2020. Worldtree V2: A corpus of science-domain structured explanations and inference patterns supporting multi-hop inference. In *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*, pages 5456–5473. European Language Resources Association.

Kaiyu Yang, Jia Deng, and Danqi Chen. 2022. Generating natural language proofs with verifier-guided search. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Eric Zelikman, Yuhuai Wu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *CoRR*, abs/2203.14465.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed H. Chi. 2022a. Least-to-most prompting enables complex reasoning in large language models. *CoRR*, abs/2205.10625.

Hattie Zhou, Azade Nova, Hugo Larochelle, Aaron C. Courville, Behnam Neyshabur, and Hanie Sedghi. 2022b. Teaching algorithmic reasoning via in-context learning. *CoRR*, abs/2211.09066.

A Implementation Details

• **Retriever.** Given a positive pair (q, k^+), we train the retriever to maximize the cosine similarity between the query q and the positive fact k^+ in the embedding space. For an entailment tree in the EntailmentBank training split, we use the hypothesis as q and the leaf facts as k^+. The contrastive loss function we used is

$$\mathcal{L} = -\log \frac{\exp(\cos(q, k^+)/\tau)}{\sum_{i=0}^{bs} \exp(\cos(q, k_i)/\tau)}, \quad (8)$$

where bs is the batch size, k_i is the fact for the i pair in this batch, and τ is the temperature factor. We set τ to 0.02. The retriever is trained with a learning rate of 1e-5 and a batch size of 40 for 10k steps. For each retrieval, we return the top 25 facts. To update the premise set X, we keep all the intermediate conclusions int_* and replace all other sentences sent_* with the newly retrieved sentences. The query sentence is also added to X if it is not in X. If a query is retrieved consecutively, we scroll down the retrieval results and return the next 25 facts.

• **Entailment Module.** We follow Hong et al. (2022) to implement the entailment module in a prefixed manner. All types of modules are implemented with a single T5-large model (Raffel et al., 2020). A type-specific prefix (e.g., "deductive substitution:") is added to specify which type of reasoning the model should perform. The model is trained on the entailment steps of the EntailmentBank training split. The reason type labels of steps are provided by Hong et al. (2022). Following Yang et al. (2022), we take the hypothesis as additional input to encourage the module to generate a more relevant conclusion. The module is trained with a learning rate of 3e-5 and a batch size of 20 following Hong et al. (2022).

• **Controller.** For each iteration of the verifier-guided iterative training, we train the controller with a learning rate of 1e-5 and a batch size of 20 for ten epochs. We use the Adafactor optimizer (Shazeer and Stern, 2018). The input sequence of the controller is the concatenation of the hypothesis H, partial proof tree T_p, and context fact set X. The question and option are also included in the input. An example input is as follows:

$question$ Which will most likely cause a decrease in predator populations? $option$ a decrease in prey populations. $hypothesis$ a decrease of prey populations will decrease predator populations. $proof$ sent1 & sent2 -> int1 $context$ int1: a decrease of food has a negative impact on organisms. sent3: if an organism eats something then that something is a source of frood to that organism. sent4: negative impacts on organisms / species will decrease the population of the organisms / the species. . . . sent25: an adaptation has a positive impact on a living thing 's survival.

The action likelihood score is calculated by $\exp(\frac{1}{N}\sum_i \text{logit}(x_i|x_{k<i}))$, where x_i is the i-th token of the action sequence and N is the number of tokens in the action sequence.

• **Step Verifier V_s.** The human-labeled data provided by Tafjord et al. (2022) contains 1,827 valid steps and 1,564 invalid steps. We also use the 4,175 valid steps extracted from the gold trees in the EntailmentBank training split. For each valid step, we perturb it to make an invalid step by substituting one premise with a random distractor from the corpus. The step data is randomly divided into training and development splits in the ratio of 9:1. We train the step verifier with a learning rate of 1e-5 and a batch size of 16 for ten epochs. An example input of step verifier is as follows:

premises: a mushroom is a kind of fungus. in the food chain process fungi have the role of decomposer. conclusion: in the food chain process mushrooms have the role of decomposer.

In addition, for the sentence similarity scorer V_h in the state verifier, we use `BLEURT-Large-512` following Dalvi et al. (2021).

