RAG-Logic: Enhance Neuro-symbolic Approaches for Logical Reasoning with Retrieval-augmented Generation

Anonymous ACL submission

Abstract

Deductive reasoning over complex natural language poses significant challenges, necessitating the integration of large language models (LLMs). Benchmarks like ProofWriter and FOLIO highlight these challenges and demonstrate the need for advanced reasoning methods. Current approaches range from direct reasoning methods like zero-shot, few-shot, and chain-of-thought learning to hybrid models integrating LLMs with symbolic solvers. However, these methods often rely on static examples, limiting their adaptability. This paper introduces RAG-Logic, a dynamic example-based framework using Retrieval-Augmented Generation (RAG), which enhances LLMs' logical reasoning capabilities by providing contextually relevant examples. This approach conserves resources by avoiding extensive fine-tuning and reduces the propensity for hallucinations in traditional models. Our results across the ProofWriter and FOLIO datasets demonstrate the effectiveness of our framework, marking an advancement in logical reasoning tasks.

1 Introduction

Deductive reasoning over complex natural language presents significant challenges for artificial intelligence. Although symbolic solvers are good at handling formal logic, their applications are limited because many problems are typically represented in natural language. This necessitates the integration of large language models (LLMs) to serve as intermediaries in translating natural language into symbolic expressions.

Benchmarks such as ProofWriter (Tafjord et al., 2021) and FOLIO (Han et al., 2022) have underscored the challenges associated with logical reasoning in natural language contexts. FOLIO, in particular, highlights the substantial hurdles posed by its natural language diversity, making it an apt setting for evaluating advanced reasoning approaches.

Current reasoning methodologies range from direct approaches like zero-shot (Wei et al., 2022a), few-shot learning (Brown et al., 2020) and chain-of-thought (CoT) (Wei et al., 2022b), to hybrid strategies integrating LLMs with symbolic solvers, such as Logic-LM (Pan et al., 2023) and LINC (Olausson et al., 2023). These hybrid models, while innovative, often rely on fixed examples to guide reasoning, which can limit their effectiveness across varied contexts. More sophisticated systems like Symbolic Chain-of-Thought (SymbCoT) (Xu et al., 2024) and LeanReasoner (Jiang et al., 2024) utilize complex methodologies to advance logical reasoning tasks.

In this paper, we present a method for obtaining dynamic examples using Retrieval Augmented Generation (RAG), thus addressing the limitations of traditional prompt methods for fixed examples. RAG-Logic enhances the logical translation capabilities of LLM by dynamically providing contextually relevant examples to improve the accuracy of reasoning problems. This approach conserves computational resources by avoiding extensive fine-tuning and reduces the propensity of LLMs to generate hallucinations. Our experimental results across the ProofWriter and FOLIO datasets substantiate the efficacy of our framework, marking an advancement in logical reasoning tasks.

2 Related Work

Due to natural language's fuzziness, ambiguity, and frequent implicit information (such as emotions), accurately translating natural language sentences into first-order logic (FOL) poses a challenge for LLMs. Nguyen et al. (2022) proposed a method combining manually translated rules or automatically logical fact formulas with deep learning to enhance translation quality. Yang et al. (2023) introduced LOGICLLAMA, which improves accuracy through supervised fine-tuning of the CoT step and reinforcement learning with human feedback. Chen et al. (2023) developed an instructive framework focused on temporal logic, employing large LLMs to facilitate bidirectional translation between natural language (NL) and signal temporal logic (STL) formats, utilizing intermediate languages such as Lifted NL and Lifted STL in this process. This approach enabled them to generate accurate NL-STL pairs. Our research also deals with translation from NL to FOL, emphasizing the consistency of information in the translation work.

