A Logical Pattern Memory Pre-trained Model for Entailment Tree Generation

Li Yuan[1,2], **Yi Cai**[1,2], **Haopeng Ren**[1,2], **Jiexin Wang**[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
seyuanli@mail.scut.edu.cn, ycai@scut.edu.cn, jiexinwang@scut.edu.cn

Abstract

Generating coherent and credible explanations remains a significant challenge in the field of AI. In recent years, researchers have delved into the utilization of entailment trees to depict explanations, which exhibits a reasoning process of how a hypothesis is deduced from the supporting facts. However, existing models often overlook the importance of generating intermediate conclusions with logical consistency from the given facts, leading to inaccurate conclusions and undermining the overall credibility of entailment trees. To address this limitation, we propose the logical pattern memory pre-trained model (LMPM). LMPM incorporates an external memory structure to learn and store the latent representations of logical patterns, which aids in generating logically consistent conclusions. Furthermore, to mitigate the influence of logically irrelevant domain knowledge in the Wikipedia-based data, we introduce an entity abstraction approach to construct the dataset for pre-training LMPM. The experimental results highlight the effectiveness of our approach in improving the quality of entailment tree generation. By leveraging logical entailment patterns, our model produces more coherent and reasonable conclusions that closely align with the underlying premises. Code and Data are released at https://github.com/YuanLi95/T5-LMPM

Keywords: Entailment Tree, Memory Network, Logical Pattern

1. Introduction

Generating coherent and credible explanations poses a significant challenge in AI, and addressing it represents a giant leap towards the goal of building reliable reasoning systems (Barredo Arrieta and Díaz-Rodríguez, 2020). The reasoning chain has traditionally served as the primary representation for constructing reasonable explanations (DeYoung et al., 2020; Tafjord et al., 2021). However, in recent years, researchers have explored the utilization of entailment trees for explanation generation (Dalvi et al., 2021; Yang et al., 2022; Hong et al., 2022). As shown in Figure 1, the task of entailment tree generation can be defined as follows: given a hypothesis (summarizing from a question+answer pair) and a set of supporting facts, the goal is to derive an entailment tree where each non-leaf node is an intermediate conclusion generated from its children. By providing valid entailment trees, users can develop a better understanding of the reasoning process and acquire reliable information for effective decision.

The early work by Dalvi et al. (2021) proposes a single-step method for generating entailment trees. However, subsequent research conducted by Hong et al. (2022) has highlighted that such an approach often yields unreliable explanations for the hypothesis. To tackle this limitation, several recent studies employing multi-step generation methods have been introduced (Yang et al., 2022; Neves Ribeiro et al., 2022; Hong et al., 2022; Liu et al., 2022). These methods have demonstrated impressive performance by iteratively selecting premises and generating intermediate conclusions to construct the entire tree. Nonetheless, a significant drawback of many existing iterative methods lies in their limited attention to the logical patterns between premises and conclusions. As a result, they might generate

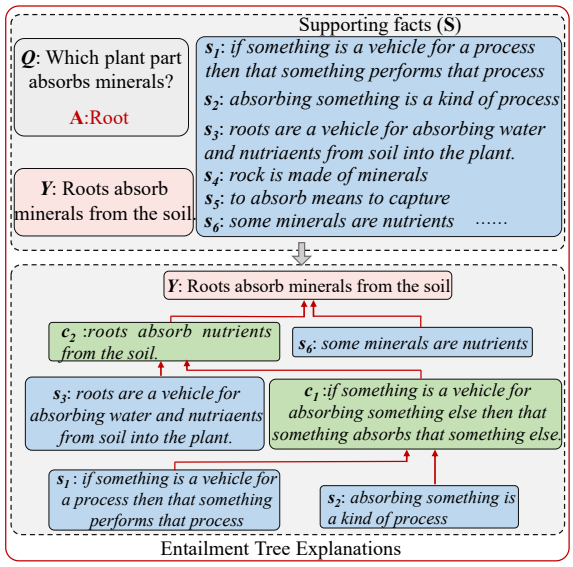


Figure 1: Illustration of Entailment Tree Generation. The top half presents the inputs, while the generated entailment tree is depicted in the bottom half. The tree consists of a hypothesis (pink), premises (blue), and generated intermediate conclusions (green).

*Corresponding authors

LREC-COLING 2024, pages 759–772
20-25 May, 2024.

intermediate conclusions that lack logical consistency and conflict with the premises.

Among the iterative methods, METGEN (Hong et al., 2022) stands out for its attempt to integrate logical patterns into the tree generation process. However, it primarily focuses on the module responsible for premise selection, with relatively limited attention to effectively capturing and learning logical patterns. METGEN's strategy for integrating logical patterns involves training the language model on a synthetic dataset constructed from Wikipedia. This dataset exhibits logical regularities, as illustrated by the green boxes in Figure 2. We argue that this approach is insufficient for comprehensively learning the intricacies of logical features. On one side, during training, the language model is forced to learn both logical patterns and textual features, which dilutes its focus on capturing logical patterns. Furthermore, using Wikipedia data as a training resource introduces a considerable amount of domain-specific information that isn't logically relevant. For instance, in Figure 2, when METGEN leverages data containing logical regularities (green boxes) to learn corresponding logical patterns (gray box) that capture generalizable logical relationships between entities, it inadvertently incorporates specific entity terms and domain-specific knowledge (e.g., *primitive society* and *social development*). This inclusion of specific information in the logical pattern training data poses challenges for METGEN in effectively acquiring logical patterns, which hinders its ability to generalize the reasoning process and identify similar relationships across various instances.

In this paper, we propose the **L**ogical Pattern **M**emory **P**re-trained **M**odel (LMPM) to address the aforementioned limitations, as depicted in Figure 3. Our primary objective is to effectively exploit logical patterns to generate logically consistent conclusions. To address the limitation regarding the inadequate focus on capturing logical patterns, we introduce an external memory component that operates independently of the language model. This memory module is designed to learn and store latent logical patterns between premises and conclusions, enabling the model to better capture and utilize logical relationships. Furthermore, to mitigate the influence of logically irrelevant domain knowledge in the Wikipedia-based data, we adopt an entity abstraction approach, as illustrated in Figure 2. This involves abstracting entities from the Wikipedia text, resulting in an entity-abstract dataset utilized for pre-training our LMPM. By leveraging the abstraction data, the model can uncover fundamental logical relationships, thereby enhancing its reasoning capabilities while reducing the impact of irrelevant knowledge and necessitating less training data. In the experiments, we integrate

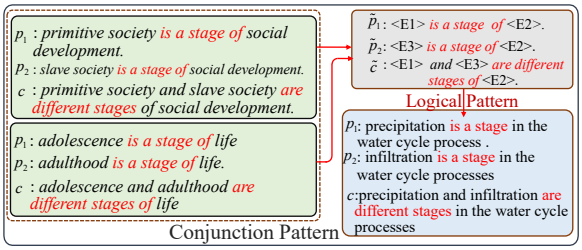


Figure 2: Examples of Logical Patterns. Each box represents a single-step deductive process. The texts within the green and blue boxes are extracted from the Wikipedia-based synthetic dataset and the entailment tree corpus, respectively. The text within the gray box is the logical patterns obtained through entity abstraction, where <E> denotes the special token <extra_id_*> reserved in T5.

