Towards Large Reasoning Models: A Survey of Reinforced Reasoning with Large Language Models

Fengli Xu[1] **Qianyue Hao**[1] **Zefang Zong**[1] **Jingwei Wang**[1] **Yunke Zhang**[1]
Jingyi Wang[1] **Xiaochong Lan**[1] **Jiahui Gong**[1] **Tianjian Ouyang**[1] **Fanjin Meng**[1]
Chenyang Shao[1] **Yuwei Yan**[2] **Qinglong Yang**[1] **Yiwen Song**[1] **Sijian Ren**[1]
Xinyuan Hu[3] **Yu Li**[1] **Jie Feng**[1] **Chen Gao**[1] **Yong Li**[1]
All authors contributed equally.
1 Tsinghua University, Beijing, China; 2 HKUST (GZ), Guangzhou, China
3 Emory University, Atlanta GA, USA
fenglixu@tsinghua.edu.cn, liyong07@tsinghua.edu.cn

Abstract

Language has long been conceived as an essential tool for human reasoning. The breakthrough of Large Language Models (LLMs) has sparked significant research interest in leveraging these models to tackle complex reasoning tasks. Researchers have moved beyond simple autoregressive token generation by introducing the concept of "thought"—a sequence of tokens representing intermediate steps in the reasoning process. This innovative paradigm enables LLMs' to mimic complex human reasoning processes, such as tree search and reflective thinking. Recently, an emerging trend of learning to reason has applied reinforcement learning (RL) to train LLMs to master reasoning processes. This approach enables the automatic generation of high-quality reasoning trajectories through trial-and-error search algorithms, significantly expanding LLMs' reasoning capacity by providing substantially more training data. Furthermore, recent studies demonstrate that encouraging LLMs to "think" with more tokens during test-time inference can further significantly boost reasoning accuracy. Therefore, the train-time and test-time scaling combined to show a new research frontier—a path toward Large Reasoning Model. The introduction of OpenAI's o1 series marks a significant milestone in this research direction. In this survey, we present a comprehensive review of recent progress in LLM reasoning. We begin by introducing the foundational background of LLMs and then explore the key technical components driving the development of large reasoning models, with a focus on automated data construction, learning-to-reason techniques, and test-time scaling. We also analyze popular open-source projects at building large reasoning models, and conclude with open challenges and future research directions.

1 Introduction

"If there is a severe deficit of language, there will be severe deficit of thought" — Noam Chomsky

Fueled by advancements in deep learning and the availability of web-scale datasets, Large Language Models (LLMs) have emerged as a transformative paradigm on the path toward Artificial General Intelligence (AGI). These massive AI models typically adopt Transformer architecture and are pre-trained on large-scale text corpus with the next-token prediction task [191]. The neural scaling law demonstrates that their performance improves significantly as the model size and training data increase [59]. More importantly, LLMs also unlock remarkable *emergent abilities* that are absent in smaller models [159], such as in-context learning [33], role playing [124] and analogical reason-

ing [157]. These abilities allow LLMs go beyond natural language processing problems to facilitate a wider range of tasks, such as code generation [41], robotic control [3], and autonomous agents [28].

Among these abilities, human-like reasoning has garnered significant attention from both academia and industry, since it demonstrates great potential for LLMs to generalize to complex real-world problems through abstract and logical reasoning. A notable breakthrough in this area is the "chain-of-thought" prompting technique [160], which can elicit step-by-step human-like reasoning processes at test time without any additional training. Such intuitive prompting techniques have been proven effective to substantially improve the reasoning accuracy of pre-trained LLMs, which also leads to the development of more advanced prompting techniques like "tree-of-thought" [172]. These approaches introduce the concept of "thought" as a sequence of tokens that represents the intermediate steps in human-like reasoning process. By incorporating such intermediary steps, LLM reasoning moves beyond simple autoregressive token generation, enabling more sophisticated cognitive architectures like tree search [172] and reflective reasoning [180].

Recently, there has been a significant research trend in learning to reason [103], which seeks to train LLMs to master human-like reasoning processes. A key challenge in this research direction is the lack of training data. Human annotation is often prohibitively expensive, particularly for step-by-step reasoning trajectories that have proven effective in supervising LLM reasoning [75]. To address this issue, recent studies have shifted from human annotation to LLM-driven search algorithms. These approaches utilize external verification for reasoning problems to automatically generate accurate reasoning trajectories through trial-and-error search [85]. More importantly, researchers have proposed to train Process Reward Models (PRMs) on these reasoning trajectories [183]. PRMs can provide dense, step-wise rewards that facilitate reinforcement learning for LLM reasoning. These methods combined to reduce the reliance on human annotation data, and create a "reinforced cycle" for augmenting LLM reasoning that effectively integrates "search" and "learning", which are the two methods that can scale endlessly as predicted by Richard Sutton [139]. Therefore, this novel paradigm enables the scaling of LLM reasoning capabilities with increased train-time compute, paving the way for more advanced reasoning models.