Neuro-symbolic methods for logical reasoning have gained attention recently. Pan et al. (2023) employed Logic-LM, a framework based on the neuro-symbolic method, introducing a self-refiner module to address unfaithful reasoning in LLMs. Olausson et al. (2023) introduced LINC, a system that integrates neural and symbolic methods to enhance logical reasoning. This integration employs a Logic Theorem Prover and incorporates a majority voting mechanism to refine the effectiveness of logical reasoning. Jiang et al. (2024) proposed LeanReasoner with a tactic generator and proof search to reduce problem complexity. Xu et al. (2024) proposed SymbCoT for converting natural language to logic (although SymbCoT solved the problem using the CoT method instead of a symbolic solver.). RAG-Logic also follows the "LLM translation + external solver" structure, but it incorporates the RAG method to enhance translation quality. Lewis et al. (2020) first proposed the idea of RAG. They studied a RAG model retrieving documents from a library to inform output generation, yielding more precise and factual results by leveraging external knowledge sources. Ding et al. (2024) highlighted RAG's integration of external data to reduce model hallucinations and augment the generation quality. Additionally, Jiang et al. (2023) introduced FLARE, enabling efficient retrieval of necessary information by language models during generation. Our research focuses on the RAG method's role in NL-FOL translation, demonstrating its capability to enhance logical reasoning in RAG-Logic frameworks.

3 RAG-Logic

In this section, we describe the framework of RAG-Logic as depicted in Figure 1. The primary idea of the framework is to enhance logical reasoning capabilities by integrating RAG with symbolic solvers. The framework is divided into four main modules: the RAG Knowledge Base Search Module, the Translation Module, the Fix Module, and the Solver Module. The prompts for each part are in Appendix A.1.

RAG Knowledge Base Search Module: This module performs a search in a pre-built knowledge base using the natural language premise in the example. It selects the examples in the knowledge base that are most similar to the current input sentence for use in subsequent modules. The basis for the knowledge base query is vector similarity, utilizing the text-embedding-3-small embedding model[1]. The similarity function used is cosine similarity, calculated as follows:

$$\text{Cosine Similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|},$$

where $\mathbf{A}$ and $\mathbf{B}$ are the vector representations of the sentences.

The knowledge base consists of two types: queries on all premises and queries on a single premise. The result returned by the knowledge base is the FOL formula for similar sentences. For example, if the input is: *"All squares have four sides."* we might get: *"All tables are round.* $\forall x(\textit{Table}(x) \rightarrow \textit{Round}(x))$ *....."* Whereas, if the input is another sentence like: *"If George likes music, he wants to compose."* we might get: *"If Sam has high ambitions and future career goals, then Sam is a big fan of pop bands and singers.* $\textit{Ambitious}(sam) \rightarrow \textit{Fans}(sam)$ *....."*

Translation Module: This module uses a specific prompt for LLMs to translate natural language sentences into logical formulas. The prompt includes examples from RAG, definitions of logical operators, instructions for avoiding certain symbols, and guidelines for building FOL rules using appropriate quantifiers and variables. Detailed prompts are provided in Appendix A.1 for reference.

Fix Module: This module ensures the syntactical correctness of the translated FOL formulas

