GraphReason: Enhancing Reasoning Capabilities of Large Language Models through A Graph-Based Verification Approach

Lang Cao
University of Illinois Urbana-Champaign
Department of Computer Science
langcao2@illinois.edu

Abstract

Large Language Models (LLMs) have showcased impressive reasoning capabilities, particularly when guided by specifically designed prompts in complex reasoning tasks such as math word problems. These models typically solve tasks using a chain-of-thought approach, which not only bolsters their reasoning abilities but also provides valuable insights into their problem-solving process. However, there is still significant room for enhancing the reasoning abilities of LLMs. Some studies suggest that the integration of an LLM output verifier can boost reasoning accuracy without necessitating additional model training. In this paper, we follow these studies and introduce a novel graph-based method to further augment the reasoning capabilities of LLMs. We posit that multiple solutions to a reasoning task, generated by an LLM, can be represented as a reasoning graph due to the logical connections between intermediate steps from different reasoning paths. Therefore, we propose the Reasoning Graph Verifier (GraphReason) to analyze and verify the solutions generated by LLMs. By evaluating these graphs, models can yield more accurate and reliable results.Our experimental results show that our graph-based verification method not only significantly enhances the reasoning abilities of LLMs but also outperforms existing verifier methods in terms of improving these models' reasoning performance.

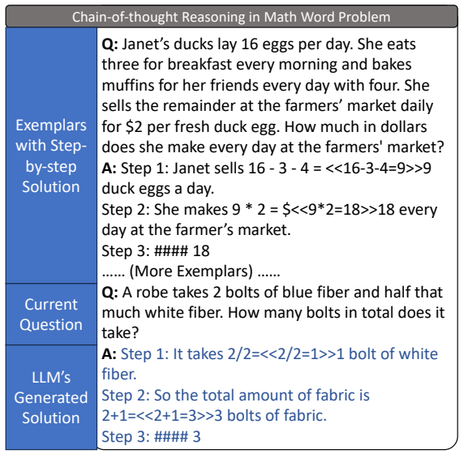


Figure 1: An example of chain-of-thought reasoning in a math word problem, using data from the GSM8K dataset. Large language models learn from exemplars that provide step-by-step solutions, subsequently generating their reasoning path for the current question.

1 Introduction

Large Language Models (LLMs) have demonstrated exceptional capabilities in a variety of human tasks (Zhao et al., 2023). Among the many abilities LLMs possess, their reasoning capacity is of paramount importance (Kojima et al., 2023; Huang and Chang, 2023). This has been substantiated by recent progresses (Wei et al., 2022; Zhou et al., 2023; Lampinen et al., 2022a). Equipped with the ability to reason, especially in a multi-step manner, LLMs can decompose complex problems into simpler tasks, thereby facilitating their resolution. In everyday life, many complex tasks typically require multi-step solutions. A prime example of a reasoning task is arithmetic reasoning, also known as solving math word problems (Zhang et al., 2019). These math word problems represent simplified versions of complex real-life situations.

The reasoning ability is inherent in Large Language Models (LLMs), but it necessitates specific methods for manifestation. To activate the robust reasoning capability of LLMs, the use of specially designed prompts should be considered. Numerous methods have been proposed to tap into this potential, among which chain-of-thought reasoning (Wei et al., 2022) and in-context learning (Lampinen et al., 2022b) are two notable approaches. Chain-of-thought reasoning can elucidate the reasoning paths during the process. In-context learning fur-

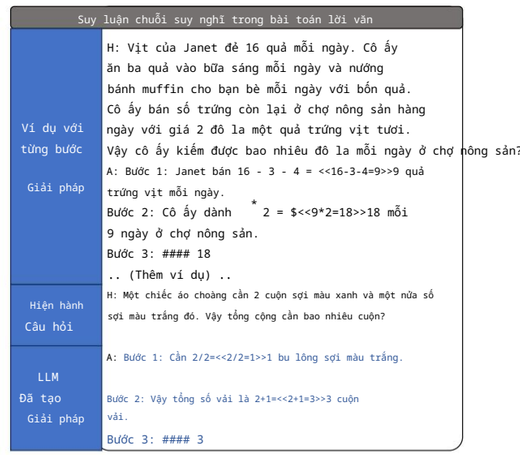
Proceedings of the 2nd Workshop on Natural Language Reasoning and Structured Explanations (@ACL 2024), pages 1–12

GraphReason: Nâng cao khả năng suy luận của ngôn ngữ lớn Mô hình thông qua phương pháp xác minh dựa trên đồ thị

Lăng Cao
Đại học Illinois Urbana-Champaign
Khoa Khoa học Máy tính
langcao2@illinois.edu

Tóm tắt

Các Mô hình Ngôn ngữ Lớn (LLM) đã thể hiện khả năng suy luận ấn tượng, đặc biệt là khi được hướng dẫn bởi các mô hình được thiết kế riêng lời nhắc trong các nhiệm vụ lý luận phức tạp như các bài toán đố. Các mô hình này thường giải quyết các nhiệm vụ bằng cách sử dụng phương pháp tiếp cận chuỗi suy nghĩ, điều này không chỉ củng cố khả năng lý luận của họ nhưng cũng cung cấp những hiểu biết có giá trị về quá trình giải quyết vấn đề. Tuy nhiên, có vẫn còn nhiều chỗ đáng kể để nâng cao khả năng lập luận của LLM. Một số nghiên cứu cho thấy rằng sự tích hợp của một trình xác minh đầu ra LLM có thể tăng cường độ chính xác của lý luận mà không cần phải đào tạo mô hình bổ sung. Trong bài báo này, chúng tôi theo dõi những nghiên cứu này và giới thiệu một cuốn tiểu thuyết phương pháp dựa trên đồ thị để tăng cường hơn nữa khả năng suy luận của LLM. Chúng tôi đưa ra giả thuyết rằng nhiều giải pháp cho một nhiệm vụ suy luận, được tạo ra bởi một LLM, có thể được biểu diễn như một lý luận đồ thị do các kết nối logic giữa các bước trung gian từ các con đường suy luận khác nhau. Do đó, chúng tôi đề xuất Đồ thị lý luận Trình xác minh (GraphReason) để phân tích và xác minh các giải pháp được tạo ra bởi LLM. Bằng cách đánh giá các biểu đồ này, các mô hình có thể mang lại nhiều kết quả chính xác và đáng tin cậy. Thí nghiệm của chúng tôi kết quả cho thấy rằng xác minh dựa trên đồ thị của chúng tôi phương pháp này không chỉ nâng cao đáng kể khả năng lý luận của LLM mà còn vượt trội hơn các phương pháp xác minh hiện có về mặt cải thiện hiệu suất suy luận của các mô hình này.



Hình 1: Một ví dụ về lý luận chuỗi suy nghĩ trong một bài toán bằng lời, sử dụng dữ liệu từ GSM8K tập dữ liệu. Các mô hình ngôn ngữ lớn học hỏi từ các ví dụ cung cấp các giải pháp từng bước, sau đó tạo ra con đường lý luận cho câu hỏi hiện tại.

1 Giới thiệu

Các Mô hình Ngôn ngữ Lớn (LLM) đã chứng minh khả năng đặc biệt trong nhiều nhiệm vụ của con người (Zhao và cộng sự, 2023). Trong số nhiều khả năng mà các LLM sở hữu, khả năng lập luận của họ có tầm quan trọng tối cao (Kojima et al., 2023; Huang và Chang, 2023). Điều này đã được chứng minh bằng những tiến bộ gần đây (Wei và cộng sự, 2022; Zhou và cộng sự, 2023; Lampinen và cộng sự, 2022a). Được trang bị với khả năng suy luận, đặc biệt là theo cách nhiều bước, LLM có thể phân tích các vấn đề phức tạp thành các nhiệm vụ đơn giản hơn, do đó tạo điều kiện thuận lợi cho họ giải quyết. Trong cuộc sống hàng ngày, nhiều nhiệm vụ phức tạp thường yêu cầu các giải pháp nhiều bước. Một ví dụ điển hình của nhiệm vụ lập luận là lập luận số học, còn được gọi là giải bài toán bằng lời văn (Zhang và cộng sự, 2019). Những bài toán đố này thể hiện phiên bản đơn giản hóa của những tình huống phức tạp trong đời thực.

Khả năng lý luận vốn có trong các Mô hình Ngôn ngữ Lớn (LLM), nhưng nó đòi hỏi những điều kiện cụ thể phương pháp biểu hiện. Để kích hoạt mạnh mẽ khả năng lập luận của LLM, việc sử dụng đặc biệt nên cân nhắc các lời nhắc được thiết kế. Nhiều các phương pháp đã được đề xuất để khai thác tiềm năng này , trong đó có lý luận chuỗi suy nghĩ (Wei et al., 2022) và học tập theo ngữ cảnh (Lampinen et al., 2022b) là hai cách tiếp cận đáng chú ý. Lý luận chuỗi suy nghĩ có thể làm sáng tỏ lý luận các con đường trong quá trình. Học tập theo ngữ cảnh

Biên bản Hội thảo lần thứ 2 về Lý luận ngôn ngữ tự nhiên và Giải thích có cấu trúc (@ACL 2024), trang 1–12

nishes LLMs with exemplary cases, thereby enabling them to learn from and simulate these examples for improved results. In the arithmetic reasoning scenario, GPT-4 can achieve an accuracy of 92% on the GSM8K dataset using 5-shot chain-of-thought prompts (Cobbe et al., 2021a). This represents a level of difficulty that a bright middle school student should be capable of handling. As depicted in Figure 1, this illustrates a multi-step arithmetic reasoning process in LLMs.

In addition to further training of LLMs and prompt design, some methods have been proposed to enhance the reasoning capabilities of LLMs from the perspective of output verification. The primary idea is to have LLMs generate reasoning paths multiple times, and then design a verifier to evaluate these paths and deliver the final results. (Wang et al., 2023) introduces the concept of *self-consistency*, based on the intuition that a complex reasoning problem usually allows for multiple thought processes, all leading to a unique correct answer. (Li et al., 2023) also proposes *All Roads Lead to Rome*, which introduces a step-aware verifier to analyze reasoning paths not just through the entire path, but at every step. However, both methods treat each reasoning path as an independent entity and do not consider the potential interrelation and interaction between different reasoning paths. Once reasoning paths are disassembled into steps, intermediate steps from one path may bear reasoning relations to other reasoning paths. These methods do not perceive all LLM outputs for a given input as a collective entity, thereby failing to analyze the internal relations of all candidate paths in depth.

Inspired by these observations, we propose **Reasoning Graph** Verifier (GraphReason) in this paper. We posit that reasoning paths of one question can form reasoning graphs, where similar intermediate reasoning steps can be merged into the same node. With a graph structure, we can more effectively model and capture the reasoning logic between intermediate steps from different reasoning paths. Specifically, we first construct a reasoning graph based on all outputs from LLMs, and then train a verifier to learn the relationship between the graph structure and the final answer. During the prediction stage, we process the data in the same way as in the training stage, and use the verifier to evaluate each reasoning graph. We then select the reasoning graph with the highest score, using its answer as the final answer. To the best of our knowledge, we are the first to approach reasoning logic of LLMs from a graph perspective. We conduct extensive experiments to demonstrate the improvements over the original LLMs, and show that our method outperforms other verifiers.

In summary, our contributions are as follows:

- We propose a graph-based verification method, GraphReason, aimed at significantly enhancing the reasoning capabilities of large language models without the need for additional training of LLMs.
- We establish an arithmetic reasoning benchmark using three Math Word Problem datasets to illustrate the fundamental reasoning performance of large language models, and to provide a fair comparison of the performance of various existing verifiers.
- Our experimental results indicate that the method proposed in this paper outperforms other enhancement methods. We also provide an extensive analysis of the limitations and future potential of GraphReason.

2 Related Works

Reasoning of Fine-tuning Models has been extensively studied. It focuses on addressing reasoning tasks using a general sequence-to-sequence approach, enhanced by reasoning-aware pre-training or fine-tuning of language models. (Cobbe et al., 2021a) proposed training a verifier to rank solutions sampled from fine-tuned language models. (Yoran et al., 2022; Wang et al., 2022) suggested equipping language models with reasoning abilities by generating training examples with human-designed templates. (Pi et al., 2022) proposed injecting reasoning capabilities into language models by continually pre-training on program execution data.