Moreover, recent study shows scaling up test-time compute can also improve LLM reasoning accuracy. Specifically, PRMs can be used to guide LLMs to evaluate and search through the intermediate "thoughts" [134], which encourages LLMs to generate deliberate reasoning steps during test-time computation and boosts reasoning accuracy. This approach gives rise to the test-time scaling law, which predicts spending more tokens for deliberate reasoning at test-time can improve accuracy [103]. Therefore, the RL-driven train-time scaling and search-based test-time scaling combined to show a promising research direction to fully unleash the reasoning capabilities of LLMs, *i.e.*, a path toward *Large Reasoning Models*. A key milestone in this research direction is OpenAI's o1 series [194], which demonstrates the effectiveness of this approach and echoes OpenAI's vision of transitioning LLMs from *conversational AI* (level 1) to more powerful *reasoning AI* (level 2) in the five-steps road map toward AGI [36]. Several open-source projects, such as OpenR [145], LLaMA-Berry [185] and Journey Learning [110], are dedicated to reproduce the strong reasoning capacity of OpenAI's o1, providing valuable insights for developing large reasoning models.

In this survey, we provide a comprehensive review of recent research efforts in the progression toward large reasoning models. Section 2 offers a brief introduction to the background of LLM reasoning. The subsequent three sections delve into the key technical components driving the development of large reasoning models. Specifically, Section 3 focuses on training data construction, emphasizing the shift from human annotation to LLM-driven automated search. Section 4 reviews reinforcement learning methods that are pivotal for scaling LLM reasoning capabilities with increased train-time compute, while Section 5 discusses test-time scaling with a particular emphasis on PRM-guided search. In Section 6, we analyze the development of OpenAI's o1 series and other open-source projects, exploring the path toward large reasoning models. Section 7 summarizes additional test-time enhancement techniques, and Section 8 reviews reasoning benchmarks. Finally, we conclude the survey with a discussion of open problems and future research directions.

2 Background

2.1 Pre-trianing

As the foundational stage of training LLMs, effective pretraining is crucial for developing reasoning abilities. Before discussing pretraining for LLMs' reasoning, we first outline the basic process of general LLM pretraining. Through pretraining, LLMs not only acquire core linguistic knowledge but also gain diverse world knowledge, establishing a robust groundwork for the emergence of advanced capabilities and effective value alignment [191]. Typically, LLM pretraining relies on high-quality text corpora [35, 168], including extensive collections of web content, books, codes and other types of data. Leveraging these rich textual corpora, LLMs are built on the transformer architecture that are trained with next-token prediction task. After pretraining, LLMs generally demonstrate exceptional in-context learning capabilities [14], enabling them to generate coherent text and provide accurate answers to a wide range of questions by utilizing their vast knowledge base. Notably, the pretraining stage plays a pivotal role in cultivating the reasoning abilities of LLMs. For example, research [160] has shown that datasets rich in code and mathematical content serve as a key foundation for developing robust reasoning skills. Following this observations, newly developed LLMs [1] begin to introduce carefully designed synthetic data for enhancing the reasoning abilities of LLMs. During pretraining, a critical challenge lies in balancing the proportion of code and mathematical data with general text corpora to maintain strong general linguistic abilities while unlocking the reasoning potential of LLMs.

2.2 Fine-tuning

While pretraining enables LLMs to exhibit reasoning abilities through in-context learning, fine-tuning techniques are widely employed to achieve zero-shot and improved reasoning capabilities for LLMs. Here, we first outline the basic fine-tuning process and then explore its potential for enhancing reasoning abilities. As described in [104], after the pretraining stage, LLMs enter a supervised fine-tuning phase (SFT), also referred to as the instruction tuning stage. The primary goal of this phase is to refine the model's output style, ensuring its responses are aligned with human needs and real-life applications. This is achieved by training via diverse instruction datasets that reflect a wide range of everyday human interactions, typically created through extensive and carefully curated manual annotation and refinement [195]. With the advent of ChatGPT, new methods have emerged for generating diverse instruction datasets. These include techniques that distill data directly from powerful LLMs [153, 167] and automated approaches for large scale dataset construction from existing corpora [158, 32]. Using these well-crafted instruction-tuning datasets, the fine-tuning process continually uses the next-token prediction objective, similar to pretraining. However, unlike pretraining, fine-tuning specifically calculates the loss for the answers while generally ignoring the loss for the questions. Additionally, incorporating datasets that include chain-of-thought (CoT) [160] reasoning and mathematical problem-solving examples has been shown to significantly enhance the reasoning capabilities of LLMs, making this an area of active research. Following the practice in general aspects, most current approaches leverage data distillation from advanced large reasoning models, followed by fine-tuning to enhance the reasoning capabilities of LLMs to obtain the final large reasoning models.