• **Entailment Tree Construction.** During reasoning, if adding a step makes the current tree not acyclic, then we consider this step invalid. For the final state, if the steps form a forest (containing multiple entailment trees), then we pick the tree with the highest hypothesis faithfulness score and discard the other trees.

• **Total Number of Learnable Parameters.** In our FAME, the numbers of learnable parameters of the controller, the retriever, the entailment module, and the step verifier are 770M, 109M, 770M, and 435M, respectively. In total, the number of learnable parameters of FAME is about 2,084M.

• **Experiment Environments.** We deploy all models on a server with four RTX 3090 GPUs. Each RTX 3090 GPU has 24G of memory. Our code mainly depends on python 3.8.0 and PyTorch 1.10.0. We use the pre-trained language models from `HuggingFace Transformers`[10].

[10] https://github.com/huggingface/transformers

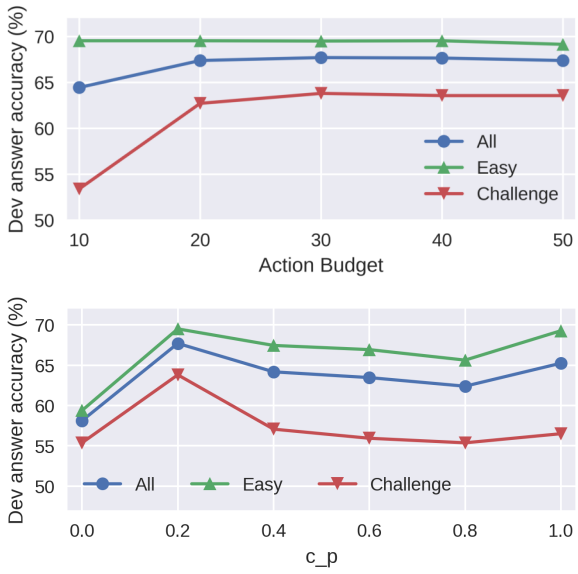


Figure 5: Results of Monte-Carlo planning algorithm with different parameters on EntailmentBankQA dev split.

• **Running time.** In our experiment environment, the average running time for each option is 30.77±5.93 seconds. This running time is acceptable. While not directly comparable, the running time for each option is about 90 seconds for NELLIE (Weir and Durme, 2022) and 20∼60 seconds for Entailer-11B (Tafjord et al., 2022), as reported in their papers.

B FAME Illustrations and Case Study

Given a question and candidate options, we first try to prove each option by generating an entailment tree (Figure 7). Then, we score each option and select the option with the highest score (Figure 8). Figure 9 shows some entailment trees and answers generated by our method on the EntailmentBankQA test split.

C Additional Experiment Results

C.1 Planning Algorithm Hyperparameter Analysis

The hyperparameters are selected using the dev split, as shown in Figure 5. We select an action budget of 30 and $c_p = 0.2$. We can also find that, as the action budget increases, the performance remains basically unchanged on easy questions but improves on challenge questions.

Beam size	Action budget	All	Easy	Chal
2	30	63.6	66.4	57.6
3	30	64.2	67.4	57.1
5	30	62.8	65.2	57.6
3	50	65.1	68.2	58.2
3	100	65.0	68.2	58.1

Table 10: Results of beam searches with different parameters on EntailmentBankQA dev split.

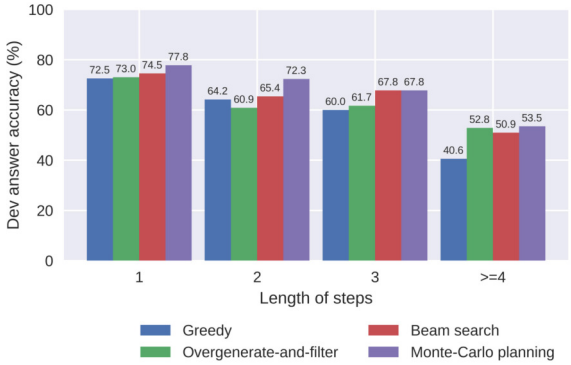


Figure 6: Dev results of different planning algorithms broken down by the length of the steps in the gold trees.

