Step Feasibility-Aware and Error-Correctable Entailment Tree Generation

Junyue Song[1,2], **Xin Wu**[1,2], **Yi Cai**[1,2,*]
[1]School of Software Engineering, South China University of Technology
[2]Key Laboratory of Big Data and Intelligent Robot
(South China University of Technology) Ministry of Education
sejysong@mail.scut.edu.cn, sexinw@mail.scut.edu.cn, ycai@scut.edu.cn

Abstract

An entailment tree is a structured reasoning path that clearly demonstrates the process of deriving hypotheses through multiple steps of inference from known premises. It enhances the interpretability of QA systems. Existing methods for generating entailment trees typically employ iterative frameworks to ensure reasoning faithfulness. However, they often suffer from the issue of false feasible steps, where selected steps appear feasible but actually lead to incorrect intermediate conclusions. Moreover, the existing iterative frameworks do not consider error-prone search branches, resulting in error propagation. In this work, we propose SPEH: an iterative entailment tree generation framework with Step fesibility Perception and state Error Handling mechanisms. Step Feasibility Perception enables the model to learn how to choose steps that are not false feasible. State Error Handling includes error detection and backtracking, allowing the model to correct errors when entering incorrect search branches. Experimental results demonstrate the effectiveness of our approach in improving the generation of entailment trees.

Keywords: Explainability, Entailment tree, Natural language reasoning

1. Introduction

The Question-Answering (QA) task provides an effective way to test the natural language reasoning abilities of AI systems (Yang et al., 2018). Previous QA research has often employed end-to-end approaches to predict answers, lacking explainability by not showcasing the internal reasoning process (Thayaparan et al., 2020). Providing the reasoning process alongside the answers helps users understand and trust the process of answer generation (Wiegreffe and Marasovic, 2021). In certain specialized domains such as engineering or medicine, it is essential for the model to generate decisions that are trustworthy and reliable, necessitating a more rigorous and interpretable reasoning process (Miller, 2019; Papagni et al., 2023; Kim et al., 2023).

Dalvi et al. (2021) proposed a task of entailment tree generation to enhance the interpretability of QA systems, as illustrated in Figure 1 (a). This task requires the model to construct an entailment tree from a given question and background knowledge, representing the multi-step reasoning process in a tree structure. The leaf nodes of the tree represent known and validated background knowledge, the intermediate nodes represent intermediate conclusions derived through inference, and the root node represents the target of the reasoning process.

Existing methods such as METGEN (Hong et al., 2022) and SI (Creswell and Shanahan, 2022) typically adopt an iterative framework (Tafjord et al., 2021) in which each iteration performs one round of inference to ensure the faithfulness of reasoning

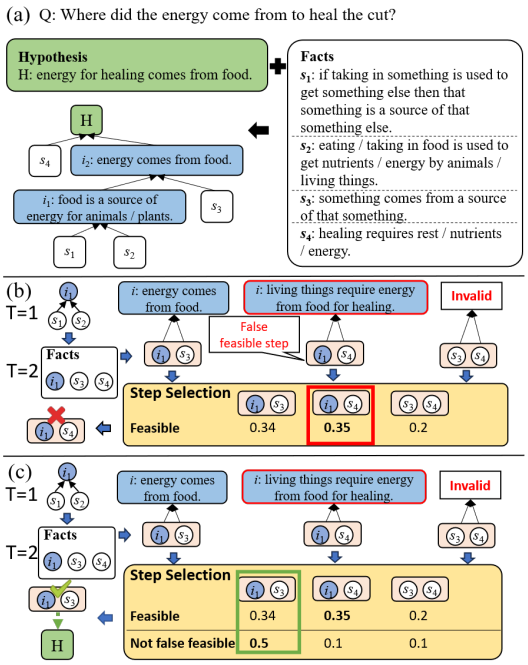

Figure 1: **(a)** An instance of Entailment Tree. **(b)** False feasible step: The premises may be incorrectly combined, for example, i_1&s_4 could consume the "source" needed for subsequent reasoning and result in an incorrect intermediate conclusion. **(c)** With the inclusion of our step feasibility perception mechanism, it becomes possible to determine which steps are more helpful for reasoning.

(Sanyal et al., 2022). This means that the generation of intermediate conclusions is not affected by irrelevant information. However, due to the model's selection of more feasible steps in each iteration,

*Corresponding authors

LREC-COLING 2024, pages 15296–15308

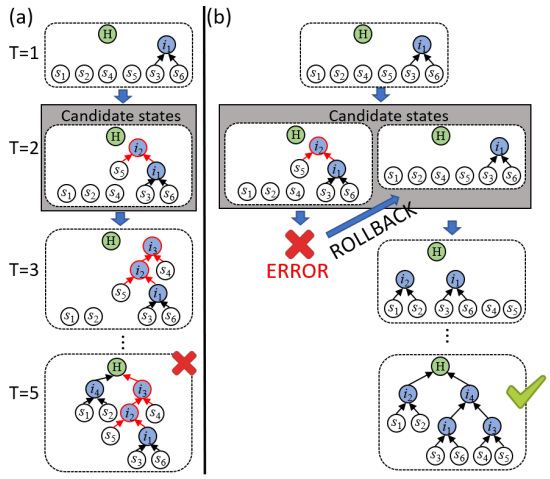

Figure 2: **(a)** Erroneous search branch: Existing iterative frameworks are unable to handle errors, which can result in error propagation. **(b)** SPEH can detect errors and roll back to the correct state.

this approach may encounter the issue of false feasible steps. In other words, the model may incorrectly choose steps that appear to be feasible but fail to combine the generated intermediate conclusions with other premises in subsequent steps. This can impede the progress of reasoning or lead to incorrect entailment trees.