2.3 Alignment

Relying solely on direct data distillation from advanced large reasoning models limits the potential of new LLMs. A more promising approach is to use reinforcement learning for data construction and model training, which precisely corresponds to the final alignment stage in general LLM training. In the general training of LLM, the alignment phase typically involves methods such as Reinforcement Learning from Human Feedback (RLHF) [104] to guide the model toward generating content that meets the criteria of being helpful, harmless, and honest. The goal of this phase is to enhance the safety and controllability of LLMs in the reality. Compared to the former SFT phase, this stage usually incorporates a large amount of carefully curated, manually labeled ranking data to accurately reflect human preferences [35, 168]. This data includes not only correct demonstrations but also undesirable cases that should be avoided. Standard RLHF typically involves an SFT model, a reward model, and a aligned model, which are iteratively optimized using methods like PPO [121]. Due to the high data requirements and training costs of standard RLHF, methods like Direct Preference

Optimization (DPO) [112], have been proposed to reduce reliance on explicit reward models. In DPO, preference loss is defined as a function of the policy to directly guide model optimization. Given the multi-step nature and complexity of reasoning problems, alignment-based post-training has become the final and most critical step in stimulating the reasoning capabilities of LLMs. By carefully decomposing the reasoning process and gradually feeding signals back to the model, various self-training methods [45, 64, 183] based on reinforcement learning and preference learning have achieved notable success.

2.4 Prompting LLMs for Advanced Reasoning

Human-like reasoning is one of the most important abilities that emerge in LLMs with sufficiently large model parameters [157]. While zero-shot reasoning may remain unreliable for some tasks, researchers have discovered various prompting techniques to enhance these capabilities. These techniques can be broadly categorized into three main approaches: step-by-step reasoning, multi-path exploration, and decomposition-based methods.

The step-by-step reasoning approach, exemplified by Chain-of-Thought prompting [160], demonstrates that explicitly showing intermediate reasoning steps significantly improves problem-solving abilities. Even simple prompts like "Let's think step by step" can effectively guide the reasoning process [62]. This approach has been further refined through Self-Consistency [153], which generates multiple reasoning paths to arrive at more reliable conclusions, and Auto-CoT [189], which automates the generation of effective reasoning chains.

Multi-path exploration approaches extend beyond linear reasoning by considering multiple potential solution paths simultaneously. Tree of Thoughts [172] organizes alternative reasoning pathways in a tree structure, enabling systematic exploration of different solution strategies. Graph-of-Thoughts [11] further generalizes this to a graph structure, allowing for more flexible reasoning patterns and backtracking capabilities. ReAct [173] enriches this paradigm by interleaving reasoning with action steps, enabling more dynamic interaction with external environments.

For complex problems, decomposition-based methods have proven particularly effective. Least-to-Most Prompting [196] and Algorithm of Thoughts [122] systematically break down complex problems into manageable components, while Plan-and-Solve [147] provides strategic guidance for tackling these subproblems. These methods are especially valuable when dealing with tasks that require multiple steps or different levels of analysis.

These extensive reasoning capabilities, enhanced through structured prompting strategies, have proven particularly effective for tasks requiring careful analysis and systematic thinking, enabling LLMs to accomplish a wide variety of complex social scientifically relevant tasks. The success of these methods demonstrates that while LLMs possess inherent reasoning abilities, their full potential can be unlocked through careful guidance and structure in the prompting process.

2.5 Agentic Workflow

On top of the instruction following and in-context learning capabilities of LLMs, researchers start to design agentic workflows that program the "thinking patterns" of LLMs [137]. Such agentic workflows allow researchers to enhance LLM's reasoning ability without any additional training, but it often requires more test-time compute. In-context learning [33, 25] is the ability to improve LLM's task specific performance by simply providing a few in-context demonstrations, enables LLMs to efficiently generalize to unseen problems without computationally expensive trainings [14]. Although the origin of such capabilities remains a largely debatable topic, recent studies suggest in-context learning improves LLMs' performance by allowing them to capture the label space, the distribution of input text, and the desired format of answers [97]. Such desirable features have enabled researchers to adapt general purpose LLMs to diverse task scenarios, such as simulating the perspective of certain demographic groups through in-context role play [22]. Recent studies suggest effective agentic workflow can largely improve LLMs abilities for simulating human behavior [105, 127], human-LLM interaction [89], and collaborative task solving [107]. The ability to program LLMs with agentic workflow lays the foundation of improving LLM's reasoning capabilities with complex cognitive architecture.