LMPM with METGEN, simply using it to generate the intermediate conclusions of logical consistency. To sum up, we make the following contributions in this work:

- We introduce a logical pattern memory pre-trained model (LMPM), which facilitates the generation of logically consistent intermediate conclusions during entailment steps. LMPM employs an external memory to learn and retain latent logical patterns between premises and conclusions. This mechanism significantly enhances the language model's capacity to capture and utilize logical patterns effectively.
- We propose to pre-train the LMPM model via a constructed entity-abstract dataset, explicitly guiding the model to learn representations of latent logical patterns. This approach mitigates the influence of irrelevant domain knowledge in the original Wikipedia data and enables the model to be well-trained with less data.
- We assess our proposed method using both automatic and human evaluations on entailment tree generation datasets. The results demonstrate that LMPM outperforms strong baseline models in most cases and provides effective representations of logical patterns.

2. Related Work

2.1. Entailment Tree Generation

Question answering (QA) has garnered significant research interest, with recent works dedicated to explaining QA as a new departure for responsible AI systems (Ye et al., 2020; Peng et al., 2020; Zhang et al., 2021; Ren et al., 2023; Mavi et al., 2022; Yuan et al., 2020, 2023). Nonetheless, most of these efforts have primarily focused on identifying relevant facts for questions and answers (Jansen and Ustalov, 2019; Yadav et al., 2019; Yuan et al., 2020, 2022), rather than constructing valid reason-

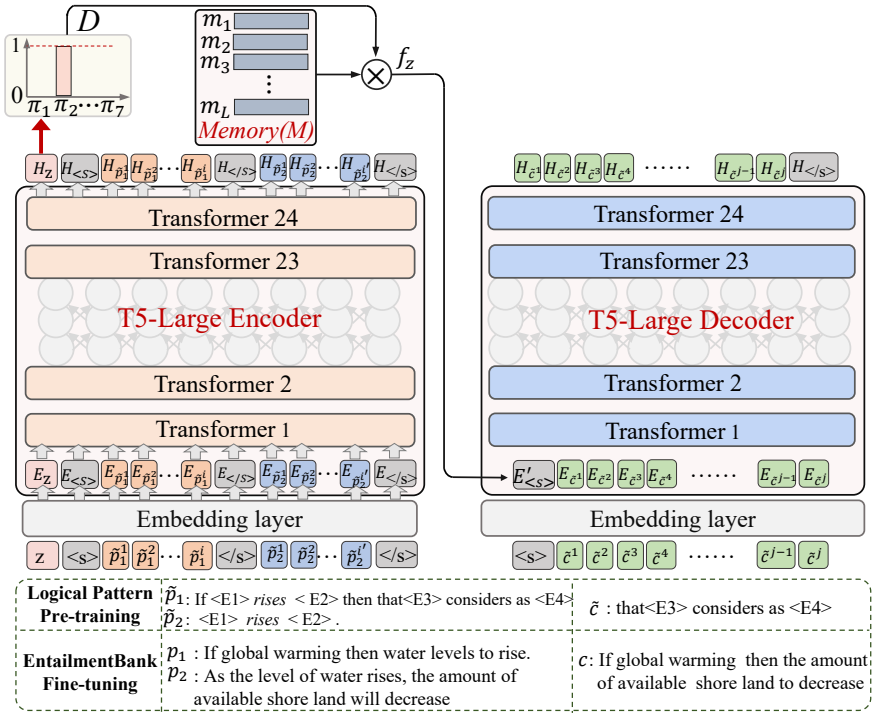


Figure 3: The overall architecture of LMPM

ing paths. While some studies have explored the generation of explanation sentences leading to answers (Camburu et al., 2018; Rajani et al., 2019; Wu et al., 2023), these explanations are often unreliable as they are not guided by explicit facts (Bostrom et al., 2021). To overcome these limitations, Dalvi et al. (2021) introduce EntailmentBank, a dataset specifically designed for the task of entailment tree generation. Each multi-step entailment tree in EntailmentBank serves as an explanation, clearly demonstrating the reasoning process behind a hypothesis based on a set of supporting facts, as shown in Figure 1.

In addition to introducing the EntailmentBank, Dalvi et al. (2021) propose the EntailmentWriter model to tackle the task. This model utilizes an end-to-end language model to generate complete entailment trees in a single step. However, a drawback of this approach is that it does not explicitly constrain the validity of every single step, leading to potentially unreliable explanations for the given hypothesis. To address the limitations of single-step methods, several recent works (Neves Ribeiro et al., 2022; Hong et al., 2022; Liu et al., 2022) have presented multi-step generation approaches, which iteratively select premise facts and generate intermediate conclusions. For instance, Neves Ribeiro et al. (2022) have devised an iterative retrieval-generation framework (IRGR) that leads to enhanced retrieval results in the context of Task 3. Liu et al. (2022) have introduced a Reinforcement Learning-based Entailment Tree generation framework (RLET), which is trained by utilizing cumulative signals across the entire tree. However, such methods suffer from a coupling issue as the premise selection and conclusion generation are performed simultaneously by the T5 model. On the other hand, Hong et al. (2022) introduced the METGEN model, which is a controller-based model consisting of proprietary modules for selection and generation. METGEN primarily focuses on the controller module, which is responsible for selecting the premises, followed by an independent T5 to generate the conclusion. Unlike other works that completely overlook logical relations, METGEN slightly improves its ability of logical summarizing by training the model with a Wikipedia-based synthetic dataset.

In contrast to the aforementioned approaches, we focus on improving the model's ability to generate logically consistent conclusions in the entailment steps by leveraging logical patterns. We underscore the significance of integrating logical patterns into the reasoning process. To achieve this, we employ an external memory to both learn and store the features of logical patterns, facilitating the generation of logically consistent conclusions. Moreover, our model can be adapted to enhance existing methods like METGE, leveraging the advancements of LMPM in effectively incorporating logical patterns into the entailment tree generation.

2.2. Memory Networks

Weston et al. (2015) first propose the memory network, which employs memory components to store scene information, thereby enabling the network to maintain long-term memory. Since then, memory networks have found widespread application in var-

ious natural language processing tasks. They have demonstrated effectiveness in storing multi-hop or long-term information across contexts, notably in applications such as dialogue systems (Wang et al., 2020; Chen et al., 2023), sentiment analysis systems (Xu et al., 2019), and QA systems (Huang et al., 2021; Li et al., 2022). In contrast to these existing works, LMPM is specifically designed to memorize and store the features of logical patterns, which can be reused and guide the model in generating intermediate conclusions of logical consistency.