As shown in Figure 1 (b), both i_1 & s_3 and i_1 & s_4 are feasible, but the latter produces an intermediate conclusion that lacks "source", preventing further combination into a feasible step and resulting in an incorrect entailment tree. Therefore, a method is needed to determine whether a step is true feasible or false feasible. Based on our observation, steps with smaller distances to the hypothesis tend to have fewer intermediate steps between the generated intermediate conclusion and the hypothesis. Consequently, their likelihood of being a false feasible step decreases. Hence, we consider constructing training samples based on distance and enabling the model to predict the probability of a step not being a false feasible step.

Furthermore, errors generated during iterations can lead to error propagation. In an iterative framework, once a decision is made and a search branch is determined, there is no mechanism to detect or address the errors that have already occurred. As shown in Figure 2 (a), in the $T = 2$ iteration, the erroneous states in the Candidate states are not detected, resulting in an incorrect entailment tree being generated. Inspired by Regret aversion theory (Loomes and Sugden, 1982), if there is a method to detect errors and allow for rollback, the model can revert to a state where the erroneous decision was not taken, thereby reducing error propagation.

In this paper, we propose SPEH: an iterative entailment tree generation framework with Step feasibility Perception and state Error Handling mechanism. The Step Feasibility Perception method selects steps based on the probability of each step being a false feasible step. We introduce a flexible sample construction method similar to hard negative samples (Robinson et al., 2021) based on metric learning. By utilizing the distance between steps and the hypothesis, additional training data is constructed to further differentiate positive samples, leading the model to select steps with a lower likelihood of being false feasible, as shown in Figure 1 (c). This helps generate more effective intermediate conclusions.

The State Error Handling mechanism incorporates states from previous iterations during the state filtering process in the reasoning procedure. The two-stage error detection process enables the identification of erroneous states, followed by the implementation of the proposed error rollback mechanism to return to a valid state, as depicted in Figure 2 (b). This effectively salvages search branches that could have potentially led to erroneous outputs.

We make three main contributions in this research:

- To enhance the effectiveness of entailment tree construction for each reasoning step, we propose a step feasibility perception approach that reduces the issue of false feasible step selection by the model.
- We introduce state error handling, a special rollback mechanism that introduces different iterations of states during reasoning to facilitate error rollback and reduce error propagation.
- Experimental results demonstrate that our innovative SPEH outperforms the baseline model in most metrics and effectively addresses the problems we are concerned about.

2. Related Works

PLM Pre-trained language models (PLMs) are a common approach in natural language processing, and there are several notable works in this area, including GPT (Radford et al.), BERT (Devlin et al., 2019), and RoBERTa (Liu et al., 2019). These models, based on the transformer architecture (Vaswani et al., 2017), follow a two-stage training process: pre-training and fine-tuning. This allows them to generate outputs in an end-to-end fashion, even for moderately complex tasks. However, there have been studies that raise concerns about the reasoning capabilities of PLMs, suggesting that their performance stems mainly from data correlation. These studies propose diagnostic reasoning tasks to assess the reasoning abilities of PLMs (Sugawara et al., 2020; Tian et al., 2021).

Interpretability of NLP In Natural Language Processing (NLP) tasks that involve multi-step reasoning, the credibility of the answers may be questioned if the process lacks explanations. Some research has introduced QA datasets with explanations, where the sources of answers are annotated within the text paragraphs (Yang et al., 2018; Ye et al., 2020; DeYoung et al., 2020; Xie et al., 2020). However, this approach fails to showcase the precise process from the given information to the answers. There are also studies that express reasoning chains in natural language (Ho et al., 2023; Lamm et al., 2021; Jhamtani and Clark, 2020; Wei et al., 2022), but this unstructured process cannot ensure that the information used in the generated sentences originates from the provided knowledge.

Some research has begun to leverage traditional single-step logical reasoning to provide assurances for interpretability (Bostrom et al., 2021; Sprague et al., 2023; Yang et al., 2022b; Young et al., 2022). To explore longer reasoning chains, Clark et al. (2020) introduced RuleTaker based on multi-step deductive reasoning. Additionally, Dalvi et al. (2021) further proposed the EntailmentBank dataset for generating entailment trees. This structured reasoning chain explicitly showcases how multi-step reasoning is performed from the background knowledge to arrive at answers. Due to the high quality of the annotated data in EntailmentBank, some researchers have expanded its utilization to other tasks and datasets (Sprague et al., 2022; Huang et al., 2022).

Recently, there has been increasing attention on Faithful QA (FQA), where faithfulness refers to a system that exhibits reasoning consistent with the standard definition of logical validity (Creswell and Shanahan, 2022). The objective of Tafjord et al. (2022); Weir and Van Durme (2022); Hong et al. (2023) is to ensure that the answers produced by the model are entirely dependent on its reasoning process. However, the current FQA tasks are mostly accomplished through answer enumeration and entailment tree generation attempts. Therefore, in our research, we focus on tackling the task of entailment tree generation.

Generation of Entailment Tree In the past, research on entailment trees often utilized the PLM approach for selection and generation tasks. As proposed in ProofWriter (Tafjord et al., 2021), these models can be broadly categorized into two types. **All-At-Once** For example, Saha et al. (2020) and Sun et al. (2021) attempt to predict all variable information of the entailment tree at once, including the existence relationships of nodes and edges. While these models can capture global reasoning information and make full use of the contextual modeling capabilities of PLMs, they have gradually been abandoned due to their opaque process and lack of reliability. **Iterative** Sanyal et al. (2022) and Creswell and Shanahan (2022) divide the selection of premises and the generation of conclusions into two separate modules and then employ beam search methods to search the reasoning space. Liu et al. (2022) and Zhao et al. (2023) attempt to introduce future information into the current iteration to enhance the model's decision-making abilities. Yang et al. (2022a) do not differentiate between selection and generation modules but instead propose a verifier to guide the reasoning process.