3 Data Construction: from Human Annotation to LLM Automation

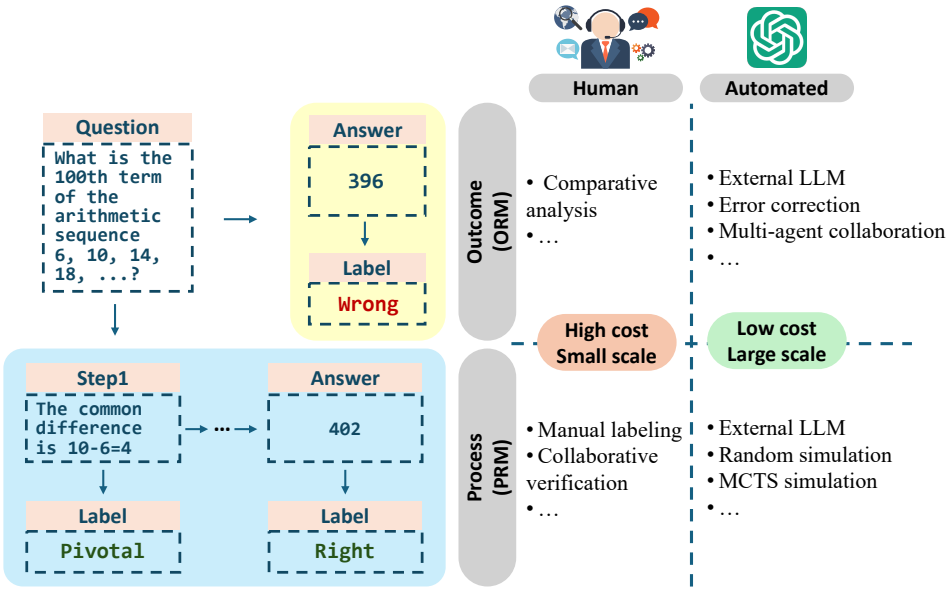

Figure 1: Illustrating different paradigms for annotating LLM reasoning data.

Creating large-scale, high-quality reasoning datasets is crucial for enhancing the reasoning capabilities LLMs. However, this task poses significant challenges due to its high cost. As shown in Figure 1, human annotation is widely considered of high-quality but is prohibitively expensive and difficult to scale. Conversely, automating the annotation process with LLMs offers a more cost-effective alternative but faces the challenge of limited validation, particularly for step-by-step reasoning processes. In this section, we review recent research efforts in this area (summarized in Table 1), highlighting the shift from human annotation to LLM automation.

3.1 Human Annotation

The role of human annotation in constructing datasets for LLMs is indispensable. Human annotators are characterized by their meticulousness, patience, and precision, as well as their adaptability to novel scenarios and capability to handle ambiguous data effectively [98]. Zhou et al. [195] demonstrate that even with minimal human-annotated data, models can achieve strong performance, highlighting the critical role of carefully curated annotations in model effectiveness. Human-annotated data plays a pivotal role in enhancing the reasoning capabilities of large language models. In the context of Reinforcement Learning with Human Feedback (RLHF)[104], preference data from human annotators enables LLMs, initially trained on general text corpora, to align with intricate human values and complex ethical considerations. This generalizable approach of annotation helps in fine-tuning models for specific tasks. Building on this foundation, Lightman et al.[75] demonstrated the efficacy of using human annotators to evaluate the reasoning quality at each step of mathematical reasoning processes, significantly improving the accuracy of LLM reasoning. This highlights how human annotation can bridge the gap between general training data and domain-specific challenges, such as complex reasoning tasks.

Enhancing reasoning capabilities in LLMs requires process supervision, where human annotators guide each step of the reasoning process [75]. However, such supervision demands extensive human-annotated data, making it resource-intensive and unsustainable. Given that LLM training typically requires terabytes of data, the volume of which is critical for model performance, constructing datasets purely through manual annotation becomes increasingly impractical. This highlights the need for alternative approaches to improve reasoning without relying solely on human annotation. One promising approach is the collaboration between humans and LLMs for annotation, where LLMs are leveraged to accelerate the process while preserving the high quality of human-generated annotations. Specifically, the annotation process can be divided into two stages: the pre-annotation

Table 1: Training data construction for LLM reasoning.