Several studies have focused on imbuing PLM with reasoning ability for specific tasks, such as arithmetic reasoning (Cobbe et al., 2021a; Miao et al., 2020; Patel et al., 2021), commonsense reasoning (Talmor et al., 2019), and inductive reasoning (Sinha et al., 2019). For instance, various strategies have been proposed to improve language models' performance on arithmetic reasoning tasks, often referred to as math word problems. (Xie and Sun, 2019) proposed a tree-structured decoder to generate an equation tree, while (Zhang et al., 2020) applied graph convolutional networks to extract relationships

kết thúc chương trình LLM với các trường hợp mẫu, do đó cho phép họ học hỏi và mô phỏng các ví dụ này để cải thiện kết quả. Trong trường hợp suy luận số học, GPT-4 có thể đạt được độ chính xác của 92% trên tập dữ liệu GSM8K sử dụng chuỗi gợi ý suy nghĩ 5 lần (Cobbe và cộng sự, 2021a). Điều này thể hiện mức độ khó khăn mà một trung tâm sáng sủa học sinh phải có khả năng xử lý. Như được mô tả trong Hình 1, điều này minh họa một bước nhiều bước quá trình suy luận số học trong LLM.

Ngoài việc đào tạo thêm LLM và thiết kế nhanh chóng, một số phương pháp đã được đề xuất để nâng cao khả năng lý luận của LLM từ góc độ xác minh đầu ra. ý tưởng chính là để LLM tạo ra lý luận đường dẫn nhiều lần, và sau đó thiết kế một trình xác minh để đánh giá những con đường này và đưa ra kết quả cuối cùng . (Wang và cộng sự, 2023) giới thiệu khái niệm sự tự nhất quán, dựa trên trực giác rằng một vấn đề lý luận phức tạp thường cho phép nhiều quá trình suy nghĩ, tất cả đều dẫn đến một sự chính xác duy nhất câu trả lời. (Li et al., 2023) cũng đề xuất All Roads Dẫn đến Rome, nơi giới thiệu một trình xác minh nhận biết từng bước để phân tích các đường dẫn lý luận không chỉ thông qua toàn bộ con đường, nhưng ở mỗi bước. Tuy nhiên, cả hai phương pháp đều coi mỗi con đường lý luận là một thực thể và không xem xét mối quan hệ tiềm tàng và tương tác giữa các lý luận khác nhau đường dẫn. Một khi các đường dẫn lý luận được phân tách thành các bước, các bước trung gian từ một con đường có thể mang mối quan hệ lý luận với các con đường lý luận khác. Những các phương pháp không nhận ra tất cả các đầu ra LLM cho một được đưa ra đầu vào như một thực thể tập thể, do đó không thể phân tích các mối quan hệ nội bộ của tất cả các đường dẫn ứng viên sâu sắc.

Lấy cảm hứng từ những quan sát này, chúng tôi đề xuất Trình kiểm chứng đồ thị suy luận (GraphReasoning Graph Verifier) trong bài báo này. Chúng tôi giả định rằng các đường dẫn suy luận của một câu hỏi có thể tạo ra các biểu đồ lý luận, trong đó các bước lý luận trung gian tương tự có thể được hợp nhất thành cùng một nút. Với cấu trúc đồ thị, chúng ta có thể mô hình hóa và nắm bắt logic lý luận hiệu quả hơn giữa các bước trung gian từ các lý luận khác nhau đường dẫn. Cụ thể, trước tiên chúng ta xây dựng một lý luận đồ thị dựa trên tất cả các đầu ra từ LLM, và sau đó đào tạo người xác minh để tìm hiểu mối quan hệ giữa cấu trúc đồ thị và câu trả lời cuối cùng. Trong giai đoạn dự đoán, chúng tôi xử lý dữ liệu trong cùng một cách như trong giai đoạn đào tạo và sử dụng trình xác minh để đánh giá từng biểu đồ lý luận. Sau đó, chúng tôi chọn đồ thị lý luận có điểm số cao nhất, sử dụng câu trả lời của nó như là câu trả lời cuối cùng. Để tốt nhất của chúng tôi kiến thức, chúng tôi là những người đầu tiên tiếp cận logic lý luận của LLM từ góc độ đồ thị. Chúng tôi tiến hành các thí nghiệm mở rộng để chứng minh cải tiến so với LLM ban đầu và hiển thị rằng phương pháp của chúng tôi vượt trội hơn các phương pháp xác minh khác.

Tóm lại, những đóng góp của chúng tôi như sau:

- Chúng tôi đề xuất một phương pháp xác minh dựa trên đồ thị, GraphReason, nhằm mục đích nâng cao đáng kể khả năng suy luận của các mô hình ngôn ngữ lớn mà không cần đào tạo thêm Thạc sĩ Luật (LLM).
- Chúng tôi thiết lập một chuẩn mực lý luận số học sử dụng ba tập dữ liệu Bài toán lời văn để minh họa hiệu suất suy luận cơ bản của các mô hình ngôn ngữ lớn và cung cấp sự so sánh công bằng về hiệu suất của nhiều mô hình hiện có người xác minh.
- Kết quả thử nghiệm của chúng tôi chỉ ra rằng phương pháp Phương pháp được đề xuất trong bài báo này vượt trội hơn các phương pháp cải tiến khác. Chúng tôi cũng cung cấp một phân tích sâu rộng về những hạn chế và tiềm năng tương lai của GraphReason.

2 Tác phẩm liên quan

Lý luận của các Mô hình Tinh chỉnh đã được nghiên cứu rộng rãi. Nó tập trung vào việc giải quyết các nhiệm vụ lý luận bằng cách sử dụng phương pháp tiếp cận chuỗi-đến-chuỗi chung , được tăng cường bằng cách đào tạo trước có nhận thức về lý luận. hoặc tinh chỉnh các mô hình ngôn ngữ. (Cobbe và cộng sự, 2021a) đề xuất đào tạo người xác minh để xếp hạng các giải pháp được lấy mẫu từ các mô hình ngôn ngữ được tinh chỉnh. (Yoran và cộng sự, 2022; Wang và cộng sự, 2022) đề xuất trang bị cho các mô hình ngôn ngữ khả năng lập luận bằng tạo ra các ví dụ đào tạo với thiết kế của con người mẫu. (Pi và cộng sự, 2022) đề xuất đưa khả năng suy luận vào các mô hình ngôn ngữ bằng cách liên tục đào tạo trước trên dữ liệu thực thi chương trình.

Một số nghiên cứu đã tập trung vào việc thấm nhuần PLM có khả năng lý luận cho các nhiệm vụ cụ thể, chẳng hạn như suy luận số học (Cobbe và cộng sự, 2021a; Miao và cộng sự, 2020; Patel và cộng sự, 2021), lẽ thường lý luận (Talmor và cộng sự, 2019) và quy nạp lý luận (Sinha và cộng sự, 2019). Ví dụ, nhiều chiến lược khác nhau đã được đề xuất để cải thiện hiệu suất của các mô hình ngôn ngữ về số học nhiệm vụ lý luận, thường được gọi là từ toán học vấn đề. (Xie và Sun, 2019) đã đề xuất một bộ giải mã có cấu trúc cây để tạo ra một phương trình cây, trong khi (Zhang và cộng sự, 2020) áp dụng đồ thị mạng lưới tích chập để trích xuất các mối quan hệ

of quantities in math problems. (Li et al., 2022) used contrastive learning to better learn patterns in math word problems. However, (Valmeekam et al., 2023; Rae et al., 2022) suggested that reasoning, particularly multi-step reasoning, is often a weakness in language models and other NLP models.

Reasoning of Large Language Models has garnered significant attention and demonstrated immense potential. Recent advancements in LLMs suggest that the ability for multi-step reasoning is already embedded within these large-scale models (Kojima et al., 2023; Huang and Chang, 2023), such as PaLM (Chowdhery et al., 2022), GPT-4 (OpenAI, 2023). Therefore, providing an adequate prompt is sufficient to utilize this reasoning ability. For example, the prompting method proposed by (Kojima et al., 2023; Wei et al., 2022), which is based on a chain-of-thought, could aid LLMs in generating text with arithmetic reasoning and common factual knowledge. Following (Wei et al., 2022), experiments on current language models demonstrated that chain-of-thought prompting could enhance the accuracy of solving math problems from 18% to 57%. (Lampinen et al., 2022b) included explanations in the in-context examples and tested the influence of explanations by evaluating the score between *explain-then-predict* and *predict-then-explain*. Moreover, (Zhou et al., 2023) suggested a two-stage prompting strategy, *least-to-most* prompting, which breaks down a complex problem into a series of subproblems and solves them step-by-step. (Li et al., 2023) proposed sampling multiple times from diverse prompts to enhance the variety of responses.

In addition to designing prompts, adopting additional strategies like verifier has contributed to enhancing the performance of reasoning abilities of large language models. For instance, (Wang et al., 2023) proposes *self-consistency*, which involves sampling different reasoning paths from the language model, and then returning the most consistent final answer via majority voting. (Li et al., 2023) used a step-aware voting verifier to enhance the reasoning ability of LLMs from two perspectives. These methods strive to augment the reasoning abilities or yield superior reasoning results without necessitating additional training of LLMs. Our work continues this research direction, with a specific focus on developing a novel graph-based verifier to boost the reasoning capabilities of LLMs.

3 Methodology

3.1 GraphReason Framework

Problem 1 (Reasoning to Solve Problems) *Given a set of n math word problems $\mathbf{Q} = \{Q_1, Q_2, ..., Q_n\}$, where each Q_i is represented by the text description of a single math word problem, the goal of reasoning to solve math word problems is to generate the answers $\mathbf{A} = \{A_1, A_2, ..., A_n\}$ for these problems. Here, each A_i represents the generated text of the corresponding answer. During the process of large language models generating answers, a set of n reasoning paths for solutions $\mathbf{S} = \{S_1, S_2, ..., S_n\}$ is also produced. Each solution S_i is represented as $S_i = \{Q, Step_1, Step_2, ..., Step_l, A\}$, where each $Step_i$ denotes the intermediate steps in the step-by-step solutions.*

We propose GraphReason to verify the solutions generated by LLMs in order to improve the final answer accuracy. This method is a graph-based verification technique that analyzes reasoning paths from generated solutions from a graph perspective. The final answer is obtained without modifying the original LLMs, functioning much like a plugin. As illustrated in Figure 2, there are two steps in the training stage: *Graph Construction* and *Graph Classification*. In the *Graph Construction* step, we obtain the generated solution from LLMs with the specific designed prompt and group them according to their final answers. We split reasoning paths by steps and then merge intermediate steps with identical expression to form reasoning graphs. In the *Graph Classification* step, we classify these reasoning graphs with the additional feature of the sum of scores from the *base verifier* to train the integrated verifier model. In the prediction stage, the candidate solutions are first generated by LLMs. We process them in the same manner as in the training stage, then we use trained verifier to evaluate the scores of each candidate solution. The best solution, denoted by the highest score, is selected as the final predicted answer. We will now provide a detailed introduction to the entire process.

3.2 Prompt Design

To improve the output of Language Models (LLMs) in providing solutions, it is essential to design effective prompts. We incorporate chain-of-thought and in-context learning to enable LLMs to generate step-by-step answers for math word problems. The language models generate output $\mathbf{y}$ based on the

về số lượng trong các bài toán. (Li và cộng sự, 2022) đã sử dụng phương pháp học tương phản để học tốt hơn các mô hình trong các bài toán có lời văn. Tuy nhiên, (Valmeekam và cộng sự, 2023; Rae và cộng sự, 2022) cho rằng lập luận, đặc biệt là lập luận nhiều bước, thường là điểm yếu trong các mô hình ngôn ngữ và các mô hình NLP khác.

Lý luận của các mô hình ngôn ngữ lớn đã thu hút được sự chú ý đáng kể và chứng minh tiềm năng to lớn. Những tiến bộ gần đây trong LLM cho thấy khả năng lý luận nhiều bước đã được nhúng trong các mô hình quy mô lớn này (Kojima và cộng sự, 2023; Huang và Chang, 2023), chẳng hạn như PaLM (Chowdhery và cộng sự, 2022), GPT-4 (OpenAI, 2023). Do đó, việc cung cấp một lời nhắc đầy đủ là đủ để sử dụng khả năng lý luận này. Ví dụ, phương pháp nhắc nhở do (Kojima và cộng sự, 2023; Wei và cộng sự, 2022) đề xuất, dựa trên chuỗi suy nghĩ, có thể hỗ trợ LLM trong việc tạo văn bản bằng lý luận số học và kiến thức thực tế phổ biến. Tiếp theo (Wei và cộng sự, 2022), các thí nghiệm trên các mô hình ngôn ngữ hiện tại đã chứng minh rằng lời nhắc nhở theo chuỗi suy nghĩ có thể nâng cao độ chính xác khi giải các bài toán từ 18% lên 57%. (Lampinen và cộng sự, 2022b) đã đưa các giải thích vào các ví dụ trong ngữ cảnh và kiểm tra ảnh hưởng của các giải thích bằng cách đánh giá điểm số giữa giải thích-rồi-dự đoán và dự đoán-rồi-giải thích. Hơn nữa, (Zhou và cộng sự, 2023) đã đề xuất một chiến lược gợi ý hai giai đoạn, gợi ý từ ít nhất đến nhiều nhất, chia nhỏ một vấn đề phức tạp thành một loạt các bài toán con và giải quyết chúng từng bước. (Li và cộng sự, 2023) đề xuất lấy mẫu nhiều lần từ các gợi ý khác nhau để tăng cường sự đa dạng của các phản hồi.