Due to the interference caused by distractors in Task 2 and Task 3 of EntailmentBank, filtering out irrelevant content from the premises has become a challenging task. Approaches (Bostrom et al., 2022; BehnamGhader et al., 2022; Ribeiro et al., 2022; Bogatu et al., 2022) have designed specialized retrieval methods to address this problem. Furthermore, to improve the efficiency and accuracy of reasoning, some studies have also considered the introduction of reverse reasoning (Liang et al., 2021; Qu et al., 2022; Hong et al., 2022).

3. Background

The task of EntailmentBank involves a given set of premises and a hypothesis (target), requiring multiple-step reasoning to reach the target and generate the entailment tree. We adopt the framework and representation proposed by Hong et al. (2022)

- **Hypothesis** The target to be proven in reasoning, represented as H.
- **Fact** Refers to the given premises containing background knowledge, represented as s.
- **Step** In each iteration, any two facts are combined to form a step, represented as p.
- **Int** After selecting a step, the single-step entailment module performs inference to obtain an intermediate conclusion, represented as i.
- **State** Represents a partial entailment tree structure composed of the remaining facts, intermediate conclusions, and the reasoning target, represented as R.

The model consists of two modules: the reasoning controller and the single-step entailment module. In each iteration, the reasoning controller first selects two suitable s for a deductive step. Alternatively, it constructs an abductive step using H and one s. The single-step entailment module then performs inference to obtain i and updates the R. The filtered state is outputted to the next iteration, and the process is repeated until no new intermediate conclusions are generated. The reasoning in the single-step entailment module can be categorized

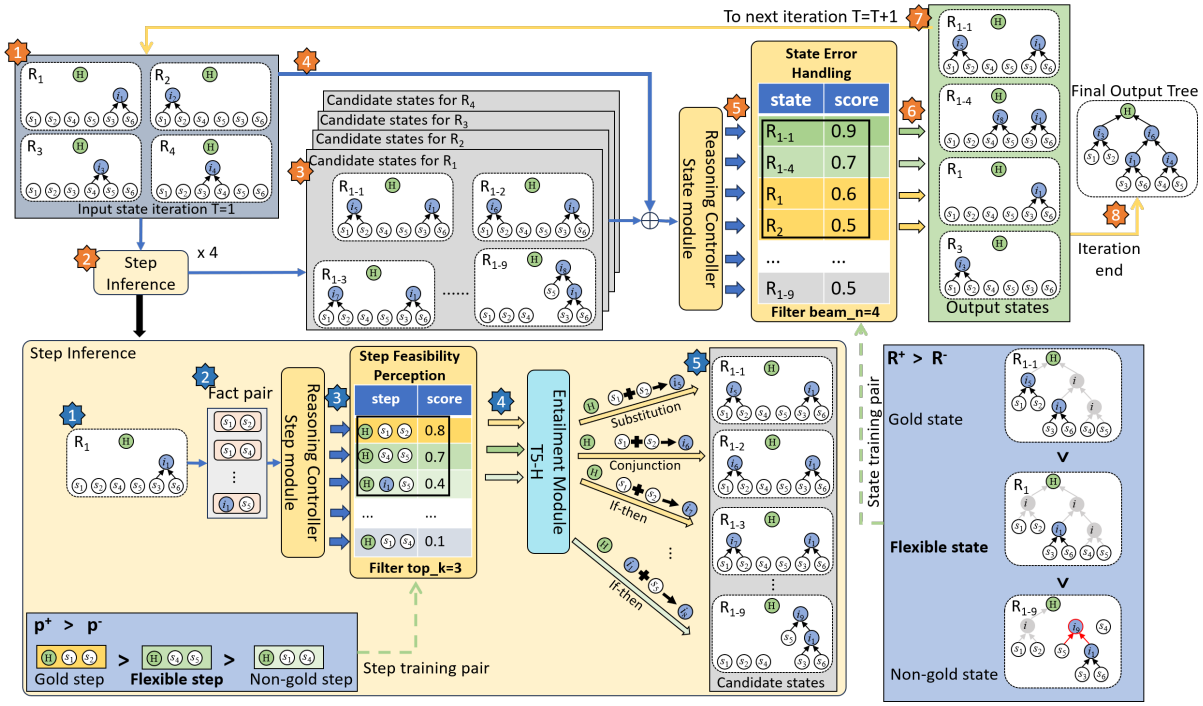

Figure 3: The reasoning process of SPEH. In step inference, the step module of the reasoning controller selects steps based on the step quality perception mechanism, excluding false feasible steps. The state module incorporates the state from the previous iteration to facilitate the detection and rollback of errors in the two-stage state error handling mechanism. The bottom-left and bottom-right corners illustrate the data construction process, where '+' denotes positive samples and '-' denotes negative samples.

into three types: Substitution, Conjunction, and If-then. Consequently, the same premises may yield different intermediate conclusions.

4. Our Methods

We propose SPEH: an iterative framework with Step feasibility Perception and state Error Handling mechanisms. The reasoning process of SPEH is illustrated in Figure 3, and the main steps are similar to those described in Section 3.

Step feasibility perception We enhance the training process to enable the model to choose steps in the step inference phase that are not false feasible.

State error handling We introduce a mechanism for handling state errors in the state filtering process, which allows for detecting and rolling back incorrect states to reduce the likelihood of generating erroneous entailment trees.

4.1. Step Feasibility Perception

In the original framework, the evaluation of steps did not consider the relationship between premises and the reasoning objective. It only took into account the logical consistency among premises, which could potentially lead to the derivation of irrelevant conclusions and be detrimental to subsequent reasoning. To address this issue, we propose modifying the step evaluation module to explicitly incorporate the features of the reasoning objective. This adjustment will enhance the ability to discern the relevance between each step and the objective.

$$G_{step}(s_i, s_j) = FFN_{ded}([h, f_i, f_j]) \quad (1)$$

Where h, f_i, and f_j represent the features of H, s_i, and s_j respectively, which are the outputs of the reasoning controller. In addition to providing hypotheses, we also guide the model to learn to select steps that are not false feasible. We discovered that it is possible to use the distance from a step to the hypothesis to filter out steps with a lower probability of being false feasible. The distance of a step is determined by the depth of the two premises it involves in the tree.

$$Distance(p) = (Depth(s_i) + Depth(s_j))/2 \quad (2)$$

The feasibility of a step depends not only on the compatibility of its premises but also on whether the generated intermediate conclusions can be further combined to form a feasible step. The shorter the distance between a step and the hypothesis, the fewer steps are required for the generated intermediate conclusions to reach the hypothesis. As a result, the likelihood of such a step being a false feasible step decreases. We construct specific training samples based on the distance to train the model

to assess whether a step is prone to deviate from the hypothesis, thereby facilitating the selection of more reliable and true feasible steps.