[1]https://openai.com/index/new-embedding-models-and-api-updates/

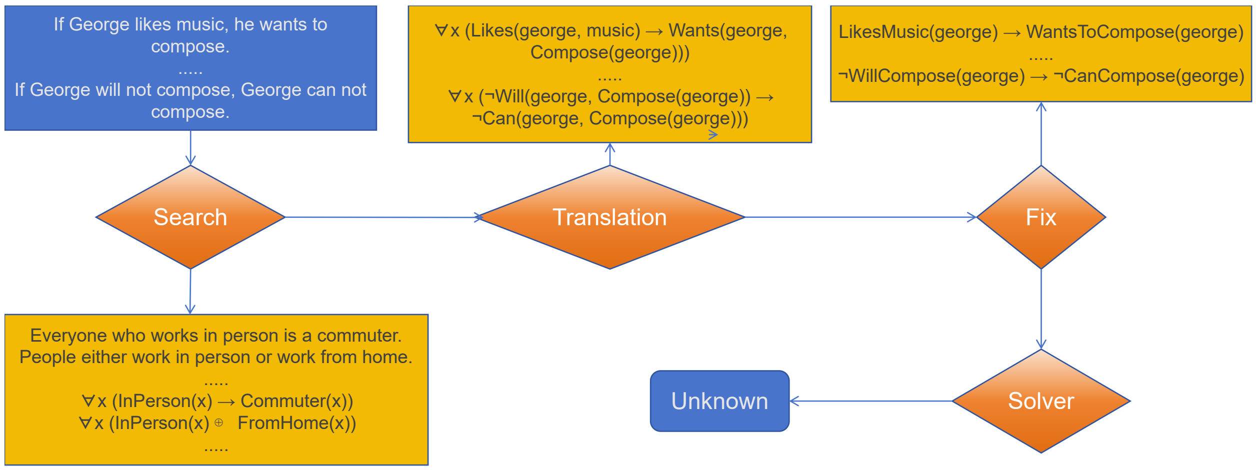

Figure 1: The framework of proposed RAG-Logic.

by detecting and correcting syntax errors, providing contextually relevant examples from the RAG search to guide corrections. For example, the input: $\forall x(\text{Tall}(x, \text{Strong}(x)))$ contains a predicate stacking error. The corrected formula would be: $\forall x(\text{Tall}(x) \rightarrow \text{Strong}(x))$. For FOLIO, it also includes domain correction to address predicate and domain repetition issues. If a predicate only appears in the antecedent of an implication and signifies that x belongs to a certain category, then the predicate will be removed, provided that the domain of discourse in the entire context includes this category. For example, within a domain where the context encompasses only humans if there are 2 premises, “A person is either a man or a woman. All men are tall.” then in $\forall x\,(\text{Person}(x) \rightarrow (\text{Man}(x) \vee \text{Woman}(x)))$; $\forall x\,(\text{Man}(x) \rightarrow \text{Tall}(x)$, the predicate "Person" can be omitted.

Solver Module: This module evaluates the logical consistency of the translated formulas using the Z3 solver [2]. It inputs the translated FOL premises and a conclusion into the solver and determines whether the conclusion is implied by the premises, labeling the conclusion as True, False, or Unknown based on the solver’s output.

[2]https://github.com/Z3Prover/z3

4 Experiments

4.1 Comparative Methods

We employ four models for comparison: gpt-3.5-turbo-0125 (GPT-3.5)[3], claude-3-haiku-20240307 (Claude3)[4], deepseek-chat (Deepseek) (DeepSeek-AI et al., 2024) and gpt-4o-2024-05-13 (GPT-4o)[5]. Each model is evaluated using the following methods: CoT, Few-shot prompting translation with a symbolic solver (FS), Few-shot prompting translation with the fix module and symbolic solver ($\text{FS}_{\textit{Fix}}$), RAG with symbolic solver (RAG-L), RAG with the fix module and symbolic solver ($\text{RAG-L}_{\textit{Fix}}$).

4.2 Setting

For the few-shot configuration, the prompt contains three fixed examples. For the RAG configurations, the prompt includes the three most similar examples retrieved from the knowledge base. The RAG and Fix configuration additionally queries five examples of single error sentences for repair guidance.

4.3 Datasets

ProofWriter: For ProofWriter, we randomly selected 180 examples with balanced label distribution from the 2-depth and 3-depth test subsets to form the test set. Additionally, 1200 examples from the 2-depth and 3-depth train subsets were used for the

[3]https://openai.com/index/new-embedding-models-and-api-updates/
[4]https://www.anthropic.com/news/claude-3-haiku
[5]https://openai.com/index/hello-gpt-4o/

RAG training set. We use Deepseek to convert training examples into FOL formulas. After verifying consistency with the labeled answers using a solver, we select the correct parts to add to the knowledge database.

To ensure the extracted results are not overly simplistic, we avoid instances where the problem statement is identical to or merely negates the premises. Since the examples in FOLIO have 3-9 premises, we only choose examples with 3-9 premises in ProofWriter.

FOLIO: For FOLIO, following the methodology from LINC, problematic examples were removed, leaving 181 examples for the test set. The entire training set was used as the knowledge base.

4.4 Results

Table 1: Results of Models on ProofWriter.

Model	CoT	FS	FS$_{Fix}$	RAG-L	RAG-L$_{Fix}$
GPT-3.5	57.22	88.33	94.41	95.00	**96.67**
Claude3	61.67	92.82	95.58	**96.69**	**96.69**
Deepseek	88.89	96.11	96.67	**97.22**	**97.22**
GPT-4o	92.22	94.44	97.22	97.22	**97.78**

Table 2: Results of Models on FOLIO.