C.2 Beam Search Hyperparameter Analysis

Table 10 shows the beam search performances with different beam sizes and action budgets. In the case of an action budget of 30 (the same as other algorithms), a beam size of 3 achieves the highest answer accuracy. We also find that simply increasing the action budget brings limited performance gains, especially for the challenge questions.

C.3 Performances Breakdown

Figure 6 shows the performances of different planning algorithms on the EntailmentBankQA development split. We break down the results by the length of steps in the gold trees of the correct options. The results show that the questions become increasingly difficult as the length of steps increases. The Monte-Carlo planning algorithm achieves better results than other algorithms at all lengths.

C.4 Error Analysis

We randomly sample 50 questions that FAME answers incorrectly. We find the following three types of errors.

(1) Declarativization Error (32%). Our generation model that converts the question+option to the declarative hypothesis is not free from conversion errors. The conversion result might lack

some key information and thus not match the original question+option. For example, while the question+option is *The most likely reason sound travels faster in saltwater than in freshwater is that saltwater+is more elastic*, the generated hypothesis is *saltwater is likely to travel faster than freshwater*. Given the mistaken hypothesis, it is difficult for the controller to reason correctly.

(2) Invalid Reasoning Step (40%). One of the reasoning steps is invalid, leading to the incorrect conclusion. A step is invalid if its conclusion does not follow from the premises, e.g., the conclusion is in conflict with or irrelevant to the premises. An example of an invalid step is *planting trees has a positive impact on an environment & negative impact is the opposite of positive impact -> planting a tree has a negative impact on the environment*.

(3) Unsupported Option (28%). The final conclusion of the reasoning steps is not relevant to the selected option or does not contain enough information to support the selected option, even if the reasoning steps are valid. For example, the hypothesis is *granite can be used to study the history of organisms*, while the final conclusion of the reasoning steps is *granite can be used to study the history of rocks on earth*.

D Prompts for GPT-3 and ChatGPT

Figure 10 shows an example prompt for GPT-3. We randomly sample six examples (to fit the maximum input of GPT-3) from the training split of EntailmentBank (Task 2). We use the Chain-of-Thought prompting (Wei et al., 2022b) by using the entailment tree as the intermediate thought. For the test example, we retrieve 25 facts from the corpus using the question as the query. We use the same retriever as our method.

Figure 11 shows a dialogue with ChatGPT. We introduce the definition of the task in detail and guide the model to respond in the desired form. We manually try several kinds of prompts and chose the best one. For example, we try to include several in-context examples in the prompt as we do for GPT-3, but it doesn't seem to yield better results.

E Automatic Evaluation Metrics on EntailmentBank

For the entailment tree generation task, we evaluate on EntailmentBank using their automatic evaluation metrics (Dalvi et al., 2021). Denote the predicted entailment tree as $\hat{T}$ and the gold one as T. First, a tree alignment algorithm is performed to align nodes of $\hat{T}$ to nodes of T. Leaf nodes are aligned based on their texts. Each non-leaf node of $\hat{T}$ is aligned to the first non-leaf node of T which has the largest Jaccard similarity of the leaf node. Then, we evaluate $\hat{T}$ with the following metrics:

• **Leaves**. The leaf nodes of $\hat{T}$ and T are compared to compute F1 score. The AllCorrect score is 1 if F1 is 1, 0 otherwise.

• **Step Structures**. To evaluate whether the steps of $\hat{T}$ are structurally correct, the steps of $\hat{T}$ and T are compared to compute the F1 score. A predicted step (corresponding to a non-leaf node of $\hat{T}$) is structurally correct if its children nodes perfectly match the gold ones. AllCorrect=1 if F1=1, 0 otherwise.

• **Intermediates**. To evaluate the predicted intermediate conclusions, the intermediate F1 score is computed by comparing the aligned intermediate conclusions. A predicted intermediate conclusion is considered correct if the `BLEURT-Large-512` score of the aligned conclusion pair is larger than 0.28. AllCorrect=1 if F1=1, 0 otherwise.

• **Overall AllCorrect**. Based on the above scores, the Overall AllCorrect score is 1 if and only if the AllCorrect scores of Leaves, Step Structures, and Intermediates are all 1.

Please refer to the EntailmentBank paper for more details.