3. Methodology

Formally, the entailment tree generation involves a hypothesis Y and a set of factual sentences $S = \{s_1, s_2, \cdots, s_m\}$ as inputs, where m denotes the number of candidate facts. The hypothesis Y can be deduced from a subset of S and the goal is to generate a valid tree T that portrays the deduction process, as depicted in Figure 1. The leaves of the tree correspond to the facts selected from S, while the root refers to Y. We use c_i to denote the intermediate node, representing the intermediate conclusion generated by the model. Moreover, each non-root node, including both the leaves and intermediate nodes, also corresponds to a premise of its parent node. Based on the scope of the candidate facts S, the task can be further categorized into the following sub-tasks of increasing difficulty:
Task 1 (no-distractor) $S = S_{gold}$,
Task 2 (distractor) $S = S_{gold}$ + 15-20 distractors,
Task 3 (full-corpus) $S =$ full corpus.
where S_{gold} is the set of golden leaf facts.

3.1. LMPM

In this section, we present the proposed **L**ogical Pattern **M**emory **P**re-trained **M**odel (LMPM), which consists of an encoder-decoder generative model (T5[1]) and an external memory denoted as *M*. The overall architecture is illustrated in Figure 3. The key insight of LMPM is to map the implicit logical patterns onto the embedding space. The training data of logical patterns, acquired through the entity abstraction techniques, force the model to concentrate specifically on essential logical connections, while simultaneously disregarding specific terms or domain-specific knowledge. For instance, consider the logical pattern depicted in the gray box of Figure 2, which explores the concept of stages related to entity <E2>. The premises assert that both <E1> and <E3> represent stages of <E2>. From these premises, we can deduce the conclusion that <E1> and <E3> are distinct stages of <E2>. Thus, by leveraging such logical patterns, we can generalize the reasoning process and identify similar relationships across various instances.

The memory *M* is responsible for learning and storing representations of logical patterns between premises and conclusions. The T5 model takes the two premises as inputs, selects the logical pattern representations from *M*, and leverages these representations to generate logically consistent conclusions during the deductive process. LMPM is trained through two essential tasks: *Logical Pattern Pre-training* and *EntailmentBank Fine-tuning*. The first task forces the model to capture and learn representations of latent logical patterns via a constructed entity-abstract dataset. The second task focuses on training the model to generate intermediate conclusions of logical consistency by adapting the pre-trained representations of logical patterns to the target EntailmentBank dataset.

3.1.1. Logical Pattern Pre-training

As previously mentioned, the first task of LMPM aims to enhance its ability to capture and learn latent logical patterns. To achieve this, we construct a dataset using entity abstraction techniques. We utilize the Wikipedia-based synthetic dataset introduced by Hong et al. (2022), which encompasses logical regularities but also contains a substantial amount of domain-specific information. Each data instance comprises two premises and a conclusion, simulating the process of generating intermediate conclusions. The logical deductive types are categorized as *substitution*, *conjunction*, and *if-then*. Examples of logical patterns belonging to the conjunction type are illustrated in the green boxes of Figure 2. To abstract the original synthesis data, 120 special tokens (e.g., "<extra_id_1>" that used in T5) are leveraged to replace the entity spans. Specifically, we replace entity spans whose part-of-speech corresponds to *noun*, *det noun*, *adj noun*, or *def adj noun*. For instance, in Figure 2, the phrase "primitive society" within the top-left box is replaced with "<E1>", and the gray box displays the result after the abstraction steps. The resulting dataset comprises a collection of logical patterns, where each data instance, denoted as $\{(\tilde{p}_1, \tilde{p}_2), \tilde{c}\}$, consists of two pseudo premises and the corresponding pseudo intermediate conclusion. This entity-abstract dataset is constructed to learn a latent variable f_z for each logical pattern, which is then utilized for generating logically consistent conclusions, and the equation is:

$$p(\tilde{c}, f_z|\tilde{p}_1, \tilde{p}_2) = p(f_z|\tilde{p}_1, \tilde{p}_2)p(\tilde{c}|f_z, \tilde{p}_1, \tilde{p}_2) \quad (1)$$

Based on the T5 encoder, we adopt a specific

[1]We choose T5 as the backbone in LMPM, similar to the works previously introduced (Dalvi et al., 2021; Yang et al., 2022; Hong et al., 2022), to facilitate ease of comparison in experiments. Note that any encoder-decoder model can serve as the backbone, not exclusively T5.

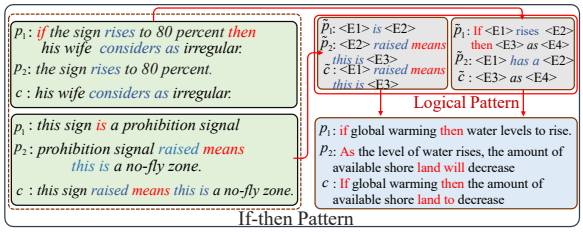


Figure 4: Examples of complex deductions. The gray boxes represent two distinct logical patterns that need to be combined to generate the appropriate conclusion shown in the blue box.

sequence format where we insert the token <s> before the first premise, and use the token </s> to either concatenate the two premises or indicate the sequence's end. Additionally, we introduce a special latent token <z> at the beginning of the sequence. Thus, the input sequence I_{logical} for logical pattern pre-training is transformed as follows:

$$I_{\text{logical}} = [<z>,<s>,\tilde{p}_1^1,..,\tilde{p}_1^l,</s>,\tilde{p}_2^1,...,\tilde{p}_2^{l'},</s>] \quad (2)$$

where l and l' are the lengths of $\tilde{p}_1$ and $\tilde{p}_2$, respectively. Subsequently, we use the T5-encoder to encode the input sequence I_{logical}. The final hidden state of the token <z>, denoted as H_z, serves as the logical representation of the two premises. H_z is utilized to select the specific logical pattern from M. The external memory M, parameterized by θ, is responsible for learning and storing the features of logical patterns. Each element m_i in M indicates a specific logical pattern:

$$M = [m_1, ..., m_L] \in \mathbb{R}^{L \times d_m} \quad (3)$$

where L corresponds to the number of logical patterns stored in the memory, and d_m is the dimension of the latent representations.

To leverage the logical pattern representations stored in the memory M for generating intermediate conclusions, LMPM must identify the specific logical pattern corresponding to the premises. This requires determining the value of i that satisfies $f_z = m_i$, where f_z denotes the representation of the target logical pattern, and m_i refers to an element within the memory, as previously mentioned. To facilitate this identification process and learn the representations, we introduce an address structure D, which employs a multi-layer perceptron architecture. This structure takes H_z as input and produces a one-hot vector $\alpha_i \in \mathbb{R}^{1 \times L}$ to help locate the potential logical pattern representation in M. In particular, we use Gumbel-softmax (Jang et al., 2016) for optimization:

$$\gamma_i = w_z H_z + b_z \quad (4)$$

$$\alpha_i = \frac{\exp(\gamma_i + g_i)/\mathcal{T}}{\sum_{i=1}^{L} \exp(\gamma_i + g_i)/\mathcal{T}} \quad (5)$$

Here, g_i is a random variable following the Gumbel distribution and $\mathcal{T}$ is the temperature coefficient. After obtaining the one-hot vector α_i, we can locate the potential logical pattern in M using:

$$f_z = \sum_{i=1}^{L} \alpha_i M_i \quad (6)$$

Next, we integrate the selected logical pattern representation into the T5-decoder by adding it to the representation of the start special token <s>. The purpose is to effectively introduce the logical pattern, as shown in the equation:

$$E'_{<s>} = f_z + E_{<s>} \quad (7)$$

The memory structure M and the address structure D can be updated through backpropagation, utilizing the entity-abstract dataset. Additionally, Section 4.3 demonstrates that this abstraction approach offers substantial benefits to the model, notably reducing the impact of irrelevant knowledge and facilitating effective training with less data.