Model	CoT	FS	FS$_{Fix}$	RAG-L	RAG-L$_{Fix}$
GPT-3.5	50.55	46.70	55.68	53.30	**56.82**
Claude3	53.85	54.95	68.16	58.10	**71.35**
Deepseek	63.74	55.49	68.89	60.34	**74.72**
GPT-4o	64.84	62.64	72.38	67.06	**75.14**

Table 1 presents the results of various methods on ProofWriter, demonstrating the effectiveness of our framework and signifying the incremental advancement contributed by our fix module. RAG-L consistently achieves higher accuracy than both FS and CoT (RAG-L has an average of 97% accuracy), regardless of the model, likely due to its ability to search for suitable examples to aid in formula translation. The impact of the fix module is less pronounced, possibly because the language used in ProofWriter is relatively simple, allowing various methods to achieve high accuracy (Especially FS and RAG-L, which are basically above 90%) and leaving limited room for improvement.

Table 2 presents the experimental results on FOLIO, reaffirming the effectiveness of the RAG-Logic framework and underscoring the role of the fix module. The RAG-L$_{Fix}$ shows improvement compared to both FS and CoT, achieving the highest score of approximately 75% (GPT-4o and Deepseek), although not as high as on ProofWriter. Notably, CoT performs better than unfixed FS or RAG-L on some models, likely because FOLIO's language is closer to natural language, making it more suitable for direct processing by LLMs. However, our fix module effectively addresses this, resulting in more than an 8% improvement across various models (except for GPT-3.5 under RAG-L). This indicates that while LLM translation results are "almost" correct, they may contain minor errors due to natural language complexity, which the fix module successfully mitigates.

5 Conclusion

In this paper, we presented a novel framework, RAG-Logic, designed to enhance neuro-symbolic approaches for logical reasoning with RAG. Our framework addresses the inherent challenges of translating NL into FOL by leveraging the contextual richness and relevance provided by RAG. Through comprehensive experiments on ProofWriter and FOLIO datasets, we demonstrated that our method improves the accuracy and reliability of logical translations, outperforming traditional CoT prompting and few-shot prompting translation with a symbolic solver.

Future work should focus on enhancing the relevance of knowledge base searches, potentially by combining embeddings with syntactical analysis to ensure the retrieval of the most pertinent examples. Additionally, constructing a more comprehensive and high-quality RAG knowledge base is crucial for further improving the performance of logical translations. By refining these aspects, we aim to push the boundaries of what can be achieved in logical reasoning tasks, making AI systems more reliable, accurate, and capable of complex cognitive processes.

Limitations

The RAG-Logic framework introduced in this paper, while showing enhancements in logical reasoning capabilities, presents certain limitations:

1. **Opacity of Vector Similarity**: The RAG-Logic framework relies on vector similarity for retrieving relevant examples from a knowledge base, inherently characterized by a "black-box" nature. Vector similarity may not capture the full logical and semantic complexity of sentences, sometimes leading to the retrieval of examples that are less relevant to the current problem. Despite the use of advanced text embedding models, the decision processes and feature capturing of these models remain insufficiently transparent.

2. **Impact of the Number of Knowledge Base Examples**: In RAG-Logic, the number of examples chosen from the knowledge base to aid logical reasoning directly impacts the accuracy and efficiency of the reasoning process. Too few examples can lead to insufficient information, and failing to provide effective logical support; conversely, too many examples might increase processing complexity and computational cost without necessarily leading to a linear improvement in performance. Determining the optimal number of examples generally requires adjustment based on experience, lacking a systematic method to predict the best solution.

3. **Dependence on the Quality and Coverage of External Knowledge Bases**: The effectiveness of RAG-Logic heavily depends on the quality and coverage of external knowledge bases. If the data in the knowledge base is of low quality or covers a narrow range, it might lead to the retrieval of inaccurate or incomplete information, thereby affecting the correctness of the reasoning results.