Ngoài việc thiết kế lời nhắc, việc áp dụng các chiến lược bổ sung như bộ xác minh đã góp phần nâng cao hiệu suất khả năng lập luận của các mô hình ngôn ngữ lớn. Ví dụ, (Wang và cộng sự, 2023) đề xuất tính tự nhất quán, bao gồm việc lấy mẫu các đường dẫn lập luận khác nhau từ mô hình ngôn ngữ, sau đó trả về câu trả lời cuối cùng nhất quán nhất thông qua bỏ phiếu đa số. (Li và cộng sự, 2023) đã sử dụng bộ xác minh bỏ phiếu nhận biết bước để nâng cao khả năng lập luận của LLM từ hai góc độ. Các phương pháp này cố gắng tăng cường khả năng lập luận hoặc mang lại kết quả lập luận vượt trội mà không cần đào tạo thêm cho LLM.

Công trình của chúng tôi tiếp tục theo hướng nghiên cứu này, tập trung cụ thể vào việc phát triển một công cụ xác minh dựa trên đồ thị mới để tăng cường khả năng lập luận của LLM.

3 Phương pháp luận

3.1 Khung GraphReason Bài toán 1

(Lý luận để giải quyết vấn đề)
Cho một tập hợp n bài toán lời văn Q = {Q1, Q2, ..., Qn}, trong đó mỗi Qi được biểu diễn bằng phần mô tả văn bản của một bài toán lời văn duy nhất, mục tiêu của việc lập luận để giải các bài toán lời văn là tạo ra các câu trả lời A = {A1, A2, ..., An} cho các bài toán này.
Ở đây, mỗi Ai biểu diễn văn bản được tạo ra của câu trả lời tương ứng. Trong quá trình tạo câu trả lời của các mô hình ngôn ngữ lớn, một tập hợp n đường dẫn suy luận cho các giải pháp S = {S1, S2, ..., Sn} cũng được tạo ra. Mỗi giải pháp Si được biểu diễn là Si = {Q, Bước 1, Bước 2, ..., Bước 1 , A}, trong đó mỗi Stepi biểu thị các bước trung gian trong các giải pháp từng bước.

Chúng tôi đề xuất GraphReason để kiểm chứng các giải pháp do LLM tạo ra nhằm cải thiện độ chính xác của câu trả lời cuối cùng. Phương pháp này là một kỹ thuật kiểm chứng dựa trên đồ thị , phân tích các đường dẫn suy luận từ các giải pháp được tạo ra từ góc nhìn đồ thị. Câu trả lời cuối cùng thu được mà không cần sửa đổi LLM gốc, hoạt động giống như một plugin.
Như minh họa trong Hình 2, có hai bước trong giai đoạn huấn luyện: Xây dựng Đồ thị và Phân loại Đồ thị. Trong bước Xây dựng Đồ thị, chúng tôi thu thập giải pháp được tạo ra từ các LLM với lời nhắc được thiết kế cụ thể và nhóm chúng theo kết quả cuối cùng. Chúng tôi chia các đường dẫn suy luận thành các bước và sau đó hợp nhất các bước trung gian có biểu thức giống hệt nhau để tạo thành các đồ thị suy luận. Trong bước Phân loại Đồ thị, chúng tôi phân loại các đồ thị suy luận này với tính năng bổ sung là tổng điểm từ bộ kiểm định cơ sở để huấn luyện mô hình kiểm định tích hợp. Trong giai đoạn dự đoán, các giải pháp ứng viên đầu tiên được tạo ra bởi các LLM.

Chúng tôi xử lý chúng theo cùng cách như trong giai đoạn huấn luyện, sau đó sử dụng bộ kiểm chứng đã được huấn luyện để đánh giá điểm của từng giải pháp ứng viên. Giải pháp tốt nhất, được biểu thị bằng điểm cao nhất, sẽ được chọn làm đáp án dự đoán cuối cùng. Bây giờ, chúng tôi sẽ giới thiệu chi tiết về toàn bộ quy trình.

3.2 Thiết kế nhanh chóng

Để cải thiện kết quả đầu ra của Mô hình Ngôn ngữ (LLM) trong việc cung cấp giải pháp, điều cần thiết là phải thiết kế các gợi ý hiệu quả. Chúng tôi kết hợp phương pháp chuỗi suy nghĩ và học tập theo ngữ cảnh để cho phép LLM tạo ra các câu trả lời từng bước cho các bài toán có lời văn. Các mô hình ngôn ngữ tạo ra kết quả đầu ra dựa trên

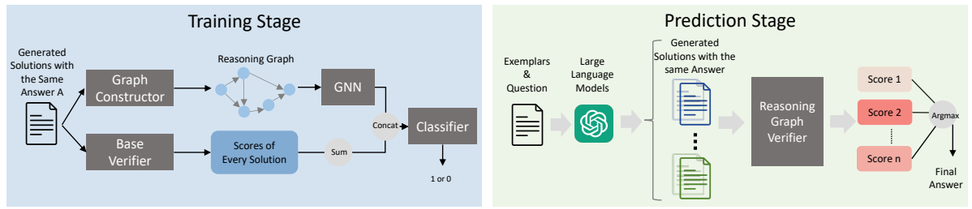


Figure 2: The framework of GraphReason. In the training stage, GraphReason processes generated solutions from LLMs to construct reasoning graphs, and then trains a verifier to judge them according to graph classification. In the prediction stage, GraphReason evaluates candidate solutions to assign a score, and selects the solution with the highest score as the final answer.

input $\mathbf{x}$ using the following equation:

$$p(\mathbf{y}|C,\mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} p_{LM}(\mathbf{y}_t|C,\mathbf{x},\mathbf{y}<t), \quad (1)$$

where, C represents the input provided to the LLMs prior to the current math word problem's question. C is a concatenation of k exemplars, denoted as:

$$C = [(Q_1,S_1,A_1);(Q_2,S_2,A_2),...;(Q_k,S_k,A_k)], \quad (2)$$

where, Q_i represents the question, S_i represents the intermediate steps of the solution, and A_i represents the answer. We set k to five in this study, resulting in a prompt that consists of five question-answer pairs sampled from the training split of a math word problem dataset. Therefore, the prompt can be denoted as:

$$\textbf{Prompt} = [C;Q], \quad (3)$$

where Q represents the question of the current math word problem.

Using a greedy decoding approach to sample one output from LLMs may not be robust. It can lead to instability and occasional errors. To address this, (Wang et al., 2023) propose the concept of *self-consistency*. This approach involves sampling different reasoning paths from the language model and then selecting the most consistent final answer through majority voting. Instead of using greedy decoding to sample only once and verify, they utilize sampling decoding to sample N_1 times. We also follow the idea presented by (Li et al., 2023) in their work named *All Roads Lead to Rome*. This approach involves generating N_2 diverse prompts for LLMs to produce multiple outputs. By employing multiple sampling decodes on diverse prompts, we can obtain generated solutions from different sources. Specifically, we obtain $N = N_1 \times N_2$ diverse reasoning paths for each question. In our main experiments, we set $N_1 = 10$ and $N_2 = 3$. These solutions will be further processed and verified using our designed verifier.

3.3 Reasoning Graph Construction

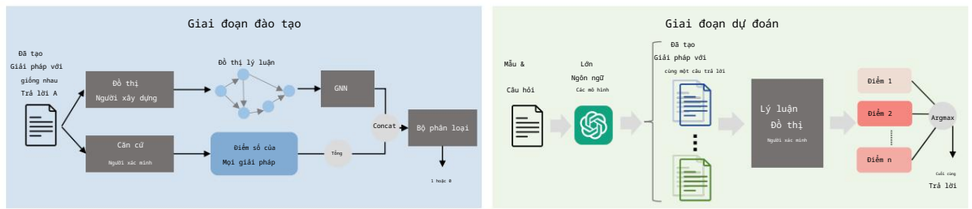
After generating multiple solutions for a question, it becomes necessary to construct reasoning graphs based on the reasoning paths taken by these solutions.

As shown in Figure 3, we begin by grouping all the generated solutions for a particular question according to their final answer. Since these solutions originate from the same question, their reasoning paths will share the same starting point. Similarly, solutions with the same final answer will have the same endpoint, as their reasoning paths converge. Therefore, a group of generated solutions with the same final answer can form a reasoning graph with a uniform start node (question node) and end node (answer node). We define this division process as follows:

$$\mathbf{S} = \{S_{A_1}, S_{A_2}, ..., S_{A_n}\}, \quad (4)$$

where $\mathbf{S}$ represents the set of generated solutions for a question, and $S_{A_i} = \{S_1, S_2, ..., S_m\}$ is the subset of generated solutions that all have the same final answer A_i.

For each subset of generated solutions S_{A_i}, we construct a reasoning graph. This construction is motivated by the understanding that each step in the reasoning path of a generated solution does not exist in isolation from the other solutions. The steps from one solution's reasoning path can impact the steps from another solution, enhancing the overall reasoning process. We utilize the graph structure to model and capture these relationships between steps from different solutions. As the different reasoning paths can benefit each other, we construct



Hình 2: Khung làm việc của GraphReason. Ở giai đoạn huấn luyện, GraphReason xử lý các giải pháp được tạo ra từ các LLM để xây dựng biểu đồ suy luận, sau đó huấn luyện một trình kiểm chứng để đánh giá chúng theo phân loại biểu đồ. Ở giai đoạn dự đoán, GraphReason đánh giá các giải pháp ứng viên để chấm điểm và chọn giải pháp có điểm cao nhất làm kết quả cuối cùng.

nhập x bằng phương trình sau:

$$p(y|C,\ x) = \prod_{t=1}^{|y|} pLM\ (yt\ |C,\ x,\ y < t),\ (1)$$

trong đó, C biểu thị thông tin đầu vào được cung cấp cho LLM trước câu hỏi của bài toán bằng lời hiện tại. C là sự kết hợp của k ví dụ, được biểu thị như sau:

C = [(Q1, S1, A1); (Q2, S2, A2), ...; (Qk, Sk, Ak)], (2) trong đó,

Qi biểu thị câu hỏi, Si biểu thị các bước trung gian của giải pháp, và Ai biểu thị câu trả lời. Trong nghiên cứu này, chúng tôi đặt k bằng năm, dẫn đến một lời nhắc bao gồm năm cặp câu hỏi-câu trả lời được lấy mẫu từ phần phân tách huấn luyện của một tập dữ liệu bài toán lời văn. Do đó, lời nhắc có thể được ký hiệu là:

Yêu cầu = [C; Q], (3)

trong đó Q biểu thị câu hỏi của bài toán bằng lời hiện tại.

Việc sử dụng phương pháp giải mã tham lam để lấy mẫu một đầu ra từ LLM có thể không mạnh mẽ. Nó có thể dẫn đến sự bất ổn định và đôi khi xảy ra lỗi. Để giải quyết vấn đề này, (Wang và cộng sự, 2023) đề xuất khái niệm tự nhất quán. Phương pháp này bao gồm việc lấy mẫu các đường dẫn suy luận khác nhau từ mô hình ngôn ngữ và sau đó chọn câu trả lời cuối cùng nhất quán nhất thông qua bỏ phiếu đa số. Thay vì sử dụng giải mã tham lam để chỉ lấy mẫu một lần và xác minh, họ sử dụng giải mã lấy mẫu để lấy mẫu N1 lần. Chúng tôi cũng tuân theo ý tưởng được trình bày bởi (Li và cộng sự, 2023) trong công trình của họ có tên "Mọi con đường đều dẫn đến Rome". Phương pháp này bao gồm việc tạo ra N2 lời nhắc đa dạng cho LLM để tạo ra nhiều đầu ra. Bằng cách sử dụng nhiều giải mã lấy mẫu trên các lời nhắc đa dạng, chúng ta có thể thu được các giải pháp được tạo ra từ các nguồn khác nhau. Cụ thể, chúng ta thu được N = N1 × N2 nhiều hướng suy luận khác nhau cho mỗi câu hỏi. Trong các thí nghiệm chính, chúng tôi đặt N1 = 10 và N2 = 3. Các giải pháp này sẽ được xử lý và xác minh thêm bằng trình xác minh do chúng tôi thiết kế.