Loss Function To optimize the model parameters in the logical pattern pre-training task, two loss functions are employed. The first one is the language modeling loss (LM), which is defined as:

$$\mathcal{L}_{LM} = -\sum_{t=1}^{|\tilde{c}|} \log p(\tilde{c}^t | f_z, \tilde{p}_1, \tilde{p}_2, \tilde{c}^{0:t-1}) \quad (8)$$

where $|\tilde{c}|$ is the length of $\tilde{c}$. In addition, to tackle the issue of vanishing latent variables (Zhao et al., 2017), a bag-of-word loss (BOW) is applied:

$$L_{BOW} = \sum_{t=1}^{|\tilde{c}|} \log p(\tilde{c}^t | f_z, \tilde{p}_1, \tilde{p}_2) = \sum_{t=1}^{|\tilde{c}|} \log \frac{e^{f(\tilde{c}^t)}}{\sum_{v \in V} e^{f(v)}} \quad (9)$$

Therefore, the total loss in the logical pattern pre-training is defined as follows:

$$\mathcal{L} = \mathcal{L}_{LM} + \mathcal{L}_{BOW} \quad (10)$$

3.1.2. EntailmentBank Fine-tuning

The second task aims to fine-tune LMPM for generating actual intermediate conclusions with logical consistency in the entailment steps, adapting the pre-trained logical pattern representations to the target EntailmentBank dataset. While the overall process resembles the previous logical pattern pre-training task, it uses the actual premises and intermediate conclusions from the EntailmentBank dataset for training. Moreover, the dataset includes some complex deductions that can not be adequately captured by a single logical pattern stored in M. To address this limitation, the model must incorporate multiple logical pattern representations to generate appropriate intermediate conclusions, as

	Train	Dev	Test	All
Entailment trees	1,131	187	340	1,840
Entailment steps	4,175	597	1,109	5,881

Table 1: EntailmentBank statistics.

illustrated in the example depicted in Figure 4. To enable the combination of multiple logical pattern representations, we replace the Gumbel-softmax in equation (5) with Softmax:

$$\widetilde{\alpha}_i = \frac{\exp(\gamma_i)}{\sum_{i=1}^{L} \exp(\gamma_i)} \tag{11}$$

In this case, $\widetilde{\alpha}_i$ is not a one-hot vector. The model is trained to combine multiple logical pattern representations in M to generate the conclusions.

4. Experiments

4.1. Datasets & Implementation Details

We perform experiments on EntailmentBank (Dalvi et al., 2021), which is created based on ARC dataset of grade-school science questions (Clark et al., 2018), and a corpus of science and general knowledge from WorldTree V2 (Xie et al., 2020). EntailmentBank contains 1840 entailment trees, where each tree corresponds to a question. The statistics are shown in Table 1.

The memory structure stores $L = 7$ logical patterns, corresponding to the number of inference types in the EntailmentBank dataset. Each memory element has a dimension of d_m set to 4096. During the logical pattern pre-training phase, the model is trained for two epochs using a batch size of 35 and a learning rate of 3e-5. For the subsequent EntailmentBank fine-tuning phase, the model is trained for 80 epochs with a batch size of 30 and a learning rate of 3e-5. To create the logical pattern pre-training dataset, we employ spaCy[2] with the *en_core_web_sm* version to parse sentences and extract the part-of-speech information for each token. Subsequently, we use GloVe (Jeffrey Pennington, Richard Socher, 2014) to identify tokens of different forms in the premises and conclusions, replacing them with the same special token. Regarding the quantity and length of logical patterns, our constructed dataset comprises a total of 565,453 patterns, with an average length of 17 and a maximum length of 43. Notably, in the experiments, we integrate LMPM with METGEN by replacing its T5 component, allowing it to utilize the intermediate conclusions of logical consistency.

4.2. Evaluation Metrics

Automatic Evaluation We assess the generated tree in comparison to the golden tree across three dimensions, following established practices in previous studies (Dalvi et al., 2021; Hong et al., 2022; Yang et al., 2022): (1) **Leaves:** measures whether the correct leaf facts are used. We compute the F_1 score by comparing the predicted leaf facts to those in the golden tree. Additionally, we report the *AllCorrect* score, which indicates exact matches[3]. (2) **Steps:** evaluates whether the entailment steps are structurally correct. We compare the steps in two trees using the F_1 score and *AllCorrect*. A predicted step is considered correct if its children's identifiers perfectly match the corresponding ones in the gold tree. (3) **Intermediates:** assesses the accuracy of intermediate conclusions, also utilizing the F_1 score and the *AllCorrect* score. A predicted intermediate conclusion is considered correct if the `BLEURT-Large-512` (Sellam et al., 2020) score between the aligned predicted intermediate and the corresponding golden intermediate is greater than 0.28.

Human Evaluation Considering the potential limitations of automated evaluation, as highlighted by Dalvi et al. (2021), which might inaccurately assess certain valid trees and underestimate their performance, we conducted a human evaluation. We randomly sampled 100 instances from the EntailmentBank test set and asked 9 graduate students to evaluate the model results based on four criteria: (1) **Validity**: evaluates whether the generated tree clearly demonstrates a coherent chain of reasoning, explaining how the hypothesis is derived from the supporting facts. (2) **Logical consistency**: assesses whether each intermediate conclusion aligns with the given premises. (3) **Readability**: assesses whether the generated intermediate conclusions are grammatically correct and fluent to read. (4) **Reasonability**: evaluates whether the generated intermediate conclusions adhere to common knowledge and facts. For the **Validity** metric, we compute the number of trees considered valid by the majority of evaluators. For the remaining three criteria, the students were asked to rate the generated intermediate conclusions from 1 (very bad) to 5 (very good), and the average scores were reported. Note that the human evaluation was only conducted for task 1, as the other two tasks would have been more challenging and time-consuming for human evaluators.

[2]https://spacy.io/models.

[3]The AllCorrect is 1 if the F_1 is 1, and 0 otherwise.