Method	Label	Paper	Year	Task	Brief Description
Human Annotation	Outcome	[98]	2024	Text classification Semantic analysis	Voting annotation
		[104]	2022	Preference Alignment	Preference ranking
	Process	[75]	2023	Mathematical reasoning	Stepwise annotation
Human-LLM Collaboration	Outcome	[42]	2023	Semantic analysis	Human correction
		[152]	2024	Text classification	Human correction
		[74]	2023	Text classification Semantic analysis	Task allocation Uncertainty assessment
LLM Automation	Outcome	[106]	2020	Commonsense reasoning	Text extraction
		[120]	2024	Tool use	Trial and error
		[65]	2024	Embodied tasks	Synthetic augmentation
		[109]	2024	Commonsense reasoning Domain knowledge reasoning	Multi-agent collaboration
	Process	[84]	2023	Mathematical reasoning	Stronger LLM
		[148]	2024	Mathematical reasoning	Monte Carlo simulation
		[156]	2024	Mathematical reasoning Programming	Monte Carlo simulation
		[85]	2024	Mathematical reasoning	MCTS simulation
LLM Automation with feedback	Outcome	[70]	2024	Text classification Mathematical reasoning Domain knowledge reasoning	Self-refining
		[135]	2024	Embodied tasks	Contrastive learning
	Process	[183]	2024	Mathematical reasoning Domain knowledge reasoning	MCTS simulation Self-refining

stage and the refinement stage. During the pre-annotation stage, LLMs can be employed to perform an initial round of annotations, taking advantage of a few manually provided examples for a quick and efficient setup [42, 61]. In the refinement stage, human annotators can assess the quality of LLM-generated annotations and focus on correcting only the subset of annotations with poor quality [61, 152, 96, 42]. To enable scalable annotation processes, recent works have increasingly focused on how to maximize automation while ensuring data quality, thus reducing human involvement without compromising the accuracy of the annotations.

3.2 LLM Automated Outcome Annotation

Data annotation is a challenging and resource-intensive task, particularly in scenarios requiring complex operations such as filtering, identifying, organizing, and reconstructing textual data. These tasks are often tedious, time-consuming, and demand significant human effort, making them a costly bottleneck in large-scale data construction efforts [142, 31]. To address these challenges, leveraging LLMs for data annotation provides a cost-effective and efficient alternative. With context window lengths exceeding 100k tokens, LLMs can effortlessly process lengthy texts and large volumes of structured data [2], handling the intricate requirements of data annotation with remarkable efficiency. Their strong instruction-following capabilities [187] enable them to flexibly accommodate diverse and complex annotation scenarios, while achieving a level of quality comparable to that of human annotators. By automating these demanding tasks, LLMs significantly reduce the reliance on human labor, streamlining the annotation process and enhancing overall productivity [181].

LLMs are capable of handling a wide variety of automated annotation tasks, ranging from simple question-answer extraction [106] to the inclusion of additional target information [161]. Without human demonstrations, LLMs rely on their powerful reasoning and in-context learning abilities to independently address more complex annotation needs. For instance, Schick et al.[120] demon-

strated how LLMs can be used to construct datasets for tool usage. For each candidate position that may require an API call, the LLM is able to comprehend the logical relationships within the surrounding context, generate relevant questions, and identify the appropriate tool API to address the issue. When human demonstrations are available, LLMs can further enhance their performance by mimicking the patterns and reasoning strategies illustrated in these examples. For complex tasks, human demonstrations provide high-quality trajectories—sequences of thoughts, observations, or actions—that guide the LLMs in replicating human decision-making processes. Existing studies have shown that even zero-shot LLMs, guided by task-agnostic prompts based on human demonstrations, can perform annotation tasks effectively [65]. Moreover, for tasks involving highly intricate and nuanced trajectories, LLMs can incorporate specialized agents, such as a Plan-Agent, Tool-Agent, and Reflect-Agent, to address different aspects of the annotation process, thereby further enhancing their ability to align with human-like reasoning and behavior [109]. These diverse capabilities extend naturally to reasoning outcome annotation tasks, where LLMs not only infer underlying logical structures but also systematically document intermediate reasoning steps and their associated conclusions. This enables the creation of annotated datasets that capture not just final outcomes but the full reasoning processes leading to them, offering richer insights for downstream applications.

Beyond annotation with human demonstrations, LLMs can independently enhance their annotation capabilities through search with feedback, a process that involves iterative refinement by learning from dynamic environment. Failed data points can be considered a classic form of feedback, serving as valuable feedback for the model to identify weaknesses and design targeted adjustments. By self-correcting erroneous samples and generating refined training data, LLMs engage in a cycle of self-improvement that strengthens both their understanding and reasoning [70]. Furthermore, LLMs can systematically analyze the causes of their errors, extracting key insights and encoding these as self-learned knowledge to guide future reasoning tasks [72]. This feedback-driven approach can also involve pairing failed trajectories with successful ones based on similarity, enabling contrastive learning strategies to refine the model's parameters. Through such iterative search and refinement mechanisms, LLMs not only address errors but also develop a more robust capacity for reasoning, enabling deeper generalization and adaptability across complex tasks [135].