References

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

Yongchao Chen, Rujul Gandhi, Yang Zhang, and Chuchu Fan. 2023. NL2TL: transforming natural languages to temporal logics using large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 15880–15903. Association for Computational Linguistics.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, Hao Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, Tao Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, and Xiaowen Sun. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *CoRR*, arXiv:2405.04434.

Yujuan Ding, Wenqi Fan, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on RAG meets llms: Towards retrieval-augmented large language models. *CoRR*, arXiv:2405.06211.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, David Peng, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Shafiq R. Joty, Alexander R. Fabbri, Wojciech Kryscinski, Xi Victoria Lin, Caiming Xiong, and Dragomir Radev. 2022. FOLIO: natural language reasoning with first-order logic. *CoRR*, arXiv:2209.00840.

Dongwei Jiang, Marcio Fonseca, and Shay B. Cohen.

2024. Leanreasoner: Boosting complex logical reasoning with lean. *CoRR*, arXiv:2403.13312.

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

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

Ha-Thanh Nguyen, Wachara Fungwacharakorn, Fumihito Nishino, and Ken Satoh. 2022. A multi-step approach in translating natural language into logical formula. In *Legal Knowledge and Information Systems - JURIX 2022: The Thirty-fifth Annual Conference, Saarbrücken, Germany, 14-16 December 2022*, volume 362 of *Frontiers in Artificial Intelligence and Applications*, pages 103–112. IOS Press.

Theo Olausson, Alex Gu, Benjamin Lipkin, Cedegao E. Zhang, Armando Solar-Lezama, Joshua B. Tenenbaum, and Roger Levy. 2023. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 5153–5176. Association for Computational Linguistics.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 3806–3824. 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.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022a. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024. Faithful logical reasoning via symbolic chain-of-thought. *Preprint*, arXiv:2405.18357.

Yuan Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. 2023. Harnessing the power of large language models for natural language to first-order logic translation. *CoRR*, arXiv:2305.15541.

A Appendix

A.1 Prompts

1. CoT

This task requires an analysis of the logical connections between a series of premises and a specified conclusion to determine the validity of the conclusion. The analysis is grounded in first-order logic. The objective is to evaluate if the conclusion is logically supported by the premises provided. Please use <label></label> tags to categorize the final assessment of the conclusion as 'True', 'False', or 'Unknown', facilitating streamlined processing.

Task Description

Input:

- Premises: A set of statements presented in first-order logic.
- Conclusion: A statement that needs to be evaluated against the premises.

Instructions:

1. Read and Understand the Premises and Conclusion:
    - <premises> {premises} </premises>
    - <conclusion> {conclusion} </conclusion>
2. Analyze the Logical Relationship:
    - Determine if the logical flow supports the conclusion based on the premises.
3. Evaluation and Labeling:
    - Based on the analysis, decide if the conclusion is:
        - True: The conclusion logically follows from the premises.
        - False: The conclusion does not logically follow from the premises.
        - Unknown: It is unclear or there is insufficient information to determine the relationship.
4. Final Output:
    - Clearly state your final assessment of the conclusion. Encapsulate your decision ('True', 'False', or 'Unknown') within <label></label> tags for clarity.
    - Example: '<label>True</label>'

Remember, your final decision must be enclosed within <label></label> tags to enhance the model's result processing capability.

Let's think step by step.

2. Translator Module

Role: Logic Translator

For FOL rule generation

1. You SHOULD USE the following logical operators: $\oplus$ (either or), $\vee$ (disjunction), $\wedge$ (conjunction), $\rightarrow$ (implication), $\forall$ (universal), $\exists$ (existential), $\neg$ (negation), $\leftrightarrow$ (equivalence)
2. You SHOULD NEVER USE the following symbols for FOL: "","$\neq$", "%", "="
3. The literals in FOL SHOULD ALWAYS have predicate and entities, e.g., "Rounded(x, y)" or "City(guilin)"; expressions such as "y = a $\vee$ y = b" or "a $\wedge$ b $\wedge$ c" are NOT ALLOWED
4. The FOL rule SHOULD ACCURATELY reflect the meaning of the NL statement
5. You SHOULD ALWAYS put quantifiers and variables at the beginning of the FOL
6. You SHOULD generate FOL rules with either:
    (a) no variables;
    (b) one variable "x";
    (c) two variables "x", "y";
    (d) three variables "x", "y" and "z"

Example to learn

Current task:

Convert the following length lines natural language sentences into length first-order logical formulas.