Method	Additional Data	Leaves		Steps		Intermediates		Overall
		F_1	AllCorrect	F_1	AllCorrect	F_1	AllCorrect	AllCorrect
Task1								
EntailmentWriter(T5-large)	No	98.4	84.1	50.0	38.5	67.0	35.9	34.4
IRGR	No	97.6	89.4	50.2	36.8	62.1	31.8	32.4
RLET	No	**100.0**	**100.0**	55.0	41.2	67.2	36.7	35.1
METGEN+T5	564K	**100.0**	**100.0**	57.7	41.9	70.8	39.2	36.5
METGEN+LMPM(*ours*)	564K	99.76	99.41	**57.78**	**43.82**†	**72.78**†	**42.78**†	**38.54**†
Task2								
EntailmentWriter(T5-large)	No	**83.2**	35.0	39.5	24.7	**62.2**	28.2	23.2
IRGR	No	69.9	23.8	30.5	22.40	47.70	26.5	21.8
RLET	No	81.9	40.4	38.8	28.7	57.4	29.1	26.0
METGEN+T5	564K	82.7	46.1	41.3	29.6	61.4	32.4	27.7
METGEN+LMPM(*ours*)	564K	81.09	**47.06**	**42.56**	**31.38** †	61.68	**34.32**†	**29.41**†
Task3								
EntailmentWriter(T5-large)	No	30.9	1.2	4.4	1.2	28.8	5.6	1.2
IRGR	No	**46.6**	10.0	11.3	8.2	38.7	**20.9**	8.2
RLET	No	46.20	**11.41**	**15.2**	**9.6**	**41.4**	17.6	9.4
METGEN+T5	564K	34.8	8.7	9.8	8.6	36.6	20.4	8.6
METGEN+LMPM(*ours*)	564K	35.3	9.23	10.28	9.23	37.8	20.33	**9.41**

Table 2: Automatic evaluation results on the EntailmentBank dataset (%). The best and second-best results are highlighted in bold and underlined, respectively. † refers to significance test $p - value < 0.05$. The "Additional Data" column denotes the size of the supplementary Wikipedia training data.

Mtrics	METGEN+T5	METGEN+LMPM
Validity	46	52
Logical	3.02	3.37
Readability	4.11	4.43
Reasonability	3.37	3.64

Table 3: Human evaluation results on 100 uniformly sampled questions from the test split.

Method	Task 1		Task 2	
	Inter	Overall	Inter	Overall
METGEN+T5	39.2	36.5	32.4	27.7
METGEN+LMPM	42.78	38.54	34.32	29.41
w/o *LPP*	39.02	36.47	32.24	27.56
w/o *memory*	40.59	37.39	33.06	28.54
w/o *abstraction*	41.27	37.45	33.78	28.48

Table 4: Ablation study results. "Inter" and "Overall" denote Intermediates AllCorrect and Overall AllCorrect, respectively.

4.3. Experimental Results

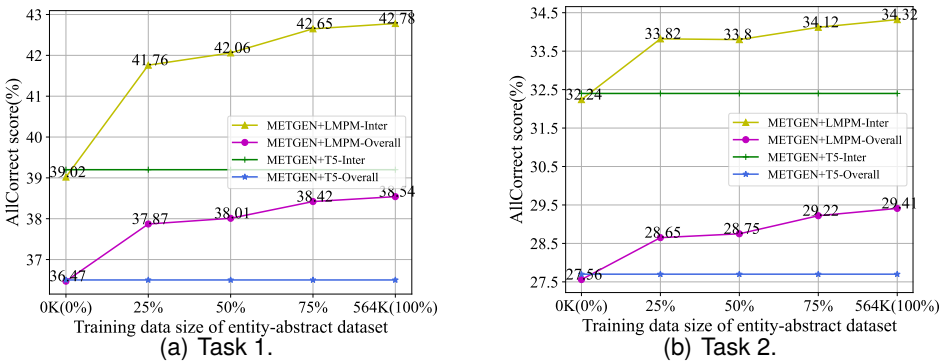
4.3.1. Automatic Evaluation Results

The automatic evaluation results are summarized in Table 2. In general, the multi-step methods (IRGR, RIET, and METGEN) outperform the single-step method (EntailmentWriter). This underscores the effectiveness of multi-step methods in selecting premises and generating intermediate conclusions that align closely with the ground truth. Benefiting from its endeavor to pre-train with additional Wikipedia-based synthetic data, METGEN has demonstrated robust and competitive performance, significantly outperforming the baseline models in both task 1 and task 2.

When comparing the performance of METGEN using different modules (T5 or LMPM) for intermediate conclusion generation, we observe a clear performance improvement when incorporating our proposed LMPM, especially in tasks 1 and task 2. Specifically, under the strictest *Overall AllCorrect*, the model with LMPM achieved performance improvements of 2.04% and 1.7%, respectively. The enhancements under the *Intermediates AllCorrect* metric further suggest that LMPM generates better intermediate conclusions with improved logical consistency. Additionally, the improved *Steps AllCorrect* indicate that LMPM positively impacts the premise selection process of the controller in METGEN. Although the incorporation of LMPM within METGEN generally outperforms the use of T5 in task 3, the improvements are not as pronounced. This could be attributed to the fact that task 3 utilizes the full-corpus distractor setting, which heavily relies on the accurate selection of premises.

4.3.2. Human Evaluation

In the human evaluation, we randomly sample 100 instances from the test set and compare the results generated by METGEN with different modules (T5 or LMPM). The results are presented in Table 3. METGEN+LMPM exhibits superior performance compared to METGEN+T5 across all metrics, particularly in terms of *Validity*, which indicates the quality of the complete entailment trees. In addition, *Logical* and *Readability* metrics also show enhanced performance, indicating LMPM's capacity to generate superior intermediate conclusions with logical consistency.



(a) Task 1. (b) Task 2.

Figure 5: The impact of logical pattern pre-training data size on performance. The AllCorrect scores for Overall and Inter (Intermediates) are provided.

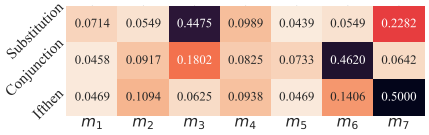


Figure 6: Probability distribution of the three logical patterns within the memory structure M.

4.3.3. Ablation Study

In the ablation study, we delve into the effectiveness of three key components: *logical pattern pre-training (LPP)*, *memory structure (memory)*, and *dataset abstraction (abstraction)*. Our analysis is conducted on both task 1 and task 2, with results presented in Table 4.

Firstly, we examine the model without *LPP*, which is exclusively trained based on the EntailmentBank Fine-tuning task. This model exhibits a significant performance degradation, highlighting the critical role of *LPP* in generating reasonable intermediate conclusions. However, even without *LPP*, the model's performance is on par with METGEN+T5, which is first pre-trained using the original synthetic dataset and then trained using the EntailmentBank. This suggets that the architecture of LMPM is more suitable than vanilla T5 in the entailment task. Next, we compare the model without a *memory structure*, having the same architecture as METGEN+T5 but first pre-trained on the entity-abstract dataset. The degraded performance of this model emphasizes the importance of the *memory structure*, which stores and learns logical patterns. Despite this, we observe that this model still outperforms METGEN+T5, indicating that the pre-training abstraction data can help generate better conclusions. This is further proved by the model's decreased performance when trained without *dataset abstraction* when incorporating LMPM. Finally, the model with the complete LMPM, which incorporates all the components simultaneously, achieves the best results.