3.3 Xây dựng đồ thị suy luận

Sau khi tạo ra nhiều giải pháp cho một câu hỏi, cần phải xây dựng biểu đồ lý luận dựa trên các đường dẫn lý luận của các giải pháp này .

Như thể hiện trong Hình 3, chúng ta bắt đầu bằng cách nhóm tất cả các giải pháp được tạo ra cho một câu hỏi cụ thể theo câu trả lời cuối cùng của chúng. Vì các giải pháp này bắt nguồn từ cùng một câu hỏi, nên các đường suy luận của chúng sẽ có cùng một điểm xuất phát. Tương tự, các giải pháp có cùng câu trả lời cuối cùng sẽ có cùng điểm kết thúc, vì các đường suy luận của chúng hội tụ.

Do đó, một nhóm các giải pháp được tạo ra với cùng một kết quả cuối cùng có thể tạo thành một đồ thị suy luận với một nút bắt đầu (nút câu hỏi) và nút kết thúc (nút câu trả lời) đồng nhất. Chúng tôi định nghĩa quy trình phân chia này như sau:

S = {SA1 , SA2 , ..., SAn }, (4)

trong đó S biểu thị tập hợp các giải pháp được tạo ra cho một câu hỏi và SAi = {S1, S2, ..., Sm} là tập hợp con các giải pháp được tạo ra có cùng câu trả lời cuối cùng Ai .

Với mỗi tập con các giải pháp được tạo ra SAi , hãy chúng tôi xây dựng một đồ thị suy luận. Cấu trúc này được thúc đẩy bởi sự hiểu biết rằng mỗi bước trong đường dẫn suy luận của một giải pháp được tạo ra không tồn tại tách biệt với các giải pháp khác. Các bước từ đường dẫn suy luận của một giải pháp có thể tác động đến các bước từ một giải pháp khác, nâng cao quá trình suy luận tổng thể. Chúng tôi sử dụng cấu trúc đồ thị để mô hình hóa và nắm bắt các mối quan hệ này giữa các bước từ các giải pháp khác nhau. Vì các đường dẫn suy luận khác nhau có thể hỗ trợ lẫn nhau, chúng tôi xây dựng

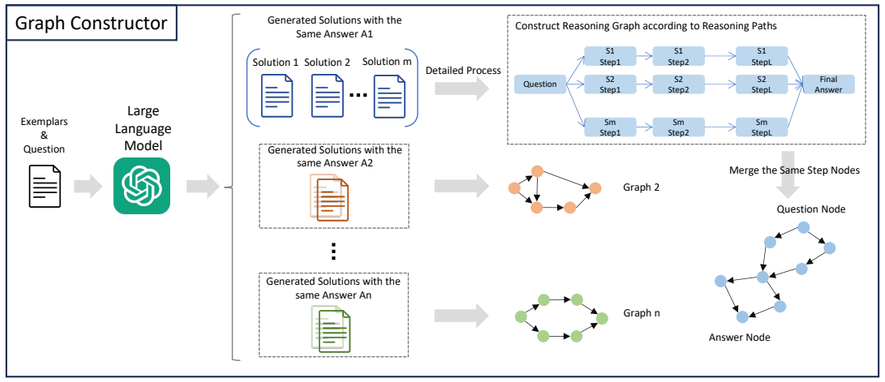


Figure 3: The graph constructor in GraphReason. We detail the process of transforming 'Generated Solutions with the Same Answer A1' to 'Graph 1'.

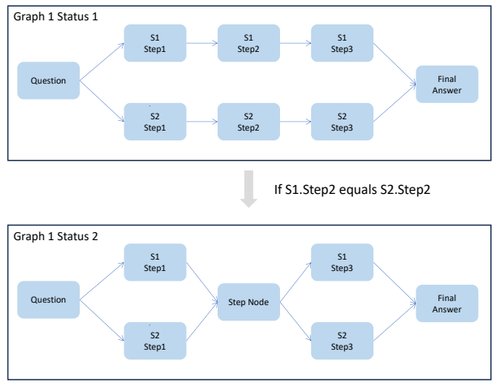


Figure 4: The process of reasoning graph construction. The primary operation here is the merging of identical intermediate steps in reasoning paths into a single graph node.

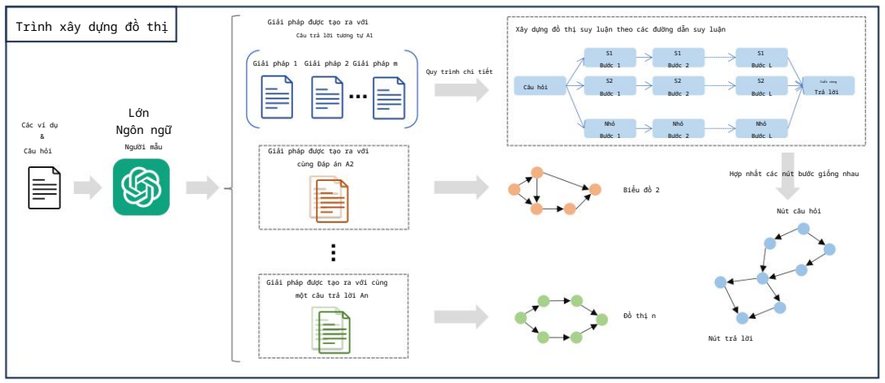
a reasoning graph to link these paths together. As shown in Figure 4, the primary operation here is the merging of identical intermediate nodes in reasoning paths into a single graph node. We first compare the reasoning steps from any two solution reasoning paths. If they have the same intermediate steps of arithmetic expression, we merge them into the same node, and if they differ, we do not. For reasoning math word problems here, we define reasoning steps as the current arithmetic expression without other language text in the current reasoning step for clarity. It can help us simplify construction of reasoning graphs in the reasoning task. The detailed algorithm for constructing a reasoning graph is shown in Algorithm 1.

The generated solutions, divided by their final answers $\{S_{A_1}, S_{A_2}, ..., S_{A_n}\}$, can be transformed into n reasoning graphs of generated solutions $\{G_{A_1}, G_{A_2}, ..., G_{A_n}\}$.

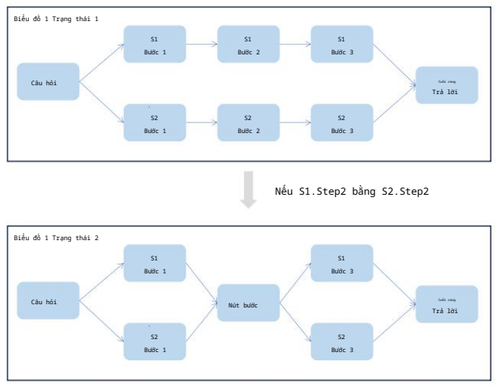
Regarding the node features in the graph, we select the score from the *Base Verifier* and the node degree. We believe the score from the *Base Verifier* encapsulates the semantic information of solutions, and the node degree contains information about the graph structure. The *Base Verifier* is trained independently from the whole framework. It is designed to judge whether a single reasoning path of one solution is correct, which is a binary text classification task. After training, it can be used to verify any single solution and assign a $score \in (0, 1)$ to evaluate the likelihood of the solution being correct, where $score = 0.99$ suggests a 99% probability of the solution being correct. We use the score from the *Base Verifier* to better incorporate solution semantic information because, according to our experiments, it is challenging to model semantic information while modeling reasoning logic information. The score of a step is the same as its solution score. Therefore, for one step node V, it has many scores $\{score^a, score^b, ..., score^c\}$ from different solutions. The feature of one node V_i in the graph is then concatenated by the selected feature, which can be represented as:

$$\mathbf{V} = [score_i^{mean}, score_i^{max}, score_i^{min}, score_i^{num}, in\_degree_i], \quad (5)$$

where $\mathbf{V} \in \mathbb{R}^5$, $score_i^{mean}$ is the mean of all scores of one step V_i, $score_i^{max}$ is the maximum score, $score_i^{min}$ is the minimum score, $score_i^{num}$ is the number of scores, and in_degree_i is the in-degree of the step node V_i.



Hình 3: Trình xây dựng đồ thị trong GraphReason. Chúng tôi trình bày chi tiết quá trình chuyển đổi 'Các giải pháp được tạo ra có cùng đáp án A1' thành 'Đồ thị 1'.



Hình 4: Quá trình xây dựng đồ thị suy luận.

Hoạt động chính ở đây là hợp nhất các bước trung gian giống hệt nhau trong các đường dẫn suy luận thành một nút đồ thị duy nhất.

một đồ thị suy luận để liên kết các đường dẫn này lại với nhau. Như thể hiện trong Hình 4, thao tác chính ở đây là hợp nhất các nút trung gian giống hệt nhau trong các đường dẫn suy luận thành một nút đồ thị duy nhất. Trước tiên, chúng ta so sánh các bước suy luận từ bất kỳ hai đường dẫn suy luận nào. Nếu chúng có cùng các bước trung gian của biểu thức số học, chúng ta sẽ hợp nhất chúng vào cùng một nút, và nếu chúng khác nhau, chúng ta sẽ không hợp nhất. Đối với các bài toán lời văn suy luận ở đây, chúng tôi định nghĩa các bước suy luận là biểu thức số học hiện tại mà không có văn bản ngôn ngữ khác trong bước suy luận hiện tại để rõ ràng hơn. Điều này có thể giúp chúng ta đơn giản hóa việc xây dựng đồ thị suy luận trong nhiệm vụ suy luận. Thuật toán chi tiết để xây dựng đồ thị suy luận được trình bày trong Thuật toán 1.

Các giải pháp được tạo ra, chia cho giá trị cuối cùng của chúng

câu trả lời {SA1 , SA2 , ..., SAn }, có thể được chuyển đổi thành n đồ thị suy luận của các giải pháp được tạo ra {GA1 , GA2 , ..., GAn }.

Liên quan đến các đặc điểm nút trong đồ thị, chúng tôi chọn điểm từ Bộ xác minh cơ sở và bậc nút. Chúng tôi tin rằng điểm từ Bộ xác minh cơ sở bao hàm thông tin ngữ nghĩa của các giải pháp, và bậc nút chứa thông tin về cấu trúc đồ thị. Bộ xác minh cơ sở được đào tạo độc lập với toàn bộ khung. Nó được thiết kế để đánh giá xem một đường dẫn suy luận duy nhất của một giải pháp có đúng hay không, đây là một nhiệm vụ phân loại văn bản nhị phân. Sau khi đào tạo, nó có thể được sử dụng để xác minh bất kỳ giải pháp đơn lẻ nào và gán một điểm (0, 1) để đánh giá khả năng giải pháp đó là đúng, trong đó điểm = 0,99 cho thấy xác suất 99% giải pháp đó là đúng. Chúng tôi sử dụng điểm từ Bộ xác minh cơ sở để kết hợp tốt hơn thông tin ngữ nghĩa của giải pháp vì, theo các thí nghiệm của chúng tôi, việc mô hình hóa thông tin ngữ nghĩa trong khi mô hình hóa thông tin logic suy luận là một thách thức . Điểm của một bước giống với điểm giải pháp của nó. Do đó, đối với một nút V bước, điểm b có nhiều điểm {scorea , , ..., điểm c} từ các giải pháp khác nhau. Đặc điểm của một nút Vi trong đồ thị sau đó được nối với đặc điểm đã chọn , có thể được biểu diễn như sau:

, Nó

$$V = [\text{điểm trung bình}, \text{điểm số}_{\text{max}}, \text{điểm số}_{\text{}}, \text{scorenum}_{\text{}}, \text{in\_degreei}], \quad (5)$$

trong đó V R ^{5}, scoremean là giá trị trung bình của tất cả các điểm số i của một bước Vi , scoremax là điểm tối đa, scoremin là điểm tối thiểu, scorenum là số điểm và in_degreei là bậc vào của nút bước Vi .

Algorithm 1 Reasoning graph construction algorithm

Input: generated solutions S_{A_i} which have the same final answers

Output: a reasoning graph G_{A_i}

$node\_num \leftarrow 0$
$node2id \leftarrow dict()$
$edges \leftarrow list()$
for each $reason\_path$ in S_{A_i} **do**
  for each $step$ in $reason\_path$ **do**
    if $step$ not in $node2id.keys()$ **then**
      $node2id[step] \leftarrow node\_num$
      $node\_num \leftarrow node\_num + 1$
    end if
  end for
end for
for each $reason\_path$ in S_{A_i} **do**
  for each $step$ in $reason\_path$ **do**
    $start\_node \leftarrow node2id[last\_step]$
    $end\_node \leftarrow node2id[step]$
    if $(start\_node, end\_node)$ not in $edges$ **then**
      $edges.add((start\_node, end\_node))$
    end if
    $last\_step \leftarrow step$
  end for
end for
$G_{A_i} \leftarrow graph(node2id, edges)$

In this way, we can obtain multiple reasoning graphs to represent all generated solutions from LLMs for a single math word problem question.