- <NL> </NL>

Output format

Use <FOL> and </FOL> to wrap the FOL formulas.

The formulas you output in the <FOL> tag should correspond line by line with the content in the <NL> tag.

Each line in the tag should be a single FOL formula.

You can analyze task during your output. But don't use natural language in the final <FOL> tag.

Let's think step by step.

3. Fix Module 1

Role: Logic Corrector

- Goals
    - Enhance the compatibility of first-order logic (FOL) formulas with formal verification tools by ensuring syntactical correctness and adherence to formal logic syntax.

- Automatically identify and suggest corrections for common syntax errors in FOL formulas to facilitate their processing by logic verifiers.

For FOL rule generation

1. You SHOULD USE the following logical operators: $\oplus$ (either or), $\vee$ (disjunction), $\wedge$ (conjunction), $\rightarrow$ (implication), $\forall$ (universal), $\exists$ (existential), $\neg$ (negation), $\leftrightarrow$ (equivalence)
2. You SHOULD NEVER USE the following symbols for FOL: "","$\neq$", "%", "="
3. The literals in FOL SHOULD ALWAYS have predicate and entities, e.g., "Rounded(x, y)" or "City(guilin)"; expressions such as "y = a $\vee$ y = b" or "a $\wedge$ b $\wedge$ c" are NOT ALLOWED
4. The FOL rule SHOULD ACCURATELY reflect the meaning of the NL statement
5. You SHOULD ALWAYS put quantifiers and variables at the beginning of the FOL
6. You SHOULD generate FOL rules with either:
   (a) no variables;
   (b) one variable "x";
   (c) two variables "x", "y";
   (d) three variables "x", "y" and "z"

Output format

Use <FOL> and </FOL> to wrap the FOL formulas.

Each line in the tag should be a single FOL formula.

You can analyze task during your output. But don't use natural language in the final <FOL> tag.

Only signal <FOL> can be in your reply.

Example to learn

Current task:

- <NL> </NL>
- {err_msg}

Firstly, follow the rules above and reply your idea about the error message.

Secondly, write length FOL formulas after fixed in the following tag <FOL> which like '<FOL>Your answer</FOL>'.

Let's think step by step.

4. Fix Module 2

Role: Logic Corrector

For FOL rule generation

1. You SHOULD USE the following logical operators: $\oplus$ (either or), $\vee$ (disjunction), $\wedge$ (conjunction), $\rightarrow$ (implication), $\forall$ (universal), $\exists$ (existential), $\neg$ (negation), $\leftrightarrow$ (equivalence)
2. You SHOULD NEVER USE the following symbols for FOL: "","$\neq$", "%", "="
3. The literals in FOL SHOULD ALWAYS have predicate and entities, e.g., "Rounded(x, y)" or "City(guilin)"; expressions such as "y = a $\vee$ y = b" or "a $\wedge$ b $\wedge$ c" are NOT ALLOWED
4. The FOL rule SHOULD ACCURATELY reflect the meaning of the NL statement
5. You SHOULD ALWAYS put quantifiers and variables at the beginning of the FOL
6. You SHOULD generate FOL rules with either:
   (a) no variables;
   (b) one variable "x";
   (c) two variables "x", "y";
   (d) three variables "x", "y" and "z"

Output format

Use <FOL> and </FOL> to wrap the FOL formulas.

Each line in the tag should be a single FOL formula. You can analyze task during your output.But don't use natural language in the final <FOL> tag.

Example to learn

Background Information

- <NL> </NL>

The formulas below may contain errors.

- <FOL> </FOL>

Current task:

Firstly,follow the rules above and reply your idea about the error message.

Secondly,write only one FOL formula for one line in the following tag <FOL> which like <FOL>Your answer</FOL>. Let's think step by step.

A.2 The results of Models

The data in the charts are the same as Table 1 and Table 2. A single chart is more intuitive to compare the differences between different methods under the same model.