4.4. Effect of Logical Pattern Data Size

We next investigate the influence of data sizes in the pre-trained entity-abstract dataset on both task 1 and task 2. To achieve this, we employ METGEN+LMPM and train the model using various data sizes. We also present the results of METGEN+T5, which is pre-trained using the entire Wikipedia-based synthetic dataset. The results are shown in Figure 5. Notably, a 0% data size implies that the entity-abstract dataset is not utilized, and the model is exclusively trained via the EntailmentBank Fine-tuning process, aligning with the model without LPP as detailed in Section 4.5.3. Additionally, we observe that even with a mere 25% of the data, the model achieves comparable performance to that using the entire dataset for both tasks. The entity-abstract dataset, in contrast to the original Wikipedia-based synthetic dataset, contains substantially less irrelevant knowledge within each logical pattern. This crucial distinction encourages the model to concentrate more precisely on the logical patterns, enabling effective training with a smaller amount of data.

4.5. Additional Analysis of Logical Pattern Representation

To gain a comprehensive understanding of the L logical patterns stored in the memory structure ($L = 7$ in the experiments), we further investigate the probability distribution of three different logical patterns within the memory structure M. We analyze a sample of 275 intermediate conclusions provided by METGEN, along with their automatically annotated inference types. The distribution of these samples is as follows: Substitution (153), Conjunction (71), and IF-then (51).

As depicted in Figure 6, we observe distinct clustering tendencies for different logical patterns. Specifically, Substitution mainly correlate with logical feature m_3 within memory M, while Conjunction is closely associated with feature m_6, and IF-then primarily relates to feature m_7. These observed patterns highlight the model's capability to capture

and differentiate various types of logical patterns. However, we also acknowledge that our model's relative neglect of a specific logical feature can be attributed to certain factors, such as potential errors introduced during the automated annotation process and instances necessitating a fusion of multiple logical pattern features.

5. Conclusion

In this paper, we present the logical pattern memory pre-trained model (LMPM), which significantly enhances the generation of intermediate conclusions in entailment steps by effectively harnessing logical patterns. LMPM utilizes an external memory structure to learn and store the latent representations of logical patterns, leveraging them to produce logically consistent conclusions. The model is pre-trained using a specially constructed entity-abstract dataset, tailored to facilitate the explicit acquisition of logical pattern features. Comprehensive evaluations demonstrate the superior ability of LMPM in facilitating the generation of better entailment trees, offering promising prospects in addressing the critical challenge of logical reasoning and paving the way for further explorations in complex reasoning tasks. For future work, we plan to extend this method to other tasks, such as the Multimodel QA, and to explore the incorporation of external knowledge to enhance tree generation.

Limitations

While the LMPM model demonstrates promising capabilities, several limitations need to be acknowledged. Firstly, the lack of diverse datasets for entailment tree generation in languages other than English hampers the comprehensive evaluation of LMPM's performance across multiple languages. Consequently, the model's effectiveness in generating logically consistent conclusions is primarily demonstrated within English entailment tree corpora. Secondly, LMPM may encounter challenges when tasked with generating logically consistent conclusions in scenarios involving more than two premises. Its performance in such cases might exhibit degradation, emphasizing the need for further enhancements to effectively manage complex entailment scenarios with multiple premises.

Acknowledgments

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.

References

James Allen. 1995. *Natural language understanding*. Benjamin-Cummings Publishing Co., Inc.

Alejandro Barredo Arrieta and Díaz-Rodríguez. 2020. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Inf. Fusion*, 58:82–115.

Kaj Bostrom, Xinyu Zhao, Swarat Chaudhuri, and Greg Durrett. 2021. Flexible Generation of Natural Language Deductions. In *Proceedings of the EMNLP 2021*, pages 6266–6278.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. *Advances in Neural Information Processing Systems*, 31.

Ruijun Chen, Jin Wang, Liang-Chih Yu, and Xuejie Zhang. 2023. Learning to memorize entailment and discourse relations for persona-consistent dialogues. In *Proceedings of the AAAI 2023*, pages 12653–12661.

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. *arXiv preprint arXiv:1803.05457*.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *Proceedings of the IJCAI 2020*, pages 3882–3890.

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 EMNLP 2021*, pages 7358–7370.

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 NAACL 2019*, pages 4171–4186.

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 ACL 2020*, pages 4443–4458.

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 NAACL 2022*, pages 1887–1905.

Qingbao Huang, Mingyi Fu, Linzhang Mo, Yi Cai, Jingyun Xu, Pijian Li, Qing Li, and Ho-Fung Leung. 2021. Entity Guided Question Generation with Contextual Structure and Sequence Information Capturing. In *Proceedings of the AAAI-22021*, pages 13064–13072.

Naoya Inoue, Pontus Stenetorp, and Kentaro Inui. 2020. R4C: A benchmark for evaluating RC systems to get the right answer for the right reason. In *Proceedings of the ACL*, pages 6740–6750.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. In *Proceedings of the ICLR 2016*.

Peter Jansen and Dmitry Ustalov. 2019. Textgraphs 2019 shared task on multi-hop inference for explanation regeneration. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 63–77.

Christopher D. Manning Jeffrey Pennington, Richard Socher. 2014. GloVe: Global vectors for word representation. In *Proceedings of the EMNLP-2014*, pages 1532–1543.

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 EMNLP 2020*, pages 137–150.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *Proceedings of the ICLR 2020*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the ACL 2020*, pages 7871–7880.

Xinmeng Li, Mamoun Alazab, Qian Li, Keping Yu, and Quanjun Yin. 2022. Question-aware memory network for multi-hop question answering in human–robot interaction. *Complex & Intelligent Systems*, 8(2):851–861.

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 EMNLP 2022*, pages 7177–7189.

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. *arXiv preprint arXiv:1907.11692*.

Vaibhav Mavi, Anubhav Jangra, and Adam Jatowt. 2022. A survey on multi-hop question answering and generation. *arXiv preprint arXiv:2204.09140*.

Danilo Neves Ribeiro, Shen Wang, Xiaofei Ma, Rui Dong, Xiaokai Wei, Henghui Zhu, Xinchi Chen, Peng Xu, Zhiheng Huang, Andrew Arnold, and Dan Roth. 2022. Entailment Tree Explanations via Iterative Retrieval-Generation Reasoner. In *Findings of the NAACL 2022*, pages 465–475.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Bo Peng, Jin Wang, and Xuejie Zhang. 2020. Adversarial learning of sentiment word representations for sentiment analysis. *Information Sciences*, 541:426–441.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2022. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

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

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the ACL 2019*, pages 4932–4942.

Haopeng Ren, Yi Cai, Raymond Y.K. Lau, Hofung Leung, and Qing Li. 2023. Granularity-aware area prototypical network with bimargin

loss for few shot relation classification. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4852–4866.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. Bleurt: Learning robust metrics for text generation. In *Proceedings of the ACL 2020*, pages 7881–7892.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional Attention Flow for Machine Comprehension. In *Proceedings of the ICLR 2017*, pages 1–13.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. 2021. ProofWriter: Generating Implications, Proofs, and Abductive Statements over Natural Language. In *Findings of the ACL 2021*, pages 3621–3634.

Ying Tai, Yun Cao, Junwei Zhu, and Chengjie Wang. 2022. Learning to Memorize Feature Hallucination for One-Shot Image Generation. In *Proceedings of the CVPR-2022*.