3.4 Verifier Design

Our designed verifier `GraphReason`,is used to evaluate the answer of a generated solutions group, which is also represented as a reasoning graph. This verifier has two inputs: the graph and the sum of solution scores. We employ the Graph Isomorphism Network (GIN) (Xu et al., 2019) to perform node feature propagation, thereby encoding the information from the reasoning graphs we obtained. The node feature is propagated and aggregated as follows:

$$h_v^{(k)} = MLP^{(k)}\big((1+\varepsilon^{(k)}) \cdot h_v^{(k-1)} + \sum\nolimits_{u \in N(v)} h_u^{(k-1)}\big), \tag{6}$$

where $h_v^{(k)}$ represents the state of node v after the k^{th} update. $MLP^{(k)}$ refers to a multi-layer perceptron in the k^{th} layer. $N(v)$ denotes all the neighbors of node v and ε is a learnable parameter. Then, we perform a sum readout to obtain the representation of the reasoning graph:

$$h_G = \sum_{v \in G} h_v^{(k)}, \tag{7}$$

where $h_G \in \mathbb{R}^5$. We set k to 3, signifying the application of three layers of GIN. Concurrently, the sum of the scores of solutions with the same final answer, A_i, denoted as $score_{A_i}$, is represented as follows:

$$score_A = \sum_{i \in S_A} score_i. \tag{8}$$

Then a reasoning graph can then be represented as:

$$\mathbf{G} = [h_G, score_A], \tag{9}$$

where $\mathbf{G} \in \mathbb{R}^6$.

The target label of the graph $\mathbf{y} \in \{0, 1\}$ indicates whether the final answer matches the correct final answer. We compute the loss and train the verifier model by:

$$\mathcal{L} = \sum_{i=1}^{n} \mathcal{L}_{BCE}(label_i, f(\mathbf{G}_i)), \tag{10}$$

where i represents the number of solution subset among all n subsets after grouping solutions. The corresponding reasoning graph for this subset is denoted by $\mathbf{G}_i$, and $f()$ is a linear classifier.

3.5 Answer Verification

During the prediction stage, all generated solutions are processed in the same way as in the training stage. The trained verifier is then used to evaluate the scores of each reasoning graph, each of which represents a group of solutions that yield the same final answer. The final answer associated with the highest score is selected as our final predicted answer:

$$\hat{\mathbf{y}} = \mathbf{Answer}[\arg\max_i score_i], \tag{11}$$

where $score_i$ denotes the score of the reasoning graph $\mathbf{G}_i$, as determined by our verifier. **Answer** represents the list of all candidate final answers. By predicting the number of the optimal reasoning graph, we can determine the final predicted result of the current reasoning task.

Thuật toán 1 Thuật toán xây dựng đồ thị suy luận

Đầu vào: các giải pháp được tạo ra SAi có cùng câu trả lời cuối cùng

Đầu ra: một đồ thị lý luận GAi

node_num 0 node2id
dict() cạnh list()
đối với mỗi reason_path
trong SAi thực hiện đối với mỗi bước trong
reason_path thực hiện:
nếu bước không có trong node2id.keys()
thì node2id[bước] node_num
node_num node_num + 1 kết thúc nếu

kết thúc cho
kết thúc cho
đối với mỗi reason_path trong SAi thực
hiện đối với mỗi bước trong reason_path thực
hiện start_node node2id[last_step]
end_node node2id[step] nếu
(start_node, end_node) không nằm trong edges thì

edges.add((start_node, end_node)) end nếu

last_step step 20:
end for
kết thúc cho
GAi đồ thị (node2id, cạnh)

Theo cách này, chúng ta có thể thu được nhiều biểu đồ lý luận để biểu diễn tất cả các giải pháp được tạo ra từ LLM cho một câu hỏi toán học bằng lời.

3.4 Thiết kế xác minh

Công cụ xác minh GraphReason do chúng tôi thiết kế được sử dụng để đánh giá câu trả lời của nhóm giải pháp được tạo ra, cũng được biểu diễn dưới dạng biểu đồ lý luận. Bộ xác minh này có hai đầu vào: đồ thị và tổng điểm của các nghiệm. Chúng tôi sử dụng Mạng Đồng hình Đồ thị (GIN) (Xu và cộng sự, 2019) để thực hiện lan truyền đặc trưng nút, từ đó mã hóa thông tin từ đồ thị suy luận mà chúng tôi thu được. Tính năng nút được truyền bá và tổng hợp như sau:

$$h_v^{(k)} = MLP^{(k)}\big((1+\varepsilon^{(k)}) \cdot h_v^{(k-1)} + \sum\nolimits_{u \in N(v)} h_u^{(k-1)}\big), \tag{6}$$

trong đó h v biểu diễn trạng thái của nút v sau khi th k cập nhật. MLP(k) biểu diễn một lớp nhận thức đa lớp . th tron trong k N(v) biểu diễn tất cả các lân cận của nút v và ε là một tham số có thể học được. Khi đó, chúng tôi thực hiện phép đọc tổng để có được biểu diễn của đồ thị lý luận:

$$h_G = \sum_{v \in G} h_v^{(k)} \tag{7}$$

5 trong đó hG R . Chúng tôi đặt k bằng 3, biểu thị việc áp dụng ba lớp GIN. Đồng thời, tổng điểm của các nghiệm có cùng đáp án cuối cùng, Ai , ký hiệu là scoreAi , được biểu diễn như sau:

$$\text{điểmA} = \sum_{i \in SA} \text{điểm số}. \tag{8}$$

Sau đó, đồ thị lý luận có thể được biểu diễn như sau:

$$G = [hG, \text{điểm A}], \tag{9}$$

6 trong đó G R .

Nhãn đích của đồ thị y {0, 1} cho biết liệu đáp án cuối cùng có khớp với đáp án cuối cùng chính xác hay không. Chúng tôi tính toán độ mất mát và huấn luyện mô hình kiểm chứng bằng cách:

$$L = \sum_{i=1}^{N} LBCE(\text{labeli}, f(Gi)), \tag{10}$$

trong đó i biểu thị số tập con nghiệm trong tổng số n tập con sau khi nhóm các nghiệm. Đồ thị suy luận tương ứng cho tập con này là và f() là một bộ phân loại tuyến tính, ký hiệu là Gi ,

3.5 Xác minh câu trả lời

Trong giai đoạn dự đoán, tất cả các giải pháp được tạo ra đều được xử lý theo cùng một cách như trong giai đoạn huấn luyện. Bộ kiểm chứng đã được huấn luyện sau đó được sử dụng để đánh giá điểm của từng biểu đồ lý luận, mỗi biểu đồ đại diện cho một nhóm các giải pháp cho ra cùng một kết quả cuối cùng. Câu trả lời cuối cùng có điểm cao nhất sẽ được chọn làm kết quả dự đoán cuối cùng của chúng tôi.
trả lời:

$$\hat{y} = \text{Answer}[\arg\max \text{scorei}], \tag{11}$$

trong đó scorei biểu thị điểm của đồ thị lý luận Gi , được xác định bởi trình xác minh của chúng tôi. Answer biểu thị danh sách tất cả các câu trả lời cuối cùng của ứng viên.
Bằng cách dự đoán số lượng đồ thị lý luận tối ưu, chúng ta có thể xác định kết quả dự đoán cuối cùng của nhiệm vụ lý luận hiện tại.

	GSM8K	SVAMP	ASDiv-a	StrategyQA
Fine-tuning SOTA	57^a	57.4^b	75.3^c	73.9^d
9–12 year olds	60	-	-	-
gpt-3.5-turbo:				
Greedy Decode	72.7	78.7	93.0	65.0
Self-Consistency (Voting)	82.3	82.9	95.6	66.0
Verifier	66.9	73.1	92.8	69.3
Voting Verifier	85.4	84.8	96.9	70.7
DIVERSE (Step-aware Voting Verifier)	85.0	85.1	96.8	66.9
Reasoning Graph Verifier (Ours)	**85.7**	**85.4**	**97.0**	**71.2**

Table 1: The comparison experiment results of GraphReason, other verifiers, and other baselines. We primarily compare GraphReason with other verifiers which are all based on the same generated solutions from *gpt-3.5-turbo*

4 Experiments

In this section, we conducted extensive experiments to demonstrate the performance of GraphReason, along with a more in-depth analysis. Universally, we reproduced all types of verifiers to report their results based on the same generated solutions. Our experiments are conducted in two settings: Arithmetic Reasoning and Commonsense Reasoning. We ensured a fair comparison by setting the same random seed, using the same hardware environment, and applying similar hyperparameters. We used accuracy as the metric to evaluate the ability of solving math word problems, which determines whether the final answer is correct or not.

4.1 Training Details

For LLMs sampling, we use *gpt-3.5-turbo* as our base LLMs and set the temperature t to 1. All verifiers use the same LLMs' output. Regarding verifier training, we fine-tune on *bert-base-uncased* (Devlin et al., 2019). We employ the AdamW optimizer (Loshchilov and Hutter, 2019) to optimize the model parameters during training. We apply differential learning rates, setting the learning rate of the final linear classifier to 4e-2, while the other graph neural network layers are set to 4e-3. The activation layer between them is ReLU (Agarap, 2019). The batch size in each training step is set to 2. The batch size is small because the verifier needs to verify multiple reasoning graphs for a single question.

To ensure a fair comparison between the Voting Verifier, Simple Verifier, and GraphReason, we use the same trained base verifier for all three approaches.

The details of the datasets and baselines are provided in Appendix A and Appendix B, respectively.

4.2 Main Results

We present the main results in Table 1. As can be seen from the table, GraphReason significantly enhances the original *gpt-3.5-turbo*'s reasoning abilities across all three datasets, for instance, improving accuracy by 13.0% (72.7% $\rightarrow$ 85.7%) on GSM8K. It is also evident that our method surpasses other verifier methods with the same output from LLMs and achieves the state-of-the-art on all three datasets.

Additionally, the Step-aware Voting Verifier improves upon the Voting Verifier by recognizing that not all steps in an incorrect reasoning path are equally erroneous, and some steps may still be useful for reasoning. We believe this hypothesis is overly simplistic and cannot describe complex logical relationships among steps. According to Table 1, it leads to some metric decline, and the same finding also observed in the original paper. Furthermore, it does not perform well in the StrategyQA task, because there are no gold reasoning paths for the training of this commonsense reasoning task. In this task, the reasoning paths are generated and pseudo, indicating a requirement for gold labels at each step of the reasoning process. However, our paper consistently improves upon the Voting Verifier by considering complex relationship between different reasoning paths through reasoning graphs. We enhance the previous method, which did not consider relations in steps between different solutions, by 0.3% (85.4% $\rightarrow$ 85.7%), 0.3% (85.1% $\rightarrow$ 85.4%), 0.1% (96.9% $\rightarrow$ 97.0%), and 0.5% (70.7% $\rightarrow$ 71.2%) across the four datasets.

Moreover, GraphReason yields only a slight improvement in performance on ASDiv-a, and the results are nearly identical. One reason for this is that the math word problems from ASDiv-a are sim-

	GSM8K	SVAMP	ASDiv-a	Chiến lượcQA
Tinh chỉnh SOTA dành	57a	57,4 [b]	75,3 [c]	73,9 [ww]
cho trẻ em từ 9–12	60	-	-	-
tuổi gpt-3.5-turbo:				
Tham lam giải mã	72,7	78,7	93,0	65.0
sự tự nhất quán (Bỏ phiếu)	82,3	82,9	95,6	66,0
Trình	66,9	73,1	92,8	69,3
xác minh bỏ phiếu	85,4	84,8	96,9	70,7
DIVERSE (Trình xác minh bỏ phiếu nhận biết bước)	85,0	85,1	96,8	66,9
Reasoning Graph Verifier (của chúng tôi)	85,7	85,4	97,0	71,2

Bảng 1: Kết quả thử nghiệm so sánh của GraphReason, các trình xác minh khác và các đường cơ sở khác. Chúng tôi chủ yếu so sánh GraphReason với các trình xác minh khác đều dựa trên cùng một giải pháp được tạo từ gpt-3.5-turbo

4 Thí nghiệm

Trong phần này, chúng tôi đã tiến hành các thí nghiệm mở rộng để chứng minh hiệu suất của GraphReason, cùng với một phân tích sâu hơn. Trên toàn thế giới, chúng tôi đã sao chép tất cả các loại trình xác minh để báo cáo kết quả dựa trên các giải pháp được tạo ra giống nhau. Của chúng tôi các thí nghiệm được tiến hành trong hai bối cảnh: Lý luận số học và Lý luận thông thường.