During the training process, we employ the margin ranking loss, defined as $\phi(x_1, x_2, m) = max(0, x_2 - x_1 + m)$, to facilitate the training of the step module in the reasoning controller. Our method constructs positive and negative training data in the form of (x_1, x_2), and divides the overall training data into two parts:

1. Preserve the original pairs of steps. Label the gold steps as p_1, and label the non-gold steps as p_2 to obtain (p_1, p_2). Train using the original loss.

$$L_{origin\_step} = \frac{1}{N_1} \sum_{(p_1,p_2)} \phi(G_{step}(p_1), G_{step}(p_2), m_{step}) \tag{3}$$

2. Pair the two gold steps together. Select the step with a smaller distance as the positive sample p_3, and the step with a larger distance as the negative sample p_4 to obtain (p_3, p_4). Train using an additional loss.

$$L_{ex\_step} = \frac{1}{N_2} \sum_{(p_3,p_4)} \phi(G_{step}(p_3), G_{step}(p_4), m_{step}) \tag{4}$$

Where N_1 and N_2 represent the number of step pairs, and m_{step} is the step margin. We refer to the gold steps that are farther apart as flexible steps. This is because they serve as positive samples in the first part of the data but as negative samples in the second part. It is not fixed, and an example of this data construction is illustrated in the bottom left corner of Figure 3.

The step loss consists of two components.

$$L_{step} = L_{origin\_step} + L_{ex\_step} \tag{5}$$

In step inference, all steps are sorted according to the G_{step}, and the top k steps are retained, where k is a hyperparameter.

4.2. State Error Handling

The state error handling mechanism primarily operates during the reasoning stage and consists of two processes: error detection and error rollback. Firstly, we utilize a state evaluation process to detect errors, which is further divided into two stages. In the first stage, an additional hard classification module is employed for rough filtering. In the second stage, the previous iteration's states are introduced as an additional reference state, and a state evaluation module is used to assess and rank the states to identify errors. Based on the final ranking results, we retain the states from the previous iteration that exhibit better performance as rollback states.

Let T represent the time step of the overall reasoning process, indicating the $T-th$ iteration. On the other hand, t represents the time step of a state, indicating that R_t has undergone t effective inferences. While Hong et al. (2022) maintained synchrony between T and t, where $t \equiv T$, we propose an asynchronous mechanism, allowing the same operation to be performed on states from different iterations within the same round of iteration. This facilitates the involvement of states from different iterations in the error detection process. Our training process ensures that the model can effectively evaluate states from different iterations simultaneously. Assuming G_{state} is the scoring function for states, this requires the state evaluation module to satisfy the following inequality constraints:

$$\begin{aligned} G_{state}(R^+_{t=T}) &> G_{state}(R^+_{t=T-1}) \\ &> G_{state}(R^-_{t=T||t=T-1}) \end{aligned} \tag{6}$$

4.2.1. Training

To improve the state module in the reasoning controller, we further divided it into two sub-modules: $G_{achievability}$ and G_{state}. For each of these, we defined separate loss functions and training data.

To identify significant errors, $G_{achievability}$ module is trained using binary cross-entropy (BCE) loss.

$$G_{achievability}(R) = \sigma(FFN_{cls}(f_{[cls]})) \tag{7}$$

$$L_{achievability} = BCE(G_{achievability}(R), state\_label) \tag{8}$$

Where $G_{achievability}$ has a value range of [0,1], and a $state\_label$ of 1 indicates that the state is a valid intermediate state in the gold tree, while a $state\_label$ of 0 indicates the opposite. We have made modifications to the G_{state} module to calculate the score of a state.

$$G_{state}(R) = \sigma(FFN_{cls}(f_{[cls]})) \tag{9}$$

$f_{[cls]}$ represents the cls feature output by the reasoning controller. This module is also trained using margin ranking loss. Hong et al. (2022) deconstructed the gold tree, using the gold state as the positive sample R_1 and the constructed non-gold states as the negative sample R_2 to obtain (R_1, R_2). We have retained this part of the data.

$$L_{origin\_state} = \phi(G_{state}(R_1), G_{state}(R_2), m_{state}) \tag{10}$$

m_{state} represents the state margin. To simulate the data distribution in inequation 6, we combine multiple states from different time steps into additional positive-negative sample pairs.

$$\begin{aligned} (R_3, R_4) = \{&(R^+_{t=T}, R^+_{t=T-1}), (R^+_{t=T}, R^-_{t=T-1}), \\ &(R^+_{t=T}, R^-_{t=T+1})\} \end{aligned} \tag{11}$$

An example of data construction can be seen in the bottom right corner of Figure 3. Similarly, we utilize flexible states that essentially correspond to gold states. Based on different combinations from equation 11, they become positive samples R_3 or negative samples R_4 in the sample pairs (R_3, R_4). Additionally, we employ an additional loss function for training purposes.

$$L_{ex\_state} = \phi(G_{state}(R_3), G_{state}(R_4), m_{state}) \tag{12}$$

$$L_{state} = L_{origin\_state} + L_{ex\_state} \tag{13}$$

The final loss is computed using:

$$L = L_{step} + L_{state} + L_{achievability} \tag{14}$$

4.2.2. Reasoning

During reasoning, the first step is to perform an inference step, which updates $R_{t=T}$ to $R_{t=T+1}$. Then, candidate states are filtered. Now, let's define the input set for T as $S_{in,T}$, and the output set as $S_{out,T}$. We have applied an asynchronous mechanism that allows states from different time steps to coexist within the same round of iteration.