Jian Wang, Junhao Liu, Wei Bi, Xiaojiang Liu, Kejing He, Ruifeng Xu, and Min Yang. 2020. Dual dynamic memory network for end-to-end multi-turn task-oriented dialog systems. In *Proceedings of the Coling 2020*, pages 4100–4110.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of the ICLR 2015*.

Sarah Wiegreffe and Ana Marasović. 2021. Teach me to explain: A review of datasets for explainable nlp. *arXiv preprint arXiv:2102.12060*.

Xin Wu, Yi Cai, Zetao Lian, Ho-fung Leung, and Tao Wang. 2023. Generating natural language from logic expressions with structural representation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

Zhengnan Xie, Sebastian Thiem, Jaycie Martin, Elizabeth Wainwright, Steven Marmorstein, and Peter 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*, pages 5456–5473.

Nan Xu, Wenji Mao, and Guandan Chen. 2019. Multi-interactive memory network for aspect based multimodal sentiment analysis. In *Proceedings of the AAAI-2019*, pages 371–378.

Vikas Yadav, Steven Bethard, and Mihai Surdeanu. 2019. Quick and (not so) dirty: Unsupervised selection of justification sentences for multi-hop question answering. In *Proceedings of the EMNLP 2019*, pages 2578–2589.

Kaiyu Yang, Jia Deng, and Danqi Chen. 2022. Generating Natural Language Proofs with Verifier-Guided Search. In *Proceedings of the EMNLP 2022*, pages 89–105.

Qinyuan Ye, Xiao Huang, Elizabeth Boschee, and Xiang Ren. 2020. Teaching machine comprehension with compositional explanations. In *Findings of the EMNLP 2020*, pages 1599–1615.

Li Yuan, Jin Wang, Liang-Chih Yu, and Xuejie Zhang. 2020. Graph attention network with memory fusion for aspect-level sentiment analysis. In *Proceedings of the AACL 2020*, pages 27–36.

Li Yuan, Jin Wang, Liang-Chih Yu, and Xuejie Zhang. 2022. Hierarchical template transformer for fine-grained sentiment controllable generation. *Information Processing & Management*, 59(5):103048.

Li Yuan, Jin Wang, Liang-Chih Yu, and Xuejie Zhang. 2023. Encoding syntactic information into transformers for aspect-based sentiment triplet extraction. *IEEE Transactions on Affective Computing*, pages 1–15.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297.

Chunju Zhang, Xueying Zhang, Wenming Jiang, Qijun Shen, and Shanqi Zhang. 2009. Rule-based extraction of spatial relations in natural language text. In *Proceedings of the International Conference on Computational Intelligence and Software Engineering*, pages 1–4. IEEE.

You Zhang, Jin Wang, Liang-Chih Yu, and Xuejie Zhang. 2021. MA-BERT: Learning Representation by Incorporating Multi-Attribute Knowledge in Transformers. In *Findings of the Association for Computational Linguistics (ACL-IJCNLP 2021)*, pages 2338–2343.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders. In *Proceedings of the ACL 2017*, pages 654–664.

A. Appendix

A.1. Baselines

To evaluate the effectiveness and applicability of our proposed method, we apply LMPM to several existing multi-step generative methods and compare the performance of the integrated model with the original methods. The first multi-step method we compared is **METGEN** (Hong et al., 2022), which uses a reasoning controller and a T5 model to independently select premises and generate a conclusion. Note that the original synthesis Wikipedia dataset is first used to pre-train the METGEN model. In our comparison, we incorporate LMPM into METGEN by replacing their T5 model for intermediate conclusion generation. The second compared multi-step method is **NlProofS** (Yang et al., 2022), which uses a T5 to simultaneously select the premise and generate the conclusion, followed by a verifier. In our comparison, we integrated LMPM into NlProofS by maintaining their T5 model for premise selection and incorporating LMPM to generate intermediate conclusions. Note also that the EntailmentBank dataset contains some entailment steps (9.1%, 535 steps) where the intermediate conclusions are generated by more than two premises. METGEN decomposes such n-premise steps (n > 2) into several valid 2-premise steps before generating the conclusion. On the other hand, NLProofs can directly generate conclusions using more than two premises. Therefore, when applying LMPM to NLProofs, we retain their generated conclusions by T5 for the n-premise steps (n > 2). In all models mentioned, we freeze the parameters of the controller or verifier. Additionally, we use EntailmentWriter (Dalvi et al., 2021) as a baseline, which generates the entire entailment trees in one step.

A.2. Case Study

In this section, we utilize three entailment trees generated by METGEN-T5 and METGEN-LMPM, which were also employed in the human evaluation for the case study. Figure 7, 8, and 9 present the task input, golden tree, trees generated by METGEN-T5 and METGEN-LMPM, along with their respective Overall AllCorrect and human evaluation results. It should be noted that Overall AllCorrect is the most stringent automatic evaluation metric. Additionally, under the Validity category, we indicate the number of evaluators (out of 9) who deemed the generated tree as valid. Furthermore, the subgraph of METGEN-LMPM showcases the identified pattern relations.

Firstly, Figure 7 presents an illustrative example wherein LMPM outperforms T5 by generating a more accurate intermediate conclusion (c_1) that aligns closely with the ground truth and achieves a higher BLEURT score. Specifically, the BLEURT score for c_1 generated by T5 is 0.21, whereas LMPM achieves a score of 0.89. Furthermore, the metrics displayed in Figure 7 demonstrate that METGEN-LMPM outperforms METGEN-T5 in all aspects, owing to the generation of superior conclusions characterized by logical consistency during the entailment step.

Figure 8 presents an intriguing example. Notably, in METGEN-LMPM, all the human evaluation metrics yield favorable results, while the Overall AllCorrect score of the automatic evaluation remains 0. This suggests that the automatic evaluation might erroneously disregard some valid trees, thereby underestimating the performance, as also noted by Dalvi et al Dalvi et al. (2021) Furthermore, when compared to METGEN-T5, our proposed approach, METGEN-LMPM, adeptly captures the logical relationship between premises, leading to the generation of a logically consistent conclusion, denoted as C_1 and c_2. Consequently, we observe an improvement in the human evaluation metrics.

Lastly, concerning the golden tree comprising n-premise steps (where n > 2), METGEN allows the predicted entailment tree to exhibit a distinct structure from the golden tree. As described in Section 4.4, METGEN decomposes the n-premise step into several valid 2-premise steps prior to conclusion generation. In Figure 9, the intermediate conclusion (c_1) generated by METGEN-T5 fails to integrate the logical patterns and textual information from premises (s_1 and s_3), rendering it an invalid intermediate conclusion according to the Validity criterion evaluated by all the participants. Conversely, LMPM successfully extracts the logical pattern between the premises and employs the latent logical pattern to generate a more plausible conclusion, which is generally accepted by the majority of evaluators.