3.3 LLM Automated Process Annotation

In complex reasoning tasks, each step of the model's output can significantly influence the final result, making it essential to label intermediate decisions as "correct," "incorrect," or assign an intermediate reward, namely process annotation. However, manually labeling these steps is costly and time-consuming. For example, Lightman *et al.* [75] inputs massive manual efforts to produce a large-scale process annotation dataset, i.e. PRM800K, which satisfies the requirement in training an effective process reward model (PRM) and greatly enhances the reasoning capability of LLMs. Therefore, automated methods are increasingly needed to efficient process annotation, ensuring scalability and cost-effectiveness. Initial automated approaches hire external stronger LLMs to annotate the intermediate process generated by smaller LLMs. Furthermore, Monte Carlo based method reduces the reliance on external stronger LLMs, and can use weaker LLMs to complete data annotation and thereby train stronger LLMs via a self-reinforced manner.

Annotation with stronger LLM: As a straightforward automated labeling method, Luo *et al.* [84] design to utilize a more powerful external model to annotate the intermediate results of a generative model's inference process. Rather than relying on manual annotation, the method employs a pre-trained, high-performance model, like GPT series, to evaluate each generated step. By leveraging the capabilities of a stronger external model, this approach enhances both the accuracy and scalability of the labeling process, making it more feasible for large-scale tasks. However, the major limitation of this approach is its reliance on the highly capable external model, which means the performance of the labeling process is ultimately constrained by the capabilities of the external model used.

Annotation by Monte Carlo simulation: To reduce reliance on the powerful external models, Wang *et al.* [148] and Wang *et al.* [156] propose an improved method that avoids directly scoring the intermediate steps. Instead, their approaches use an external model to continue the reasoning for several steps from the given intermediate output and randomly repeat this simulation process multiple times. The quality of the intermediate step is then assessed based on the average outcome of these extended inferences. This Monte Carlo method has shown promising results in tasks such as mathematical problem solving and code generation.

Annotation by tree search simulation: The approach of using multiple-step Monte Carlo simulation with an external model to assess the quality of intermediate steps based on the average outcomes has become one of the most widely used methods for automated process annotation. To further enhance the efficiency of this method, Luo *et al.* [85] proposes an improvement by replacing the repeated Monte Carlo simulations with a Monte Carlo Tree Search (MCTS) strategy. In this improved method, multiple leaf nodes representing the final inference results are generated from the intermediate step using MCTS. The quality of the intermediate step is then evaluated based on the average outcomes of these leaf nodes. Compared to random repeated inferences, MCTS leverages tree search to improve the inference quality, while also allowing leaf nodes to share high-quality parent nodes, reducing computational overhead and increasing efficiency. This method has demonstrated superior performance in mathematical problem solving, outperforming human annotations.

One step forward from the MCTS-based simulation, Zhang *et al.* [183] introduces a self-refining mechanism into the process annotation. They leverage the obtained process annotations to train a process reward function (PRM), which in turn improves the performance of the large language model (LLM). The refined LLM is then used to repeat the MCTS-based simulation, generating higher-quality annotations. This iterative process, involving repeated cycles of improvement, results in progressively enhanced process annotations. This method has shown excellent performance across several tasks, including mathematical mathematical problem solving, questioning and answering, and multi-domain knowledge reasoning, demonstrating its effectiveness in continuously refining and improving the quality of the annotations through iterative enhancement.

4 Learning to Reason: from Supervised to Reinforcement Fine-tuning

While pre-trained models excel across various tasks, they often struggle with complex reasoning and aligning outputs with human expectations. Fine-tuning is crucial to address these limitations, refining a model's performance on specific tasks and enhancing its reasoning capabilities. Initially, supervised fine-tuning (SFT) is used, where models learn task-specific patterns from labeled datasets. However, as reasoning challenges grow, methods like reinforcement learning (RL) and Direct Preference Optimization (DPO) offer a more effective approach, using reward models to more efficiently align the model's output with human-like reasoning, fostering more coherent, responsible, and contextually aware outputs.

4.1 Optimizing Pre-trained LLM: Supervised Fine-tuning

Supervised Fine-Tuning is a learning technique that refines pre-trained models' capabilities for specific tasks or domains using labeled data, while retains model's understanding on pre-trained knowledge. While pre-training allows models to learn broad, general-purpose features from massive amounts of unstructured data, fine-tuning specializes the model by exposing it to smaller, task-specific datasets with clear input-output mappings.