Question: In photosynthesis, plants use chlorophyll to produce
Option: sugar
Hypothesis H: plants use chlorophyll to produce sugar in photosynthesis

s^0: H; $T_p = \emptyset$; $X = \emptyset$

a^1: `Retrieve`: H

s^1: H; $T_p = \emptyset$; $X = \{$

sent1: a chloroplast contains chlorophyll
sent2: photosynthesis is a kind of chemical process
sent3: photosynthesis is a kind of process
sent4: leaves contain chlorophyll
sent5: chlorophyll is used for absorbing light energy by plants
sent6: photosynthesis makes food for the plant by converting carbon dioxide, water, and sunlight into carbohydrates
sent7: chlorophyll is a kind of pigment
sent8: photosynthesis is a source of / makes food / energy for the plant by converting carbon dioxide, water, and sunlight into carbohydrates
sent9: photosynthesis is a source of food for the plant by converting carbon dioxide, water, and sunlight into carbohydrates
sent10: carbohydrates are made of sugars
sent11: photosynthesis makes energy for the plant by converting carbon dioxide, water, and sunlight into carbohydrates
sent12: photosynthesis means producers / green plants convert from carbon dioxide and water and solar energy into carbohydrates and food and oxygen for themselves
sent13: a plant cell contains chloroplasts
sent14: chlorophyll is green in color
sent15: sugar is a kind of food
sent16: green plants perform photosynthesis to provide food for themselves
sent17: a producer is a source of sugar in an ecosystem
sent18: photosynthesis is a source of energy for the plant by converting carbon dioxide, water, and sunlight into carbohydrates
sent19: photosynthesis means green plants convert carbon dioxide and water and solar energy into carbohydrates and food and oxygen for themselves
sent20: green plant is a kind of plant
sent21: photosynthesis means green plants convert from carbon dioxide and water and solar energy into carbohydrates and food and oxygen for themselves
sent22: plants perform photosynthesis
sent23: green plants are a kind of plant
sent24: chlorophyll is a part of chloroplasts
sent25: a plant is a kind of organsim }

a^2: `Entail`: sent6 & sent10

s^2: H; $T_p = \{$sent6 & sent10 → int1$\}$; $X = \{$

int1: photosynthesis converts carbon dioxide, water, and sunlight into sugar for the plant
sent1: a chloroplast contains chlorophyll
……
sent5: chlorophyll is used for absorbing light energy by plants
sent7: chlorophyll is a kind of pigment
sent8: photosynthesis is a source of / makes food / energy for the plant by converting carbon dioxide, water, and sunlight into carbohydrates
sent9: photosynthesis is a source of food for the plant by converting carbon dioxide, water, and sunlight into carbohydrates
sent11: photosynthesis makes energy for the plant by converting carbon dioxide, water, and sunlight into carbohydrates
……
sent25: a plant is a kind of organsim }

a^3: `Retrieve`: int1

s^3: H; $T_p = \{$sent6 & sent10 → int1$\}$; $X = \{$

int1: photosynthesis converts carbon dioxide, water, and sunlight into sugar for the plant
sent1: photosynthesis makes energy for the plant by converting carbon dioxide, water, and sunlight into carbohydrates
sent2: a leaf converts carbon dioxide, water, and sunlight into carbohydrates
sent3: chlorophyll is a part of a plant used to perform photosynthesis
sent4: photosynthesis means green plants convert carbon dioxide and water and solar energy into carbohydrates and food and oxygen for themselves
sent5: photosynthesis means green plants convert from carbon dioxide and water and solar energy into carbohydrates and food and oxygen for themselves
sent7: photosynthesis is a kind of chemical process
sent8: photosynthesis is a kind of process
sent9: sugar is a kind of simple carbohydrate
sent11: photosynthesis makes food for the plant
sent12: photosynthesis means producers / green plants convert from carbon dioxide and water and solar energy into carbohydrates and food and oxygen for themselves
sent13: photosynthesis is a source of energy for the plant by converting carbon dioxide, water, and sunlight into carbohydrates
sent14: sugar is a kind of food
sent15: photosynthesis means green plants convert from solar energy into carbohydrates and food and oxygen
sent16: photosynthesis is a source of / makes food / energy for the plant by converting carbon dioxide, water, and sunlight into carbohydrates
sent17: photosynthesis is a source of food for the plant by converting carbon dioxide, water, and sunlight into carbohydrates
sent18: sugar is made of carbon, hydrogen, oxygen
sent19: sugar is a kind of pure substance
sent20: table sugar is a kind of sugar
sent21: a plant requires sunlight for photosynthesis
sent22: plants perform photosynthesis
sent23: photosynthesis is a kind of chemical reaction
sent24: sugar is made of sugar crystals
sent25: sugars are transported from the leaves to the roots of a plant }