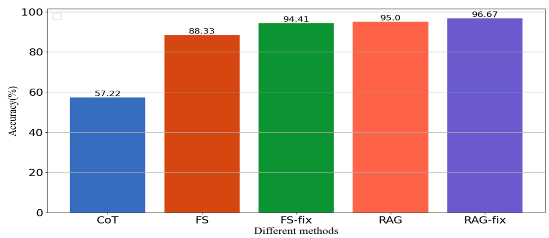

Figure 2: Results of GPT-3.5 on ProofWriter.

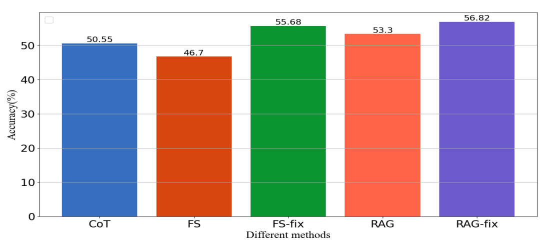

Figure 3: Results of GPT-3.5 on FOLIO.

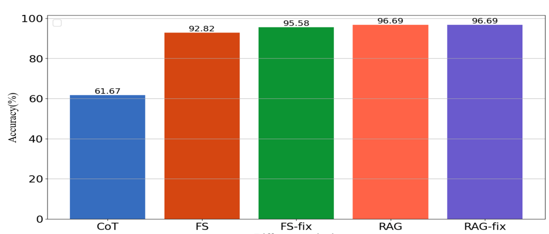

Figure 4: Results of Claude3 on ProofWriter.

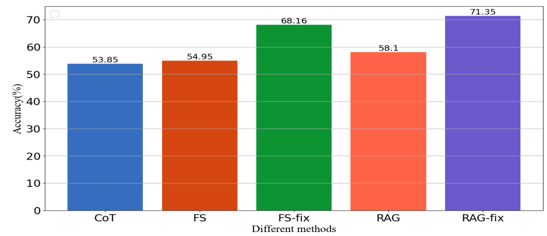

Figure 5: Results of Claude3 on FOLIO.

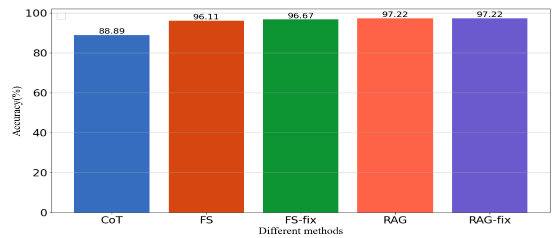

Figure 6: Results of Deepseek on ProofWriter.

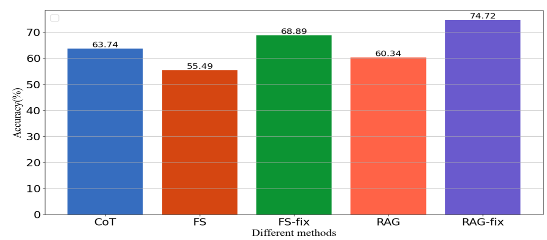

Figure 7: Results of Deepseek on FOLIO.

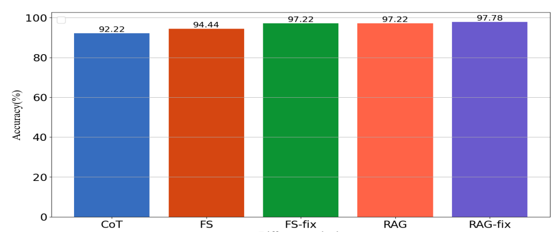

Figure 8: Results of GPT-4o on ProofWriter.

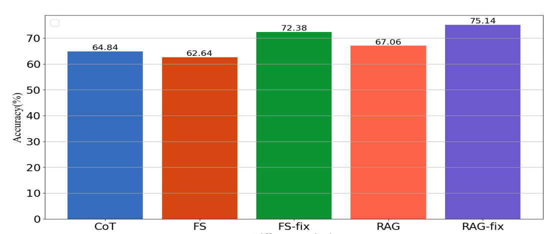

Figure 9: Results of GPT-4o on FOLIO.