$$S_{in,T} = S_{out,T-1} = \{R_{t=T-2}, R_{t=T-1}\} \tag{15}$$

When performing inference step, we handle all the states in the set collectively.

$$\begin{aligned} S_{inference,T} &= Inference(S_{in,T}) \\ &= \{R_{t=T-1}, R_{t=T}\} \end{aligned} \tag{16}$$

We retain a subset of states from $S_{in,T}$, denoted as $S_{hold,T} = \{R_{t=T-1} \in S_{in,T}\}$. From this, we obtain the candidate set:

$$\begin{aligned} S_{candidate,T} &= S_{inference,T} \cup S_{hold,T} \\ &= \{R_{t=T-1}, R_{t=T}\} \end{aligned} \tag{17}$$

In the subsequent two-stage filtering process, the model is able to evaluate both the pre-inference and post-inference states. In the first stage, the erroneous states are filtered out as follows: $S_{ach,T} = Filter_{ach}(S_{candidate,T})$. This process utilizes the median value M of all $G_{achievability}$ scores as a threshold. States with scores below this threshold are considered erroneous and eliminated.

In the second stage, a more granular sorting based on the G_{state} criterion is performed to select a $beam\_num$ number of states, resulting in the output: $S_{out,T} = Filter_{state}(S_{ach,T})$. As indicated by the previous inequation 6, for incorrect R_t, the previous iteration's state R_{t-1} contained in $S_{out,T}$ serves as a suitable rollback target.

5. Experiments

5.1. Dataset and Evaluation Metrics

We conducted experiments on Task 1 of the EntailmentBank dataset to evaluate our proposed model, SPEH. This dataset consists of a total of 1840 samples, each comprising a question-answer pair along with its corresponding entailment tree. The evaluation metrics include:

Leaves Evaluating the correctness of the premises involved in the entire reasoning process.

Steps Evaluating the correctness of the premises for each reasoning step.

Intermediates Evaluating the correctness of the intermediate conclusions for each reasoning step.

Overall Evaluating the extent to which the predicted tree matches the gold tree.

5.2. Baseline

We compared SPEH with several previous classic approaches.

EntailmentWriter (Dalvi et al., 2021) An All-at-once baseline model.

IRGR (Ribeiro et al., 2022) An architecture that combines retrieval and generation.

RLET (Yang et al., 2022a) The first iterative framework that incorporates reinforcement learning.

METGEN (Hong et al., 2022) A module-based iterative framework.

ChatGPT-3.5 We tested the results using the All-at-once approach in an in-context learning manner.

5.3. Experiment Setup

We adopted the settings from METGEN (Hong et al., 2022) and utilized albert-xxlarge-v2 (Lan et al., 2020) as the backbone for our reasoning controller. The single-step entailment module is based on the T5-large (Raffel et al., 2020) prefixed model provided by them, which was further fine-tuned by incorporating the corresponding hypothesis into the training input data.

During the training phase, we set the batch size to 5, learning rate to 1e-5, conducted 1000 epochs, and set all margins to 0.1. The weights for all losses were set to 1, and during the inference phase, we used a beam number of 4. In the error handling process, for each iteration, we only allowed comparison and rollback between the states of two adjacent time steps, referred to as a span of 1 for rollback. After testing, we found no differences in the results for other spans, so we continued to use a span of 1.

Method	Leaves		Steps		Intermediates		Overall
	F1	AllCorrect	F1	AllCorrect	F1	AllCorrect	AllCorrect
EntailmentWriter (T5-11B)△	99	89.4	51.5	38.2	71.2	52.9	35.6
ChatGPT-3.5	94	73.8	34.7	22.1	54.3	21.8	18.2
IRGR(T5-Large)△	97.6	89.4	50.2	36.8	62.1	31.8	32.4
RLET△	**100**	**100**	54.6	40.7	66.9	36.3	34.8
METGEN-prefixed△	**100**	**100**	**57.7**	41.9	70.8	39.2	36.5
SPEH-prefixed(ours)	**100**	**100**	57.4	**42.7**	**73.3**	**39.4**	**37.7**

Table 1: Entailment tree generation results on the EntailmentBank Task 1 test split. △ indicates results from published papers.

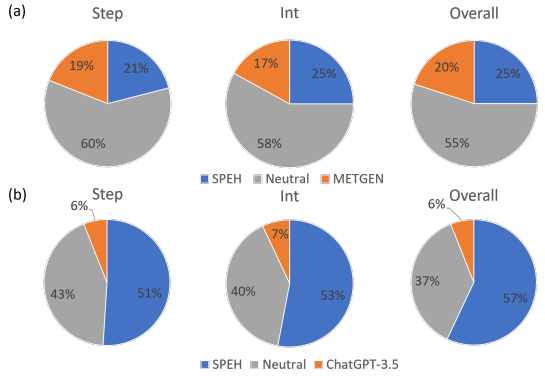

Figure 4: Comparison results of manual evaluation for 50 entailment tree samples. (a) Comparison between SPEH and METGEN. (b) Comparison between SPEH and ChatGPT-3.5.

Method	Steps	Intermediates	Overall
SPEH	**40.3**	**38.5**	**35.9**
w/o step feasibility	39.7	36.8	34.4
w/o state error	40.0	38.2	34.7

Table 2: Ablation results.

Method	Steps	Intermediates	Overall
METGEN-base	38.5	37.1	34.4
Feasibility(None)	**40.0**	35.6	34.1
Feasibility(Similarity)	36.8	36.5	34.1
Feasibility(Distance)	**40.0**	**38.2**	**34.7**

Table 3: Evaluation of constructing training data for different step feasibility construction methods. "None" indicates no additional data construction.