SFT is a critical step in improving the reasoning ability of LLMs, enabling their application in downstream tasks by adapting them from general-purpose systems to domain-specific tools. For example, LLMs like GPT [111], BERT [30], and T5 [113] are pre-trained on vast amounts of text data using self-supervised learning, equipping them with broad language understanding and generation capabilities. However, their outputs are not always aligned with task-specific requirements. Without fine-tuning, LLMs tend to perform poorly on certain reasoning tasks, such as object counting [182], satellite understanding [91], and engineering questions answering [154]. Through SFT, we can partially address these challenges by refining the model's outputs based on labeled task-specific datasets.

However, the direct application of SFT may not fully explore the model's reasoning capabilities in the desired domains, particularly in tasks that require more complex decision-making or multi-step problem-solving. The introduction of CoT techniques [160] has revolutionized the SFT process, by explicitly training the model to generate intermediate reasoning steps before arriving at an answer. With CoT-based SFT, LLMs are encouraged to generate intermediate reasoning steps explicitly, thus enhancing their reasoning ability to tackle tasks that require more structured and organized thoughts. For instance, ReasonBert [29] shows that fine-tuning models with reasoning chains significantly enhances their performance on tasks such as math word problems and logical reasoning by incorporating step-by-step reasoning processes. Another key study [80] investigates how fine-tuning models

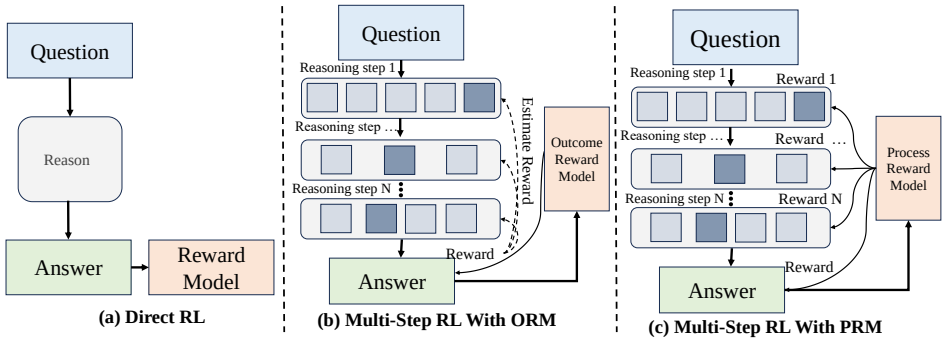

Figure 2: Reward models for Train-time Reinforcement of LLM Reasoning.

with reasoning improves their interpretability and reduces errors in complex decision-making scenarios by generating more transparent, stepwise thought processes. By fine-tuning with CoT, models not only improve their final answers but also enhance their ability to "think through" the problem, providing clearer insights into the model's reasoning process.

Despite the diverse methods and outstanding performance of SFT, it comes with several limitations. First, SFT heavily relies on high-quality labeled datasets, which can be expensive and time-consuming to curate, especially for niche domains or tasks requiring expert annotations. Second, SFT may lead to catastrophic forgetting, where the model loses some of its pre-trained general-purpose knowledge during the fine-tuning process, reducing its utility for tasks reasoning outside the fine-tuning domain. Finally, the computational cost of fine-tuning large-scale models can still be prohibitive, even with parameter-efficient approaches, posing challenges for organizations with limited resources. Addressing these limitations requires careful dataset curation, regularization techniques, and the exploration of alternative methods, such as prompt tuning or multi-task fine-tuning, to balance task specialization and generalization.

4.2 Optimizing Pre-trained LLM: Reinforcement Learning

Due to the high reliance on expensive, high-quality labeled datasets, and high computational costs of SFT, reinforcement learning has emerged as a powerful alternative framework for training models to master reasoning processes. Unlike supervised learning, RL enables models to learn through trial and error reward signals, discovering optimal strategies for achieving specific objectives. As shown in Figure 2 (a), the model takes action based on its current state and receives feedback in the form of a reward signal. This feedback guides the model to update its parameters over time, optimizing for cumulative rewards.

Classic reinforcement learning. RL has become a critical step in the development of LLMs. In RL framework, the parameters of LLMs are updated based on the rewards for their actions. Specifically, the value function or Q-function is updated based on the feedback of reward model, attributing the credit for an action's outcome entirely to its immediate effect. This approach simplifies the framework, making it conceptually straightforward while enhancing the model's ability to respond effectively. Two key methods currently dominate RL training for LLMs: Reinforcement Learning from Human Feedback (RLHF) and Reinforcement Learning from AI Feedback (RLAIF).