Chúng tôi đảm bảo một sự so sánh công bằng bằng cách thiết lập cùng một hạt giống ngẫu nhiên, sử dụng cùng một môi trường phần cứng và áp dụng các siêu tham số tương tự. Chúng tôi sử dụng độ chính xác làm thước đo để đánh giá khả năng của việc giải các bài toán bằng lời, điều này xác định câu trả lời cuối cùng có đúng hay không.

4.1 Chi tiết đào tạo

Đối với việc lấy mẫu LLM, chúng tôi sử dụng gpt-3.5-turbo làm LLM cơ sở và đặt nhiệt độ t thành 1. Tất cả người xác minh sử dụng cùng đầu ra của LLM. Về đào tạo người xác minh, chúng tôi tinh chỉnh trên bert-base-uncased (Devlin và cộng sự, 2019). Chúng tôi sử dụng bộ tối ưu hóa AdamW (Loshchilov và Hutter, 2019) để tối ưu hóa các tham số mô hình trong quá trình đào tạo. Chúng tôi áp dụng tỷ lệ học tập khác biệt, thiết lập tỷ lệ học tập của bộ phân loại tuyến tính cuối cùng thành 4e-2, trong khi bộ kia các lớp mạng nơ-ron đồ thị được đặt thành 4e-3. lớp kích hoạt giữa chúng là ReLU (Agarap, 2019). Kích thước lô trong mỗi bước đào tạo được thiết lập đến 2. Kích thước lô nhỏ vì trình xác minh cần xác minh nhiều biểu đồ lý luận cho một câu hỏi duy nhất.

Để đảm bảo sự so sánh công bằng giữa Vot- ing Verifier, Simple Verifier và GraphReason, chúng tôi sử dụng cùng một trình xác minh cơ sở được đào tạo cho cả ba phương pháp .

Chi tiết về các tập dữ liệu và đường cơ sở được cung cấp lần lượt trong Phụ lục A và Phụ lục B.

4.2 Kết quả chính

Chúng tôi trình bày các kết quả chính trong Bảng 1. Như có thể được nhìn thấy từ bảng, GraphReason đáng kể cải thiện lý luận ban đầu của gpt-3.5-turbo khả năng trên cả ba tập dữ liệu, ví dụ, cải thiện độ chính xác lên 13,0% (72,7% 85,7%) trên GSM8K. Cũng rõ ràng là phương pháp của chúng tôi vượt trội hơn các phương pháp xác minh khác có cùng đầu ra từ LLM và đạt được trình độ tiên tiến nhất về tất cả ba tập dữ liệu.

Ngoài ra, Bộ xác minh bỏ phiếu nhận biết bước cải tiến Bộ xác minh bỏ phiếu bằng cách nhận ra rằng không phải tất cả các bước đều nằm trong một con đường lý luận không chính xác đều sai lầm như nhau và một số bước vẫn có thể hữu ích cho việc lập luận. Chúng tôi tin rằng giả thuyết này quá đơn giản và không thể diễn tả được sự phức tạp mối quan hệ logic giữa các bước. Theo Bảng 1, nó dẫn đến một số suy giảm số liệu và tương tự phát hiện này cũng được quan sát thấy trong bài báo gốc. Hơn nữa, nó không hoạt động tốt trong StrategyQA nhiệm vụ, vì không có con đường lý luận vàng nào cho việc đào tạo nhiệm vụ lý luận thông thường này. Trong nhiệm vụ này, các đường dẫn lý luận được tạo ra và giả, chỉ ra yêu cầu đối với nhãn vàng tại từng bước của quá trình lý luận. Tuy nhiên, giấy liên tục cải thiện Trình xác minh bỏ phiếu bằng cách xem xét mối quan hệ phức tạp giữa các con đường lý luận khác nhau thông qua biểu đồ lý luận. Chúng tôi cải tiến phương pháp trước đó, mà không xem xét mối quan hệ theo các bước giữa các giải pháp khác nhau , bằng 0,3% (85,4% 85,7%), 0,3% (85,1% 85,4%), 0,1% (96,9% 97,0%) và 0,5% (70,7% 71,2%) trên cả bốn tập dữ liệu.

Hơn nữa, GraphReason chỉ mang lại một cải tiến nhỏ về hiệu suất trên ASDiv-a và kết quả gần như giống hệt nhau. Một lý do cho điều này là rằng các bài toán đố từ ASDiv-a là tương tự-

	GSM8K	▽	SVAMP	▽	ASDiv-a	▽
Reasoning Graph Verifier (Ours)	**85.7**	-	**85.4**	-	**97.0**	-
w/o solution semantic from base verifier	81.2	-4.5	83.1	-2.3	94.3	-2.7
w/o solution scores sum	82.8	-2.9	83.2	-2.2	95.6	-1.4
w/o reasoning graphs	85.4	-0.3	84.8	-0.6	96.9	-0.1

Table 2: The ablation experiment results of GraphReason. Missing each component leads to a decline in the final result.

pler compared to those in the other two datasets, based on our observations. In most cases, these problems do not require complex reasoning from a graph perspective to generate a satisfactory answer. It demonstrates that our method is particularly well-suited for such situations. We believe that GraphReason can offer more substantial improvements in the more complex scenario.

4.3 Ablation Study

We conducted an ablation study to evaluate the impact of each component on the overall performance of our method. Table 2 presents the results of this study, highlighting how these modules contribute to the improvement of the base model in distinct ways. It can be observed that the omission of any component leads to a decline in the final result. The solution semantics from the base verifier appear to be most crucial to GraphReason. The current method still relies on semantic information, which is reasonable since reasoning steps from different solutions require semantic information for better reasoning. We also notice that reasoning graphs bring a slight improvement to the entire method, thereby proving effectiveness of graph structure. The improvement is not substantial because we do not model the graph structure and semantic information simultaneously, and create a training gap here. Another essential factor is the complexity of graph classification, compounded by the presence of noise and limitations in our training data.

4.4 GraphReason with Different LLMs

To evaluate the compatibility of GraphReason and its effectiveness across various models, we additionally include *gpt-4* (OpenAI, 2023) and *PaLM-2* (Google, 2023) in our experiments. Given our limited computing resources, we utilize the same training data previously sampled from *gpt-3.5-turbo*. For testing in the GSM8K task, we select samples from 100 pieces of data from *gpt-4* and *PaLM-2*

	gpt-3.5-turbo	gpt-4	PaLM-2
Greedy Decode	72.7	87.0	53.0
Voting	82.3	94.0	71.0
Simple Verifier	66.9	89.0	36.0
Voting Verifier	85.4	**97.0**	77.0
DIVERSE	85.0	**97.0**	75.0
Ours	**85.7**	94.0	**78.0**

Table 3: The experimental results of GraphReason with different LLMs.

respectively. We conduct the sampling 10 times using three types of five exemplars, maintaining the same settings as in our previous experiments. Our method aims to enhance the original reasoning capabilities. Therefore, we do not include small-sized LMs, which typically exhibit weaker reasoning abilities.

From Table 3, it is evident that our method enhances the original reasoning performance of both *GPT-4* and *PaLM-2*. However, there is a performance decline in *gpt-4* when compared with the best baselines. The performance of GraphReason is comparable to that of the voting method. We hypothesize that this is because the reasoning patterns of *GPT-4* differ from those of *GPT-3.5-Turbo*, and our verifier is trained specifically on *GPT-3.5-Turbo* samples in this setting.

5 Conclusion

In this paper, we propose GraphReason, a novel and general method to enhance the reasoning abilities of large language models. Our method is the first to approach reasoning logic of large language models from a graph perspective and verifies candidate reasoning paths accordingly. We demonstrate the superiority of GraphReason through extensive experiments.

	GSM8K	SVAMP	ASDiv-a			
Reasoning Graph Verifier (của chúng tôi)	85,7	-	85,4	-	97,0	-
không có giải pháp ngữ nghĩa từ trình xác minh cơ sở	81,2	-4,5	83,1	-2,3	94,3	-2,7
không có giải pháp tổng điểm	82,8	-2,9	83,2	-2,2	95,6	-1,4
đồ thị không có lý luận	85,4	-0,3	84,8	-0,6	96,9	-0,1

Bảng 2: Kết quả thí nghiệm cắt bỏ của GraphReason. Thiếu từng thành phần dẫn đến sự suy giảm trong kết quả cuối cùng. kết quả.

pler so với những dữ liệu trong hai tập dữ liệu khác,
dựa trên những quan sát của chúng tôi. Trong hầu hết các trường hợp, những
các vấn đề không đòi hỏi phải suy luận phức tạp từ
một góc nhìn đồ thị để đưa ra câu trả lời thỏa đáng.
Điều này chứng minh rằng phương pháp của chúng tôi đặc
biệt phù hợp cho những tình huống như vậy. Chúng tôi tin rằng
rằng GraphReason có thể cung cấp những cải tiến đáng kể
hơn trong những tình huống phức tạp hơn.

	gpt-3.5-turbo	gpt-4	PaLM-2
Giải mã tham lam	72,7	87,0	53.0
Bỏ phiếu	82,3	94,0	71.0
Trình xác minh đơn giản	66,9	89,0	36.0
Người xác minh phiếu bầu	85,4	97,0	77,0
PHONG PHÚ	85,0	97,0	75.0
Của chúng tôi	85,7	94,0	78.0

Bảng 3: Kết quả thử nghiệm của GraphReason với các LLM khác nhau.

4.3 Nghiên cứu cắt bỏ

Chúng tôi đã tiến hành một nghiên cứu cắt bỏ để đánh
giá tác động của từng thành phần đối với hiệu suất tổng thể
của phương pháp của chúng tôi. Bảng 2 trình bày kết quả của phương pháp này
nghiên cứu, làm nổi bật cách các mô-đun này đóng góp vào
sự cải tiến của mô hình cơ sở theo những cách riêng biệt.
Có thể thấy rằng việc bỏ sót bất kỳ thành phần nào đều
dẫn đến kết quả cuối cùng giảm sút.
ngữ nghĩa giải pháp từ trình xác minh cơ sở xuất hiện
là quan trọng nhất đối với GraphReason. Hiện tại
phương pháp vẫn dựa vào thông tin ngữ nghĩa,
là hợp lý vì các bước lý luận từ các nguồn khác nhau
các giải pháp đòi hỏi thông tin ngữ nghĩa để tốt hơn
lý luận. Chúng tôi cũng nhận thấy rằng biểu đồ lý luận
mang lại một sự cải thiện nhỏ cho toàn bộ phương pháp,
qua đó chứng minh tính hiệu quả của cấu trúc đồ thị.
Sự cải thiện không đáng kể vì chúng ta làm
không mô hình hóa cấu trúc đồ thị và thông tin ngữ nghĩa
cùng lúc và tạo ra khoảng cách đào tạo
ở đây. Một yếu tố thiết yếu khác là sự phức tạp của
phân loại đồ thị, kết hợp với sự hiện diện
của tiếng ồn và hạn chế trong dữ liệu đào tạo của chúng tôi.

4.4 GraphReason với các LLM khác nhau

Để đánh giá khả năng tương thích của GraphReason và
hiệu quả của nó trên nhiều mô hình khác nhau, chúng tôi cũng
bao gồm gpt-4 (OpenAI, 2023) và PaLM-2
(Google, 2023) trong các thí nghiệm của chúng tôi. Do
tài nguyên tính toán hạn chế, chúng tôi sử dụng cùng
dữ liệu đào tạo được lấy mẫu trước đó từ gpt-3.5-turbo.
Để thử nghiệm trong nhiệm vụ GSM8K, chúng tôi chọn các mẫu
từ 100 dữ liệu từ gpt-4 và PaLM-2
tương ứng. Chúng tôi tiến hành lấy mẫu 10 lần
sử dụng ba loại năm mẫu, duy trì
các thiết lập tương tự như trong các thí nghiệm trước đây của chúng tôi.
Phương pháp của chúng tôi nhằm mục đích nâng cao lý luận ban đầu
khả năng. Do đó, chúng tôi không bao gồm các LM có quy
mô nhỏ, thường thể hiện khả năng lập luận yếu hơn.