6. Result Analysis

6.1. Main Results

As shown in Table 1, SPEH outperforms all the compared baseline models. We achieved a 0.8% improvement in the Step AllCorrect metric compared to METGEN-prefixed. We speculate that the slight decrease in F1 may be due to poor performance on a small number of samples. However, SPEH resulted in improvements of 2.5%/0.2% in the intermediate F1 and AllCorrect metrics, respectively, when considering the combined effect of both methods. Please note that SPEH's framework, like METGEN-prefixed, is unable to handle steps that include n-premises. Approximately 26% of the test samples cannot be completely aligned with the gold tree. However, we still outperformed METGEN by 1.2% in the Overall AllCorrect metric, demonstrating the effectiveness of our approach.

6.2. Manual Evaluation

Since automatic evaluation metrics require aligning the predicted tree with the gold tree, it can lead to inaccurate assessments. Therefore, we conducted a certain amount of manual evaluation. We randomly selected 50 entailment trees generated by SPEH and the baseline models. Three graduate students were asked to evaluate which side performed better based on three aspects: Step feasibility(Step), Intermediate conclusion validity(Int), and Overall persuasiveness(Overall).

Our comparison results with METGEN are shown in Figure 4 (a), where we mostly achieved comparable evaluations across three metrics. We obtained a slight advantage in the Step metric, which may be attributed to the decline in step F1. In terms of the Int and Overall metrics, we obtained higher recognition due to the incorporation of SPEH, which effectively reduced the selection of false feasible steps and erroneous search branches, significantly enhancing our persuasiveness.

Considering that ChatGPT-3.5 is untrained and at a disadvantage in automated evaluation metrics, we also compared our method against ChatGPT-3.5 as another control group. However, 4 (b) indicates that while ChatGPT reaches a similar level to ours in some samples, we have overwhelming advantages in all three metrics. This is as expected for a language model without further fine-tuning.

6.3. Ablation Study

We present the results of ablative experiments in Table 2, where we replaced the backbone model with albert-base, incorporating our further fine-tuned single-step entailment module, and report the all-correct scores for step, intermediate, and overall. **Step Feasibility Perception** Upon removing step feasibility perception, all metrics showed a decrease, particularly noticeable in the intermediates category. Our analysis suggests that this mechanism enables the model to learn to select steps that are not false feasible, thereby improving not only step accuracy but also aiding in generating more effective intermediate conclusions. **States Error Handling** The impact of removing state error handling resulted in a relatively minor effect on the results, primarily reflected in the overall metric. As previously analyzed, this approach allows for comparing states before and after step inference to detect errors and perform rollbacks, thereby reducing the likelihood of errors and subsequently enhancing overall performance.

6.4. Step Feasibility

In the step feasibility perception method, there may be other criteria for determining whether a step is false feasible. Therefore, when constructing the training samples, we also tested both not creating additional data and using similarity as a comparison metric. We used the longest common substring as the similarity calculation, and in Table 3, we also reported the AllCorrect metric for Step, Int, and Overall. The performance decrease indicates that distance is a more suitable criterion.

6.5. Case Study

6.5.1. Step Feasibility Perception

The case presented in Figure 5 demonstrates the process of step feasibility perception. In this example's reasoning process, all three premises contain the term "object" indicating that it must be preserved throughout the inference process until the final step, where it is consumed to infer the hypothesis. The model initially determines that both s_1&s_2 and s_1&s_3 can generate valid intermediate conclusions. However, the former consumes the term "object" rendering the intermediate conclusion unable to form a feasible step. After undergoing step feasibility perception, the model further filters and identifies the true feasible step s_1&s_3 ensuring the correctness of the reasoning process.

6.5.2. State Errors Handling

The case presented in Figure 6 illustrates the process of handling state errors. At $T = 2$, the input

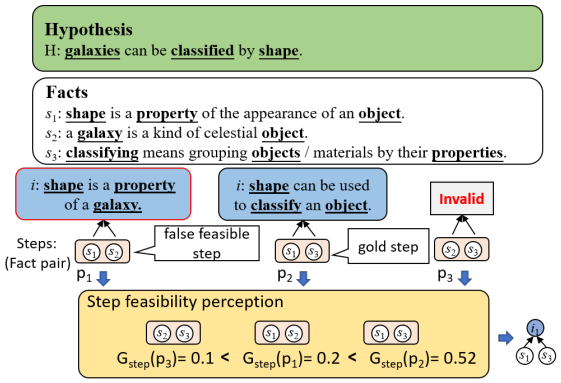

Figure 5: Case of step feasibility perception. G_{step} represents the scores outputted by step evaluation. Red indicates incorrect steps.

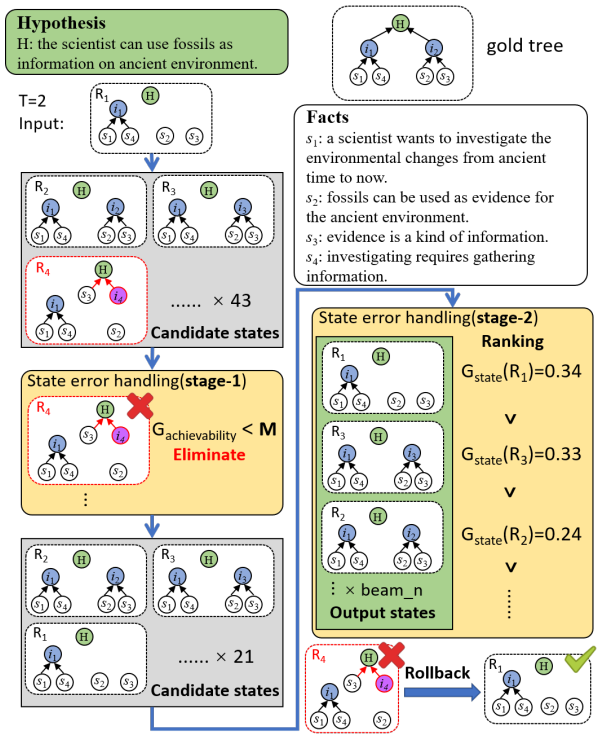

Figure 6: Case of state error handling. $G_{achievability}$ represents the scores of state achievability, where M is its median value. G_{state} represents the scores of state evaluation. Red indicates incorrect states.

state has already executed step s_1&s_4. After the step inference, it is possible to execute the correct step s_2&s_3 or the erroneous abd_step H&s_3 (note that this is an abductive reasoning operation that infers an intermediate conclusion using a hypothesis and a premise). Without the mechanism for handling state errors, the incorrect state would be outputted in the candidate state. However, our method is capable of detecting errors in the first

stage and, in the second stage, rolling back to the state where only step s_1&s_4 is executed. As a result, we can obtain the correct entailment tree.