a^4: `Entail`: int1 & sent3

s^4: H; $T_p = \{$sent6 & sent10 → int1; int1 & sent3 → int2$\}$; $X = \{$

int2: plants use chlorophyll to produce sugar in photosynthesis
sent1: photosynthesis makes energy for the plant by converting carbon dioxide, water, and sunlight into carbohydrates
sent2: a leaf converts carbon dioxide, water, and sunlight into carbohydrates
sent3: chlorophyll is a part of a plant used to perform photosynthesis
sent4: photosynthesis means green plants convert carbon dioxide and water and solar
……
sent25: sugars are transported from the leaves to the roots of a plant }

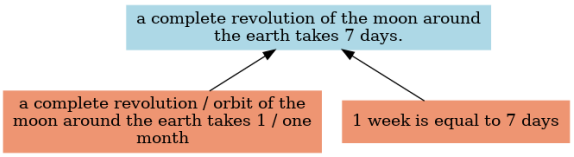
a^5: `End`: proved

Figure 7: An illustration of the reasoning process of FAME. For a question and the option that we want to prove, we first convert the question+option to a hypothesis H. We start from the initial reasoning state where the partial tree $T_p = \emptyset$ and the candidate premises $X = \emptyset$. In each interaction, we execute an action and update the reasoning state. For the action `Retrieve`, we sent the query to the retriever and update X with the retrieval results. For the action `Entail`, we sent the selected premises to the entailment modules. The novel conclusion is added to X and the novel step is added to T_p. For the action `End`, we end the reasoning process.

Question: About how long does it take for the Moon to complete one revolution around Earth?
Options: (A) 7 days (B) 30 days (C) 90 days (D) 365 days

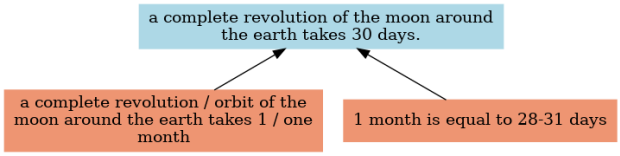
$V(s) = (\texttt{Valid}(T_p) + \texttt{Faithful}\ (T_p, H))/2$
$P(s) = P(s, \texttt{End}{:}\text{proved})$
$score(o) = (V(s) + P(s))/2$

Option o_1: 7 days
Hypothesis H_1: a complete revolution of the moon around the earth takes 7 days
Entailment Tree:



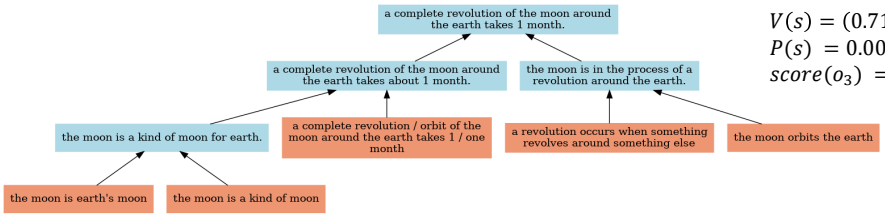
$V(s) = (0.0009 + 1.0000)/2 = 0.5005$
$P(s) = 0.0084$
$score(o_1) = 0.2544$

Option o_2: 30 days
Hypothesis H_2: a complete revolution of the moon around the earth takes 30 days
Entailment Tree:



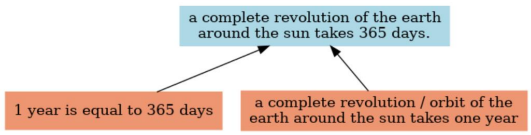
$V(s) = (0.9884 + 1.0000)/2 = 0.9942$
$P(s) = 0.9347$
$score(o_2) = 0.9645$ ✓

Option o_3: 90 days
Hypothesis H_3: a complete revolution of the moon around the earth takes 90 days
Entailment Tree:



$V(s) = (0.7107 + 0.1792)/2 = 0.4450$
$P(s) = 0.0070$
$score(o_3) = 0.2260$

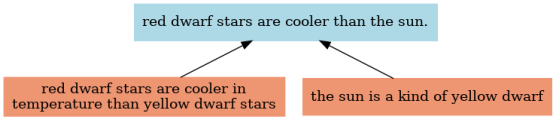
Option o_4: 365 days
Hypothesis H_4: a complete revolution of the moon around the earth takes 365 days
Entailment Tree:



$V(s) = (0.3532 + 0.7930)/2 = 0.5731$
$P(s) = 0.0076$
$score(o_4) = 0.2904$

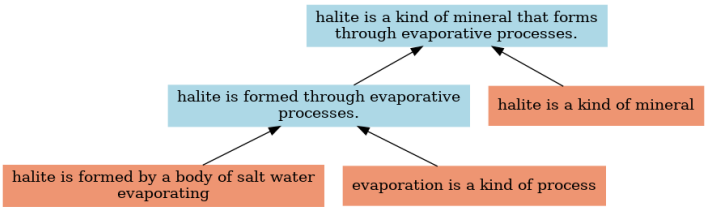
Figure 8: An example of the option selection. For each option, we try to generate an entailment tree to prove the option. We select the option based on the state verifier score $V(s)$ and the controller score $P(s)$.

(a) **Question**: When compared to the Sun, red dwarf stars are
Options: (A) older. (B) cooler. (C) lower density. (D) larger diameter.
Entailment Tree:



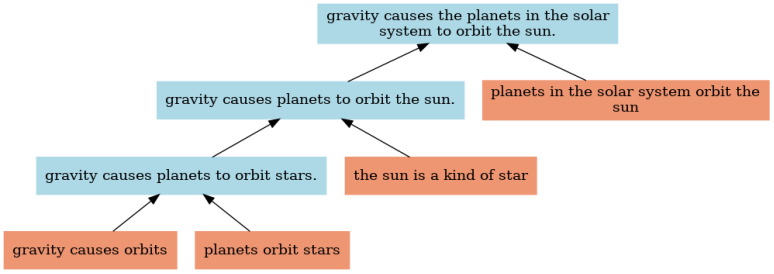
Answer: cooler

(b) **Question**: Which is a mineral that forms through evaporative processes?
Options: (A) halite (B) silver (C) gold (D) quartz
Entailment Tree:



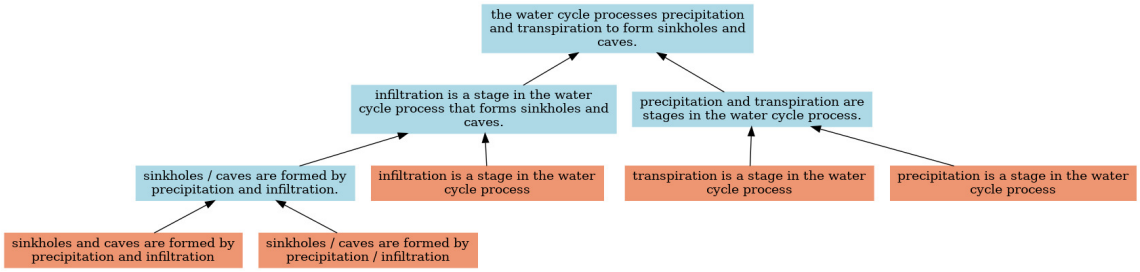
Answer: halite

(c) **Question**: Planets in the solar system are in constant motion. What factor has the greatest effect on the orbits of the planets?
Options: (A) the size of the planets (B) gravitational pull of the Sun (C) the composition of the planets (D) electromagnetic radiation from the Sun
Entailment Tree:



Answer: gravitational pull of the Sun

(d) **Question**: Some sinkholes and caves are created when water dissolves certain rocks and minerals below ground. Which two parts of the water cycle are most directly responsible for the formation of sinkholes and caves?
Options: (A) evaporation and infiltration (B) evaporation and transpiration (C) precipitation and infiltration (D) precipitation and transpiration
Entailment Tree:



Answer: precipitation and transpiration

Figure 9: Reasoning steps and answers generated by FAME. The correct options are indicated by underlining. In cases (a), (b), and (c), FAME can generate valid entailment trees and eventually select the correct options. In case (d), FAME selects the wrong option. The causes of the selection errors are traceable. Some steps in the entailment tree are invalid, leading to the wrong reasoning and the wrong option.