Từ Bảng 3, rõ ràng là phương pháp của chúng tôi nâng
cao hiệu suất suy luận ban đầu của cả hai
GPT-4 và PaLM-2. Tuy nhiên, hiệu suất của gpt-4 giảm khi
so sánh với
đường cơ sở tốt nhất. Hiệu suất của GraphReason
có thể so sánh với phương pháp bỏ phiếu. Chúng tôi
đưa ra giả thuyết rằng điều này là do các mô hình lý luận của
GPT-4 khác với mô hình của GPT-3.5-Turbo,
và người xác minh của chúng tôi được đào tạo cụ thể về GPT-3.5-
Mẫu Turbo trong cài đặt này.

5 Kết luận

Trong bài báo này, chúng tôi đề xuất GraphReason, một
và phương pháp chung để nâng cao khả năng lập luận của
các mô hình ngôn ngữ lớn. Phương pháp của chúng tôi là
đầu tiên tiếp cận logic lý luận của ngôn ngữ lớn
mô hình từ góc độ đồ thị và xác minh các đường dẫn suy luận của
ứng viên một cách phù hợp. Chúng tôi chứng minh
sự vượt trội của GraphReason thông qua mở rộng
các thí nghiệm.

Limitations

There are several limitations in the current research that contribute to performance that is not as good as expected:

- **Computing Resources**. Despite the impressive performance it achieves, our framework requires large language models like GPT3.5. Inference with these models is more time-consuming and costly than fine-tuning models like BERT(Devlin et al., 2019). Some experiments, such as hyper-parameter analysis, have already been conducted in related previous work and are not replicated here. Furthermore, due to limited computing resources, we have not conducted experiments with additional LLMs. We have chosen solely to use the representative LLM, GPT3.5, to compare the performance of the verifiers.
- **Labeled CoT data**. `GraphReason` is a complex verifier method that builds on graph classification, which requires more labeled data with well-annotated chain-of-thought reasoning paths for training. In the training of `GraphReason`, we use reasoning paths from LLMs' output which may introduce significant noise. If the training data included labeled reasoning graphs, the performance would improve significantly.
- **Other Reasoning Tasks**. There are many types of reasoning tasks beyond math word problems, such as Commonsense Reasoning (Talmor et al., 2019), Inductive Reasoning (Sinha et al., 2019), etc. Given that graph construction is a complex process, we have focused mainly on solving math word problems (Arithmetic Reasoning). This focus allows for a more convenient implementation of the merging of intermediate steps. In other cases, identifying similar steps can be challenging. On the other hand, a math word problem typically presents a greater variety of potential solutions.

Nevertheless, we believe that future studies, conducted by us or others, can overcome these limitations and further improve upon our approach.

References

Abien Fred Agarap. 2019. Deep learning using rectified linear units (relu). *Preprint*, arXiv:1803.08375.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *Preprint*, arXiv:2204.02311.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021a. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021b. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics (TACL)*.

Google. 2023. Palm 2 technical report. *Preprint*, arXiv:2305.10403.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. *Preprint*, arXiv:2212.10403.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. *Preprint*, arXiv:2205.11916.

Andrew Lampinen, Ishita Dasgupta, Stephanie Chan, Kory Mathewson, Mh Tessler, Antonia Creswell, James McClelland, Jane Wang, and Felix Hill. 2022a. Can language models learn from explanations in context? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 537–563,

Hạn chế

Có một số hạn chế trong nghiên cứu hiện tại góp phần vào hiệu suất không tốt như mong đợi:

- Tài nguyên máy tính. Mặc dù ấn tượng hiệu suất mà nó đạt được, khuôn khổ của chúng tôi yêu cầu các mô hình ngôn ngữ lớn như GPT3.5. Suy luận với những mô hình này tốn nhiều thời gian hơn và tốn kém hơn so với việc tinh chỉnh các mô hình như BERT(Devlin và cộng sự, 2019). Một số thí nghiệm, chẳng hạn như phân tích siêu tham số, đã được tiến hành trong công trình trước đây có liên quan và không được sao chép ở đây. Hơn nữa, do hạn chế về tài nguyên tính toán , chúng tôi chưa tiến hành các thí nghiệm với LLM bổ sung. Chúng tôi đã chọn chỉ sử dụng LLM đại diện, GPT3.5, để so sánh hiệu suất của người xác minh.
- Dữ liệu CoT được gắn nhãn. GraphReason là một phương pháp xác minh dựa trên phân loại đồ thị , đòi hỏi nhiều dữ liệu được gắn nhãn hơn với các đường dẫn suy luận chuỗi suy nghĩ được chú thích rõ ràng cho đào tạo. Trong quá trình đào tạo GraphReason, chúng tôi sử dụng các con đường lý luận từ đầu ra của LLM có thể đưa ra tiếng ồn đáng kể. Nếu dữ liệu đào tạo bao gồm các biểu đồ lý luận có nhãn, hiệu suất sẽ được cải thiện đáng kể.
- Các nhiệm vụ lý luận khác. Có nhiều loại của các nhiệm vụ lý luận vượt ra ngoài các bài toán bằng lời, chẳng hạn như Lý luận thông thường (Talmor và cộng sự, 2019), Lý luận quy nạp (Sinha và cộng sự, 2019), v.v. Vì việc xây dựng đồ thị là một quá trình phức tạp quá trình này, chúng tôi tập trung chủ yếu vào việc giải toán các bài toán có lời văn (Lý luận số học). Trọng tâm này cho phép thực hiện thuận tiện hơn của sự hợp nhất các bước trung gian. Trong những Trong các trường hợp, việc xác định các bước tương tự có thể là một thách thức . Mặt khác, một bài toán bằng lời thường mang lại nhiều tiềm năng đa dạng hơn giải pháp.

Tuy nhiên, chúng tôi tin rằng các nghiên cứu trong tương lai do chúng tôi hoặc những người khác thực hiện có thể khắc phục những hạn chế này và cải thiện hơn nữa phương pháp tiếp cận của chúng tôi.

Tài liệu tham khảo

Abien Fred Agarap. 2019. Học sâu bằng cách sử dụng đã chỉnh sửa đơn vị tuyến tính (relu). Bản in trước, arXiv:1803.08375.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vin-odkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Bành Thành Yin, Công tước Toju, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Diễn viên: Vedant MisraKevin RobinsonLiam FedusDenny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, An- draw M. Dai, Thanumalayan Sankaranarayana Pil-lai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Chu, Xuezhi Wang, Brennan Saeta, Mark Diaz,Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, và Noah Fiedel. 2022. Palm: Mô hình hóa ngôn ngữ theo tỷ lệ với các lộ trình. Bản in trước, arXiv:2204.02311.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse và John Schulman. 2021a. Đào tạo người xác minh để giải các bài toán có lời văn . Bản thảo arXiv arXiv:2110.14168.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse và John Schulman. 2021b. Đào tạo người kiểm chứng để giải các bài toán có lời văn . Bản in trước, arXiv:2110.14168.

Jacob Devlin, Ming-Wei Chang, Kenton Lee và Kristina Toutanova. 2019. Bert: Tiền đào tạo sâu máy biến áp hai chiều để hiểu ngôn ngữ . Bản in trước, arXiv:1810.04805.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth và Jonathan Berant. 2021. Aristotle có sử dụng máy tính xách tay không? Một tiêu chuẩn trả lời câu hỏi với các chiến lược suy luận ngầm. Giao dịch của Hiệp hội Ngôn ngữ học Tính toán (TACL).

Google. 2023. Báo cáo kỹ thuật Palm 2. Bản in trước, arXiv:2305.10403.

Jie Huang và Kevin Chen-Chuan Chang. 2023. Hướng tới lý luận trong các mô hình ngôn ngữ lớn: Một cuộc khảo sát. Bản in trước, arXiv:2212.10403.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yu -taka Matsuo và Yusuke Iwasawa. 2023. Lớn mô hình ngôn ngữ là những trình suy luận không cần dùng cú pháp. Bản in trước, arXiv:2205.11916.

Andrew Lampinen, Ishita Dasgupta, Stephanie Chan, Kory Mathewson, Mh Tessler, Antonia Creswell, James McClelland, Jane Wang và Felix Hill. 2022a. Các mô hình ngôn ngữ có thể học hỏi từ những giải thích trong ngữ cảnh không? Trong Những phát hiện của Hiệp hội Ngôn ngữ học tính toán: EMNLP 2022, trang 537-563,

Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Andrew K Lampinen, Ishita Dasgupta, Stephanie CY Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L McClelland, Jane X Wang, and Felix Hill. 2022b. Can language models learn from explanations in context? *arXiv preprint arXiv:2204.02329*.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333, Toronto, Canada. Association for Computational Linguistics.

Zhongli Li, Wenxuan Zhang, Chao Yan, Qingyu Zhou, Chao Li, Hongzhi Liu, and Yunbo Cao. 2022. Seeking patterns, not just memorizing procedures: Contrastive learning for solving math word problems. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2486–2496, Dublin, Ireland. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Xinyu Pi, Qian Liu, Bei Chen, Morteza Ziyadi, Zeqi Lin, Qiang Fu, Yan Gao, Jian-Guang Lou, and Weizhu Chen. 2022. Reasoning like program executors. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 761–779, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2022. Scaling language models: Methods, analysis & insights from training gopher. *Preprint*, arXiv:2112.11446.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. 2019. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4506–4515, Hong Kong, China. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Large language models still can't plan (a benchmark for llms on planning and reasoning about change). *Preprint*, arXiv:2206.10498.

Siyuan Wang, Wanjun Zhong, Duyu Tang, Zhongyu Wei, Zhihao Fan, Daxin Jiang, Ming Zhou, and Nan Duan. 2022. Logic-driven context extension and data augmentation for logical reasoning of text. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1619–1629, Dublin, Ireland. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Abu Dhabi, Các Tiểu vương quốc Ả Rập Thống nhất. Hiệp hội Ngôn ngữ học tính toán.

Andrew K Lampinen, Ishita Dasgupta, Stephanie CY Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L McClelland, Jane X Wang và Felix Hill. 2022b. Liệu các mô hình ngôn ngữ có thể học hỏi từ các giải thích trong ngữ cảnh không? Bản thảo trước arXiv arXiv:2204.02329.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou và Weizhu Chen. 2023. Làm cho các mô hình ngôn ngữ có khả năng suy luận tốt hơn bằng trình xác minh nhận biết từng bước . Trong Biên bản Hội nghị thường niên lần thứ 61 của Hiệp hội Ngôn ngữ học tính toán (Tập 1: Bài báo dài), trang 5315–5333, Toronto, Canada. Hiệp hội Ngôn ngữ học tính toán.

Zhongli Li, Wenxuan Zhang, Chao Yan, Qingyu Chu, Chao Li, Hongzhi Liu và Yunbo Cao. 2022. Tìm kiếm các khuôn mẫu, không chỉ các quy trình ghi nhớ: Học tập tương phản để giải các bài toán đố. Trong Phát hiện của Hiệp hội Ngôn ngữ học Tính toán: ACL 2022, trang 2486–2496, Dublin, Ireland. Hiệp hội Ngôn ngữ học Tính toán.

Ilya Loshchilov và Frank Hutter. 2019. Chính quy hóa sự suy giảm trọng lượng tách rời. Bản in trước, arXiv:1711.05101.

Shen-yun Miao, Chao-Chun Liang và Keh-Yih Su. 2020. Một kho ngữ liệu đa dạng để đánh giá và phát triển các công cụ giải toán bằng lời tiếng Anh. Trong Kỷ yếu Hội nghị Thường niên lần thứ 58 của Hiệp hội Ngôn ngữ học Tính toán, trang 975–984.

OpenAI. 2023. Báo cáo kỹ thuật Gpt-4. Bản in trước, arXiv:2303.08774.

Arkil Patel, Satwik Bhattamishra và Navin Goyal. 2021. Liệu các mô hình NLP có thực sự có thể giải quyết các bài toán đơn giản không? Trong Biên bản Hội nghị năm 2021 của Chi hội Bắc Mỹ thuộc Hiệp hội Ngôn ngữ học tính toán: Công nghệ ngôn ngữ của con người, trang 2080–2094, Trực tuyến. Hiệp hội Ngôn ngữ học tính toán.