7. Conclusion

We propose SPEH for step feasibility-aware and error-correctable entailment tree generation. The step feasibility perception learns to select steps that are not false feasible, while the state error handling allows the model to detect errors and rollback to a reasonable state, mitigating the adverse effects of error search branches. Experimental results demonstrate that our approach surpasses existing baseline models.

8. Acknowledgements

This work was supported by the National Natural Science Foundation of China (62076100), Fundamental Research Funds for the Central Universities, SCUT (x2rjD2230080), the Science and Technology Planning Project of Guangdong Province (2020B0101100002), Guangdong Provincial Fund for Basic and Applied Basic Research - Regional Joint Fund Project (Key Project) (2320191025000318,308155351064), CAAI-Huawei MindSpore Open Fund, CCF-Zhipu AI Large Model Fund.

Bibliographical References

Parishad BehnamGhader, Santiago Miret, and Siva Reddy. 2022. Can retriever-augmented language models reason? the blame game between the retriever and the language model. *CoRR*, abs/2212.09146.

Alex Bogatu, Zili Zhou, Dónal Landers, and André Freitas. 2022. Active entailment encoding for explanation tree construction using parsimonious generation of hard negatives. *CoRR*, abs/2208.01376.

Kaj Bostrom, Zayne Sprague, Swarat Chaudhuri, and Greg Durrett. 2022. Natural language deduction through search over statement compositions. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 4871–4883. Association for Computational Linguistics.

Kaj Bostrom, Xinyu Zhao, Swarat Chaudhuri, and Greg Durrett. 2021. Flexible generation of natural language deductions. 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 6266–6278. Association for Computational Linguistics.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3882–3890. ijcai.org.

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

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4443–4458. Association for Computational Linguistics.

Matthew Ho, Aditya Sharma, Justin Chang, Michael Saxon, Sharon Levy, Yujie Lu, and William Yang Wang. 2023. Wikiwhy: Answering and explaining cause-and-effect questions. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

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.

Ruixin Hong, Hongming Zhang, Hong Zhao, Dong Yu, and Changshui Zhang. 2023. Faithful question answering with monte-carlo planning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 3944–3965. 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. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 4698–4724. Association for Computational Linguistics.

Harsh Jhamtani and Peter Clark. 2020. Learning to explain: Datasets and models for identifying valid reasoning chains in multihop question-answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 137–150. Association for Computational Linguistics.

Sunnie S. Y. Kim, Elizabeth Anne Watkins, Olga Russakovsky, Ruth Fong, and Andrés Monroy-Hernández. 2023. "help me help the ai": Understanding how explainability can support human-ai interaction. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI 2023, Hamburg, Germany, April 23-28, 2023*, pages 250:1–250:17. ACM.

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.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zhengzhong Liang, Steven Bethard, and Mihai Surdeanu. 2021. Explainable multi-hop verbal reasoning through internal monologue. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 1225–1250. 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. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 7177–7189. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Graham Loomes and Robert Sugden. 1982. Regret Theory: An Alternative Theory of Rational Choice Under Uncertainty. *The Economic Journal*, 92(368):805–824.

Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.*, 267:1–38.

Guglielmo Papagni, Jesse de Pagter, Setareh Zafari, Michael Filzmoser, and Sabine T. Köszegi. 2023. Artificial agents' explainability to support trust: considerations on timing and context. *AI Soc.*, 38(2):947–960.

Hanhao Qu, Yu Cao, Jun Gao, Liang Ding, and Ruifeng Xu. 2022. Interpretable proof generation via iterative backward reasoning. 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 2968–2981. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training.

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.

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.

Joshua David Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. 2021. Contrastive learning with hard negative samples. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. Prover: Proof generation for interpretable reasoning over rules. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 122–136. Association for Computational Linguistics.

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.

Zayne Sprague, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. 2022. Natural language deduction with incomplete information. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 8230–8258. Association for Computational Linguistics.

Zayne Sprague, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. 2023. Deductive additivity for planning of natural language proofs. *CoRR*, abs/2307.02472.

Saku Sugawara, Pontus Stenetorp, Kentaro Inui, and Akiko Aizawa. 2020. Assessing the benchmarking capacity of machine reading comprehension datasets. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8918–8927. AAAI Press.

Changzhi Sun, Xinbo Zhang, Jiangjie Chen, Chun Gan, Yuanbin Wu, Jiaze Chen, Hao Zhou, and Lei Li. 2021. Probabilistic graph reasoning for natural proof generation. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 3140–3151. Association for Computational Linguistics.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 3621–3634. Association for Computational Linguistics.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. 2022. Entailer: Answering questions with faithful and truthful chains of reasoning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 2078–2093. Association for Computational Linguistics.

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

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. 2021. Diagnosing the first-order logical reasoning ability through logicnli. 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 3738–3747. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

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 *NeurIPS*.

Nathaniel Weir and Benjamin Van Durme. 2022. Dynamic generation of interpretable inference rules in a neuro-symbolic expert system. *arXiv preprint arXiv: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. 2022a. Generating natural language proofs with verifier-guided search. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 89–105. Association for Computational Linguistics.