Question: New vaccines for diseases are being developed all the time. How do vaccines most likely help people?
Options: (A) Vaccines help prevent illnesses. (B) Vaccines keep the environment cleaner. (C) Vaccines help cure people who are sick. (D) Vaccines kill bacteria that cause infections.
Context:
sent1: something is used for that something 's purpose
…
sent25: as the amount of a source of something decreases , the amount of that something will decrease
Steps: sent17 & sent19 -> int1: decreasing illness has a positive impact on an organism's health; int1 & sent7 -> int2: preventing illness has a positive impact on an organism's health; int2 & sent10 -> int3: vaccines have a positive impact on an organism's health by helping to prevent illness; int3 & sent3 -> hypothesis: vaccines help organisms by helping to prevent illness;
Answer: (A)

Question: When the Northern Hemisphere is tilted toward the Sun, what season is occurring in Australia?
Options: (A) fall (B) winter (C) spring (D) summer
Context:
sent1: being in the sun is synonymous with being in the sunlight
…
sent25: distance is a property of space and includes ordered values of close / far
Steps: sent21 & sent4 -> int1: the northern hemisphere is in summer; int1 & sent11 -> int2: the southern hemisphere is in winter; int2 & sent17 -> hypothesis: australia is in winter;
Answer: (B)

Question: Grooves running down the sides of a riverbank are most likely caused by
Options: (A) wind. (B) rocks. (C) leaves. (D) rainwater.
Context:
sent1: weathering occurs at the the surface of the earth
..
sent25: damp means a large amount of water
Steps: sent20 & sent5 -> int1: soil erosion caused by water will cause grooves in soil; int1 & sent22 -> int2: rainwater can cause grooves in soil; int2 & sent24 -> hypothesis: rainwater can cause grooves in a riverbank;
Answer: (D)

Question: When hunting by humans causes a species to become extinct, this may produce damaging effects throughout the ecosystem of the extinct species. What is the cause of this damage?
Options: (A) alteration of a food web (B) degradation of a habitat (C) modification of a climate (D) reversal of a flow of energy
Context:
sent1: if a living thing dies then that living thing is dead
…
sent25: an ecosystem is a kind of habitat
Steps: sent12 & sent15 -> hypothesis: extinctions of organisms can cause damage to an ecosystem by changing the food web;
Answer: (A)

Question: Which of the following activities is the best example of instinctive behavior in an animal?
Options: (A) A dog sits when told to sit by its owner. (B) A bird avoids an insect that has a bad taste. (C) A newly hatched sea turtle walks toward the ocean. (D) A chimpanzee uses a stick to pull termites from a tree stump.
Context:
sent1: an organism is a living thing
…
sent25: actions mean behavior
Steps: sent2 & sent24 & sent4 -> int1: a sea turtle is an animal that walks toward the ocean after it is born; int1 & sent15 -> hypothesis: walking toward the ocean is sea turtles' instinctive behavior;
Answer: (C)

Question: Which of the following would most likely happen if grasses and shrubs were removed from a rural Massachusetts ecosystem?
Options: (A) There would be an increase in consumers in the ecosystem. (B) There would be an increase of photosynthesis in the ecosystem. (C) There would be a decrease in food energy produced by the ecosystem. (D) There would be a decrease of carbon dioxide available to the ecosystem.
Context:
sent1: soil contains nutrients for plants
…
sent25: energy flows in an ecosystem through food chains
Steps: sent2 & sent8 -> int1: grass and shrubs are kinds of plants; int1 & sent11 -> int2: grass and shrubs are kinds of producers; sent22 & sent7 -> int3: if a source of something is eliminated then the amount of that something that is produced decreases; int3 & sent17 -> int4: if a producer is eliminated then the amount of food energy produced in an ecosystem decreases; int2 & int4 -> int5: if grass and shrubs are eliminated then the amount of food energy produced in that ecosystem decreases; int5 & sent9 -> hypothesis: if grass and shrubs are removed then the amount of food energy produced in that ecosystem decreases;
Answer: (C)

Question: About how long does it take Earth to make one revolution around the Sun?
Options: (A) a day (B) a week (C) a month (D) a year
Context:
sent1: a complete revolution / orbit of the earth around the sun takes one year
…
sent25: the sun transfers energy from itself to the earth through sunlight
Steps:

Figure 10: A Chain-of-Thought prompt for GPT-3. The prompt consists of six in-context examples (with entailment trees and correct answers) and the test example (without the entailment tree and correct answer). For simplicity, we show only two facts for each example.