Xinyu Pi, Qian Liu, Bei Chen, Morteza Ziyadi, Zeqi Lin, Qiang Fu, Yan Gao, Jian-Guang Lou và Weizhu Chen. 2022. Lý luận như người thực thi chương trình. Trong Kỷ yếu Hội nghị năm 2022 về Phương pháp thực nghiệm trong Xử lý ngôn ngữ tự nhiên, trang 761–779 , Abu Dhabi, Các Tiểu vương quốc Ả Rập Thống nhất. Hiệp hội Ngôn ngữ học tính toán.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susan-nah Young, Eliza Rutherford, Tom Hennigan, Ja- cob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Mari-beth Rauh, Po-Sen Huang, Amelia Glaese, Jo-hannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden , Esme Sutherland, Karen Simonyan, Michela Paganini , Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsim- poukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorraine Bennett, Demis Hassabis, Ko-ray Kavukcuoglu và Geoffrey Irving. 2022. Mở rộng quy mô mô hình ngôn ngữ: Phương pháp, phân tích và hiểu biết sâu sắc từ đào tạo gopher. Bản in trước, arXiv:2112.11446.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau và William L. Hamilton. 2019. CLUTRR: Một chuẩn mực chẩn đoán cho lý luận quy nạp từ chữ. Trong Kỷ yếu Hội nghị năm 2019 về Phương pháp thực nghiệm trong Xử lý ngôn ngữ tự nhiên và Hội nghị chung quốc tế lần thứ 9 về Xử lý ngôn ngữ tự nhiên (EMNLP-IJCNLP), trang 4506–4515, Hồng Kông, Trung Quốc. Hiệp hội Ngôn ngữ học tính toán.

Alon Talmor, Jonathan Herzig, Nicholas Lourie và Jonathan Berant. 2019. CommonsenseQA: Một câu hỏi trả lời thách thức nhắm vào ý thức chung kiến thức. Trong Biên bản Hội nghị năm 2019 của Chi hội Bắc Mỹ thuộc Hiệp hội Ngôn ngữ học tính toán: Công nghệ ngôn ngữ của con người , Tập 1 (Bài báo dài và ngắn), trang 4149–4158, Minneapolis, Minnesota. Hiệp hội Ngôn ngữ học tính toán.

Karthik Valmeekam, Alberto Olmo, Sarath Sreedha-ran và Subbarao Kambhampati. 2023. Các mô hình ngôn ngữ lớn vẫn không thể lập kế hoạch (một chuẩn mực cho llms về việc lập kế hoạch và lý luận về sự thay đổi). Bản in trước, arXiv:2206.10498.

Siyuan Wang, Wanjun Zhong, Duyu Tang, Zhongyu Wei, Zhihao Fan, Daxin Jiang, Ming Chu và Nan Duan. 2022. Dữ liệu và mở rộng ngữ cảnh theo hướng logic tăng cường khả năng suy luận logic của văn bản. Trong Phát hiện của Hiệp hội Ngôn ngữ học tính toán: ACL 2022, trang 1619–1629, Dublin, Ireland.
Hiệp hội Ngôn ngữ học tính toán.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quốc Lê, Ed Chi, Sharan Narang, Aakanksha Chowdhery và Denny Chu. 2023. Tự thống nhất cải thiện chuỗi của tư duy lý luận trong các mô hình ngôn ngữ. Bản in trước, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quốc Lê và Denny Chu. 2022.
Chuỗi suy nghĩ gợi ra lý luận trong các mô hình ngôn ngữ lớn. Bản thảo trước của arXiv arXiv:2201.11903.

Zhipeng Xie and Shichao Sun. 2019. A goal-driven tree-structured neural model for math word problems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5299–5305. International Joint Conferences on Artificial Intelligence Organization.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *International Conference on Learning Representations*.

Ori Yoran, Alon Talmor, and Jonathan Berant. 2022. Turning tables: Generating examples from semi-structured tables for endowing language models with reasoning skills. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6016–6031, Dublin, Ireland. Association for Computational Linguistics.

Dongxiang Zhang, Lei Wang, Luming Zhang, Bing Tian Dai, and Heng Tao Shen. 2019. The gap of semantic parsing: A survey on automatic math word problem solvers. *Preprint*, arXiv:1808.07290.

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 2020. Graph-to-tree learning for solving math word problems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3928–3937, Online. Association for Computational Linguistics.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *Preprint*, arXiv:2303.18223.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. *Preprint*, arXiv:2205.10625.

A Datasets

We compared GraphReason with other methods on three different math word problem datasets: **GSM8K** (Cobbe et al., 2021a), **SVAMP** (Patel et al., 2021), and **ASDiv-a** (Miao et al., 2020) and one commonsense reasoning dataset: **StrategyQA** (Geva et al., 2021). We selected the subset ASDiv-a (arithmetic) from the original dataset ASDiv, which only involves arithmetic operations.

These three arithmetic reasoning datasets are more challenging than other math word problem datasets, making them more suitable for testing the reasoning capability of LLMs with a verifier. As the GSM8K dataset is the only one providing step-by-step solutions as chain-of-thought exemplars, we chose exemplars from the GSM8K training dataset and tested them on all three datasets. Additionally, the training data for the verifier also used the GSM8K training data. In this setting, we could also demonstrate the transfer learning and generalization ability of our method. The size of the training split from GSM8k is 1000. The test data sizes for GSM8K, SVAMP, and ASDiv-a are 1319, 1000, and 1218, respectively.

In the StrategyQA commonsense reasoning task, we set the number of exemplars to 8 and select pseudo-exemplars from (Li et al., 2023). Additionally, we conduct five sampling iterations for each context of LLMs. From the entire dataset, we select a subset of 1,000 instances, allocating 700 for training and 300 for testing.

B Baselines

In our evaluation, we consider the following baselines:

- **Greedy Decode** is a simple method that uses a greedy decoding strategy to sample once.
- **Self-Consistency (Voting)** (Wang et al., 2023) samples multiple times and selects the final answers based on majority voting.
- **Simple Verifier** (Cobbe et al., 2021b), which is also known as the Sampling and Re-ranking strategy, uses a verifier to assign scores to sampled solutions and selects the final answer with the highest score.
- **Voting Verifier** (Li et al., 2023) combines the Voting and Verifier approaches. It assigns total scores to answers from scores of all candidate solutions and selects the final answer with the highest score.
- **DIVERSE (Step-aware Voting Verifier)** (Li et al., 2023), which is the state-of-the-art method, considers the reasoning steps throughout the entire reasoning path. It recognizes that not all steps in an incorrect reasoning path are equally wrong and that some steps may still be useful for reasoning.

We primarily compare GraphReason with other verifiers using the same generated solutions from *gpt-3.5-turbo*. Additionally, we include some previous Fine-tuning state-of-the-art methods to reflect

Zhipeng Xie và Shichao Sun. 2019. Định hướng theo mục tiêu mô hình nơ-ron có cấu trúc cây cho các bài toán có lời văn. Trong Biên bản Hội nghị chung quốc tế lần thứ hai mươi tám về Trí tuệ nhân tạo, IJCAI-19, trang 5299–5305. Tổ chức Hội nghị chung quốc tế về Trí tuệ nhân tạo.

Keyulu Xu, Weihua Hu, Jure Leskovec và Stefanie Jegelka. 2019. Mạng nơ-ron đồ thị mạnh mẽ như thế nào ? Trong Hội nghị quốc tế về biểu diễn học tập.

Ori Yoran, Alon Talmor và Jonathan Berant. 2022. Bàn xoay: Tạo các ví dụ từ các bảng bán cấu trúc để cung cấp cho các mô hình ngôn ngữ kỹ năng lập luận. Trong Biên bản Hội nghị thường niên lần thứ 60 của Hiệp hội Ngôn ngữ học tính toán (Tập 1: Bài báo dài), trang 6016–6031, Dublin, Ireland. Hiệp hội Ngôn ngữ học tính toán .

Dongxiang Zhang, Lei Wang, Luming Zhang, Bing Tian Dai và Heng Tao Shen. 2019. Khoảng cách ngữ nghĩa phân tích cú pháp: Một cuộc khảo sát về bài toán toán tự động người giải quyết. Bản in trước, arXiv:1808.07290.

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao và Ee-Peng Lim. 2020. Học đồ thị trên cây để giải các bài toán đố. Trong Biên bản Hội nghị thường niên lần thứ 58 của Hiệp hội Ngôn ngữ học tính toán, trang 3928–3937 , Trực tuyến. Hiệp hội Ngôn ngữ học tính toán .

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Be-ichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zhang Zhang, Peiyu Liu, Jian-Yun Nie và Ji-Rong Wen.
2023. Khảo sát các mô hình ngôn ngữ lớn. Bản in trước, arXiv:2303.18223.

Denny Chu, Nathanael Schärli, Lê Hầu, Jason Wei, Nathan Cân, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quốc Lê và Ed Chi.
2023. Việc nhắc nhở từ ít đến nhiều cho phép lập luận phức tạp trong các mô hình ngôn ngữ lớn. Bản in trước, arXiv:2205.10625.

Một tập dữ liệu

Chúng tôi đã so sánh GraphReason với các phương pháp khác trên ba tập dữ liệu bài toán lời văn khác nhau: GSM8K (Cobbe và cộng sự, 2021a), SVAMP (Patel và cộng sự, 2021) và ASDiv-a (Miao và cộng sự, 2020) và một tập dữ liệu suy luận thông thường: StrategyQA (Geva và cộng sự, 2021). Chúng tôi đã chọn tập con ASDiv-a (số học) từ tập dữ liệu gốc ASDiv, chỉ bao gồm các phép toán số học.

Ba tập dữ liệu lý luận số học này có tính thách thức hơn các tập dữ liệu bài toán bằng lời khác, khiến chúng phù hợp hơn để thử nghiệm khả năng lý luận của LLM với người xác minh.
Vì tập dữ liệu GSM8K là tập dữ liệu duy nhất cung cấp các giải pháp từng bước dưới dạng ví dụ về chuỗi suy nghĩ , chúng tôi đã chọn các ví dụ từ tập dữ liệu đào tạo GSM8K và thử nghiệm chúng trên cả ba tập dữ liệu.
Ngoài ra, dữ liệu huấn luyện cho bộ xác minh cũng sử dụng dữ liệu huấn luyện GSM8K. Trong bối cảnh này, chúng tôi cũng có thể chứng minh khả năng học chuyển giao và khả năng khái quát hóa của phương pháp. Kích thước của phần huấn luyện tách từ GSM8K là 1000. Kích thước dữ liệu thử nghiệm cho GSM8K, SVAMP và ASDiv-a lần lượt là 1319, 1000 và 1218.

Trong bài tập suy luận theo lẽ thường StrategyQA, chúng tôi đặt số lượng mẫu là 8 và chọn các mẫu giả từ (Li và cộng sự, 2023). Ngoài ra, chúng tôi thực hiện năm lần lặp lại lấy mẫu cho mỗi ngữ cảnh của LLM. Từ toàn bộ tập dữ liệu, chúng tôi chọn một tập con gồm 1.000 trường hợp, phân bổ 700 cho đào tạo và 300 cho thử nghiệm.

B Đường cơ sở

Trong quá trình đánh giá, chúng tôi xem xét các cơ sở sau :

- Giải mã tham lam là một phương pháp đơn giản sử dụng chiến lược giải mã tham lam để lấy mẫu một lần.
- Tự nhất quán (Biểu quyết) (Wang và cộng sự, 2023) lấy mẫu nhiều lần và chọn câu trả lời cuối cùng dựa trên biểu quyết đa số.
- Simple Verifier (Cobbe et al., 2021b), còn được gọi là chiến lược Lấy mẫu và Xếp hạng lại , sử dụng một trình xác minh để chỉ định điểm cho các giải pháp lấy mẫu và chọn câu trả lời cuối cùng có điểm cao nhất.
- Voting Verifier (Li và cộng sự, 2023) kết hợp các phương pháp Voting và Verifier. Phương pháp này gán tổng điểm cho câu trả lời từ điểm của tất cả ứng viên.
giải pháp và chọn câu trả lời cuối cùng với
điểm cao nhất.
- DIVERSE (Bộ xác minh bỏ phiếu nhận biết bước) (Li và cộng sự, 2023), là phương pháp tiên tiến nhất, xem xét các bước suy luận trong toàn bộ đường dẫn suy luận. Phương pháp này thừa nhận rằng không phải tất cả các bước trong một đường dẫn suy luận không chính xác đều sai như nhau và một số bước vẫn có thể hữu ích cho việc suy luận.

Chúng tôi chủ yếu so sánh GraphReason với các trình xác minh khác sử dụng cùng các giải pháp được tạo ra từ gpt-3.5-turbo. Ngoài ra, chúng tôi còn đưa vào một số phương pháp tinh chỉnh tiên tiến trước đây để phản ánh

the strong reasoning ability of LLMs. The previous Fine-tuning SOTA methods are denoted as follows: a: (Cobbe et al., 2021b), b: (Pi et al., 2022), c: (Miao et al., 2020), d: (Chowdhery et al., 2022).

khả năng lập luận mạnh mẽ của các LLM. Các phương pháp Fine-tuning SOTA trước đây được ký hiệu như sau: a: (Cobbe và cộng sự, 2021b), b: (Pi và cộng sự, 2022), c: (Miao và cộng sự, 2020), d: (Chowdhery và cộng sự, 2022).

12