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

Zonglin Yang, Li Dong, Xinya Du, Hao Cheng, Erik Cambria, Xiaodong Liu, Jianfeng Gao, and Furu Wei. 2022b. Language models as inductive reasoners. *CoRR*, abs/2212.10923.

Qinyuan Ye, Xiao Huang, Elizabeth Boschee, and Xiang Ren. 2020. Teaching machine comprehension with compositional explanations. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 1599–1615. Association for Computational Linguistics.

Nathan Young, Qiming Bao, Joshua Bensemann, and Michael Witbrock. 2022. Abductionrules: Training transformers to explain unexpected inputs. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 218–227. Association for Computational Linguistics.

Hongyu Zhao, Kangrui Wang, Mo Yu, and Hongyuan Mei. 2023. Explicit planning helps language models in logical reasoning. *CoRR*, abs/2303.15714.

A. Training Details

In the methodology section, we briefly outlined the general steps for constructing additional training data. Given that the primary training process relies on margin ranking loss $\phi(x_1, x_2, m) = max(0, x_2 - x_1 + m)$, we made some modifications to the positive and negative samples provided by Hong et al. (2022). The construction process relies on flexible samples, which refer to samples that, although considered gold samples from a human cognitive perspective, are made flexible enough to be positioned either as positive or negative samples.

Pair Type	Pair Sample	Probabilities
(R_1, R_2)	$(R^+_{t=T}, R^-_{t=T})$	50%
(R_3, R_4)	$(R^+_{t=T}, R^+_{t=T-1})$	25%
(R_3, R_4)	$(R^+_{t=T}, R^-_{t=T-1})$	25%
(R_3, R_4)	$(R^+_{t=T}, R^-_{t=T+1})$	25%

Table 4: Probabilities of different training data. R^+ denotes gold state and R^- denotes non-gold state.

For step feasibility perception samples, we randomly pair them from positive step sets and categorize the shallower one as a positive sample p_3 and the deeper one as a negative sample p_4, hence p_4 being termed as flexible samples, creating a new pair of step training data. To control the volume of training data, (p_1, p_2) and (p_3, p_4) are each added to the final training data with a 50% probability.

Regarding state error handling samples, each positive state is treated as R_3, matched with its corresponding positive state from the previous time step as R_4, or matched with negative states from preceding or succeeding time steps as R_4. Similarly, to manage the volume of training data, each pair of (R_1, R_2) or (R_3, R_4) is finally sampled with a certain probability as the ultimate training data, with these probabilities as shown in Table 4.

B. Entailment Module

We constructed new data following the format: *hypothesis:* $\{H\}$ *premise:* $\{p_1\}$ *premise:* $\{p_2\}$, and continued fine-tuning on the T5 checkpoint provided by Hong et al. (2022) for 100 epochs, using a batch size of 8 and a learning rate of 3e-5. Additionally, during the inference process, we modified the inputs of the selected deductive step using the same format to match the entailment module after re-fine-tuning.

C. Reasoning Algorithm

To provide a more detailed explanation of the reasoning process for error handling, we present pseudocode of the inference process in Algorithm 1. It is important to note that in each iteration, $S_{hold,T}$, and $S_{inference,T}$ respectively contain the retained

Algorithm 1 State Error Handling

Input: Hypothesis H, initial state R_0, bean size K, Max reasoning time T_{max}
Output: Completed state set $S_{completed}$

$T \leftarrow 0$
$S_{in,T} \leftarrow \{\}$
$S_{completed} \leftarrow \{\}$
while $not\ reasoning\_complete$ **do**
  if $T > 0$ **then**
    $S_{in,T} \leftarrow S_{out,T-1}$
  end if
  $S_{inference,T} \leftarrow \{\}$
  $S_{hold,T} \leftarrow \{\}$
  for $R_t \in S_{in,T}$ **do**
    if $t == T - 1$ **then**
      $S_{hold,T} \leftarrow S_{hold,T} \cup \{R_t\}$
    end if
    $S_{inference,T} \leftarrow S_{inference,T} \cup \{inference(R_t)\}$
  end for
  $S_{candidate,T} \leftarrow S_{inference,T} \cup S_{hold,T}$
  $S_{ach,T} \leftarrow Filter_{ach}(S_{candidate,T})$ get top 50% states
  $S_{out,T} \leftarrow Filter_{state}(S_{ach,T})$ get top K states
  for $R_t in S_{out,T}$ **do**
    if $t == T_{max}$ **then**
      $S_{completed} \leftarrow S_{completed} \cup R_t$
    end if
    if size of $S_{completed} == K$ **then**
      $reasoning\_complete \leftarrow True$
      $break$
    end if
  end for
  $T \leftarrow T + 1$
end while
return $S_{completed}$

states from the previous round and the states after the current inference. During the selection process, filtering is based on the same criteria for all states.

D. ChatGPT-3.5 Prompt

We employed an in-context learning approach in our experiments to directly generate each entailment tree using ChatGPT-3.5. The process involved the following prompt:

You are required to generate a proof from the given premises and hypothesis, here is the example:
Given premises:sent1: united states is located in the northern hemisphere sent2: december is during the winter in the northern hemisphere sent3: new york / new york state is a state located in the united states of america sent4: winter has the least sunlight
Given hypothesis:new york state has the least sunlight during december
Output proof:sent1 & sent3 -> int1: new york state is located in the northern hemisphere; int1 & sent2 -> int2: december is during the winter for new york state; int2 & sent4 -> hypothesis;
Please generate proof similar to the example for the following sample:
Premise:sent1: $\{s_1\}$ *sent2:* $\{s_2\}$ *sent3:* $\{s_3\}$ *...*
Hypothesis:$\{H\}$
Only generate output proof in one line, each step ends with '; '

After obtaining output proof conforming to the specified format, we automatically convert them into entailment trees for evaluation. Outputs that cannot be parsed are directly considered as failed samples.