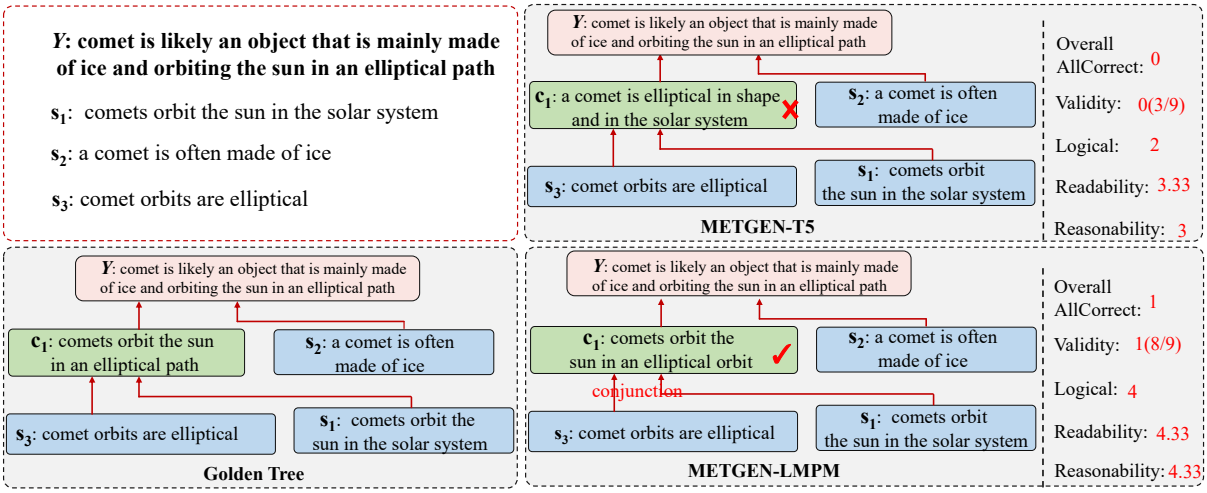


Figure 7: Case 1. A case shows that METGEN-LMPM achieves better results than METGEN-T5 in Overall AllCorrect metric and all metrics in human evaluations, and generates a better intermediate conclusion with logically consistency.

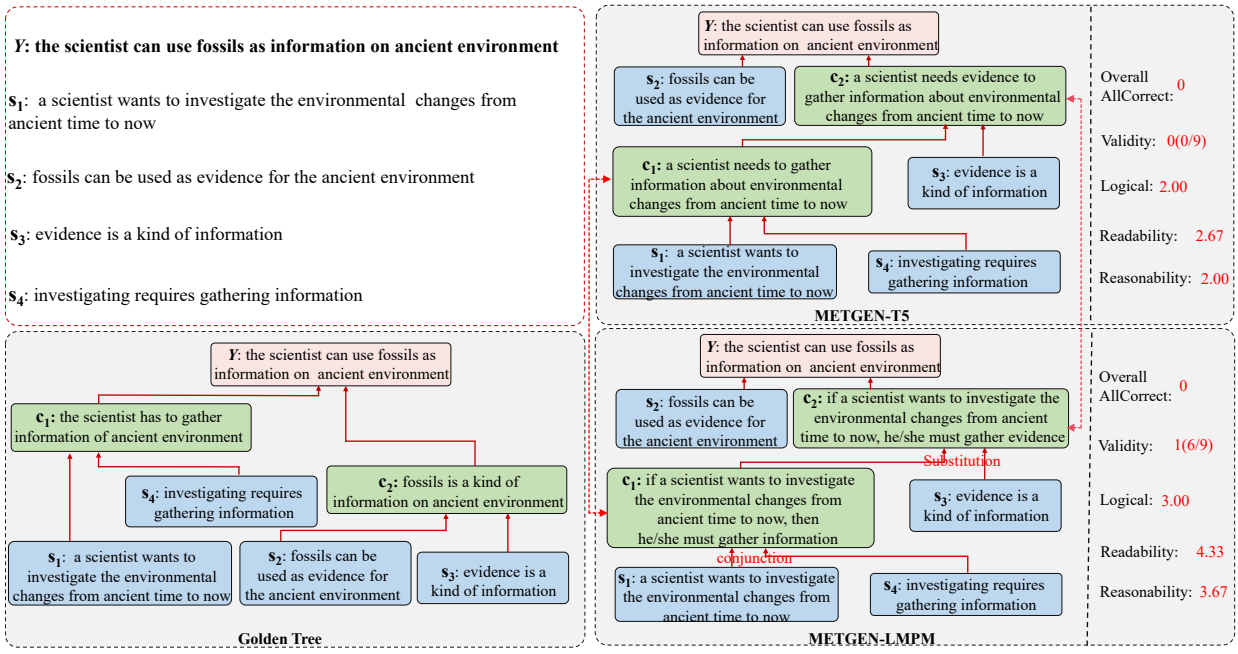


Figure 8: Case 2: An interesting case shows that automatic evaluation can misjudge certain valid trees, leading to an underestimated performance. This observation is based on the results obtained from METGEN-LMPM, where the generated entailment tree is considered valid by the majority of evaluators.

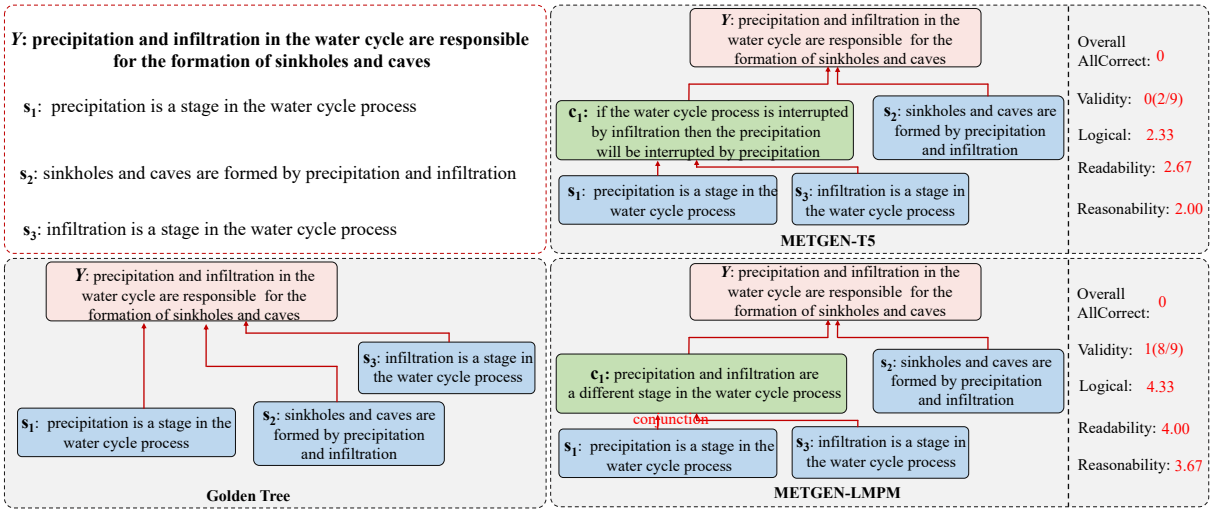


Figure 9: Case 3: A case shows how METGEN deals with the n-premise step (n > 2), and also shows that METGEN-LMPM can effectively fuse the logical patterns and textual information from the premises for generating the conclusions in the entailment steps.