I want you to act as a QA system. I will give you a question, several options, and the context. I want you to answer the question and show the reasoning steps. A reasoning step uses sentences from the context as premises and obtains a new conclusion. The conclusion should be a valid entailment of the premises. The final answer should faithfully follow the reasoning steps.

An example of the context is
"sent1: dead organisms are the source of nutrients for decomposers
sent2: dead organisms rot
sent3: a mushroom is a kind of living thing
sent4: decomposers perform decomposition
sent5: in the food chain process mushrooms have the role of decomposer"
Here, sent1 etc. are the sentence id from the context.

Your response should be in the following format.
"Steps: sent1 & sent2 -> int1: rotting organisms are a source of nutrients for decomposers; int1 & sent5 -> hypothesis: rotting organisms are a source of food for mushrooms;
Answer: (A)"
Here, int1 is the intermediate conclusion. hypothesis is the final conclusion that can support the answer. The premises consist of at least two sentences. You don't need to use all the sentences in the context.

Now answer the following question.
Question: Melinda learned that days in some seasons have more daylight hours than in other seasons. Which season receives the most hours of sunlight in the Northern Hemisphere?
Options: (A) fall (B) spring (C) summer (D) winter
Context:
sent1: when a hemisphere is tilted towards the sun , that hemisphere receives more direct sunlight
sent2: the equator receives the most amount of direct sunlight throughout the year
sent3: when a hemisphere is tilted away from the sun , that hemisphere receives less direct sunlight
sent4: if a place is in summer, then it will have the most sunlight
sent5: a hemisphere is a part of earth
sent6: the northern hemisphere is a kind of hemisphere of earth
sent7: if something receives sunlight, it will absorb the sunlight
sent8: to receive sunlight means to absorb sunlight
sent9: summer has the most sunlight
sent10: the equator receives more sunlight than the poles on earth
sent11: the earth being tilted on its axis causes one side of the earth / one hemisphere to receive less energy from the sun than the other side
sent12: the poles receives the least amount of direct sunlight throughout the year
sent13: a hemisphere of earth is a kind of place
sent14: if something is outside during the day then that something will receive sunlight
sent15: if something receives sunlight, then it will increase in temperature
sent16: inheriting is when an inherited characteristic is passed down from parent to offspring by genetics / dna
sent17: the amount of daylight is greatest in the summer
sent18: different seasons occur during different times of the year
sent19: if something receives sunlight, it will increase in temperature
sent20: when sunlight strikes something , that something receives sunlight
sent21: a hemisphere is a kind of place
sent22: a learned characteristic is a kind of acquired characteristic
sent23: objects that are more exposed to sunlight are ideal for absorbing sunlight
sent24: when the season changes , the amount of daylight will change
sent25: the southern hemisphere is a kind of hemisphere of earth

Steps: sent1 & sent6 -> int1: the Northern Hemisphere is tilted towards the sun and receives more direct sunlight; int1 & sent4 -> hypothesis: the Northern Hemisphere receives the most sunlight during summer;
Answer: (C)

Figure 11: A dialogue example with ChatGPT. We introduce the task in detail. ChatGPT can respond in the desired form.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
This "limitations" section is at the end of the paper, after the conclusions section and before the references.

☑ A2. Did you discuss any potential risks of your work?
This "limitations" section.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Introduction.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 5 and Appendix

☑ B1. Did you cite the creators of artifacts you used?
Section 5 and Appendix

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
supplemental material

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Section 5 and Appendix

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
supplemental material

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 5 and Appendix

C ☑ Did you run computational experiments?

Section 5,6 and Appendix

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 5 and Appendix

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 5 and Appendix

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 5 and Appendix

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 5 and Appendix

D ☑ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Section 6

☑ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
supplemental material

☑ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
supplemental material

☑ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
supplemental material

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.