Ouyang *et al.* [104] use RLHF to align LLMs with human intent. Besides, by fine-tuning GPT-3 on human-labeled demonstrations and rank comparisons, they develop a reward model predicting human annotator's preferences. It effectively aligns trained LLMs with human preferences, outperforming GPT-3 in reasoning and instruction-following despite being smaller. Bai *et al.* [8] also leverage RLHF to create helpful and harmless language models. Following a Helpful, Honest, and Harmless framework, they fine-tune a base model, train a preference model with rejection sampling, and iteratively refine it with human feedback. This process produces AI assistants that excel in NLP tasks and demonstrate strong ethical reasoning.

To reduce the reliance on large human-labeled datasets, Bai *et al.* [9] propose Constitutional AI, a framework for training AI assistants to be helpful and harmless using principles instead of expensive human feedback. The process includes two phases: supervised learning and RLAIF. In the super-

vised phase, the model critiques and refines its outputs based on constitutional principles, creating a fine-tuning dataset. In the RLAIF phase, the model generates self-assessments to guide training, bypassing the need for human-labeled data on harmfulness. Ramamurthy *et al.* [114] focus on using RL to align LLMs with human preferences. They introduce RL4LMs, a library for RL-based fine-tuning, and the GRUE benchmark, which evaluates models using reward functions reflecting human preferences. To address training challenges, they propose the natural language policy optimization algorithm, which stabilizes training by constraining token sampling. This work provides a strong foundation for integrating RL into LLM fine-tuning for improved alignment and performance.

Direct preference optimization Classic RL methods rely on training a reward model to score outputs based on human preferences. While DPO streamlines this process by directly leveraging preference data without requiring an explicit reward model. Instead of optimizing a complex reward function, DPO uses pairwise preference comparisons, *i.e.,* data indicating which of two outputs is preferred by humans. This direct approach simplifies the learning pipeline while preserving the alignment benefits of RL-based methods, which is often simpler and more effective. Rafailov *et al.* [112] introduce DPO, a novel framework for aligning language models, which directly optimizes the policy to align with human preferences through a simple classification loss. By parameterizing the reward model to derive an optimal policy in closed form, DPO eliminates the need for sampling and extensive hyperparameter tuning during fine-tuning. Experiments show that DPO matches or surpasses RLHF methods like PPO in tasks such as sentiment control, summarization, and dialogue generation, while being more stable, computationally efficient, and effective in producing reasoning outputs. Amini *et al.* [4] propose Direct Preference Optimization with an Offset (ODPO), an extension of DPO for aligning language models with human preferences. ODPO improves upon DPO by considering the degree of preference between responses rather than treating all preference pairs equally. It introduces an offset in the likelihood difference between preferred and dispreferred responses, proportional to their quality difference. This approach not only improves alignment but also strengthens the model's reasoning capability, particularly in tasks such as sentiment control, toxicity reduction, and summarization. Experiments demonstrate that ODPO achieves better alignment and responsible behavior, especially when preference data is limited.

In conclusion, the RL and DPO methods offer a straightforward and effective method for fostering reasoning ability in LLMs. By focusing on immediate rewards following each action, these methods also align models with human preferences. The emphasis on short-term feedback simplifies the learning process, avoiding the complexities of credit assignment across long sequences. This streamlined approach is particularly well-suited for real-time applications and tasks requiring clear, concise reasoning, ultimately strengthening the ability of LLMs to deliver coherent and ethical outcomes.

4.3 Enhancing Multi-step Reasoning with Outcome Reward Model

For complex reasoning tasks, such as mathematical problem-solving, LLMs need to perform multi-step reasoning like Chain-of-Thought to ultimately reach an accurate solution. In these tasks, the reward feedback is typically only available after all the reasoning steps are completed and the final solution is obtained. As shown in Figure 2 (b), this is known as the outcome reward model (ORM). In such cases, the key to improving the LLM's reasoning ability lies in distinguishing the correctness and importance of intermediate reasoning steps based on the outcome rewards.

Classic reinforcement learning. ReFT [143] applies the PPO [121] method from RLHF [104] to reasoning tasks. Based on the outcome reward model, the value function in PPO is able to infer the contribution of intermediate reasoning steps. Compared to supervised fine-tuning, ReFT is capable of learning more diverse reasoning paths, exhibiting stronger generalization abilities in reasoning tasks. However, VinePPO [60] discovers that the value network trained with ORM in PPO exhibits significant bias when identifying the value of intermediate reasoning steps, a well-known challenge in RL called the credit assignment problem. To address this issue, VinePPO abandons the value network in PPO and instead employs a Monte Carlo sampling method to compute unbiased estimates of the value function. Experimental results demonstrate that VinePPO consistently outperforms typical PPO in mathematical reasoning tasks. Critical Plan Step Learning (CPL) is a method designed to enhance LLM's generalization in reasoning tasks by searching within high-level abstract plans [150]. CPL employs Monte Carlo Tree Search (MCTS) to explore different planning steps in multi-step reasoning tasks, and utilizes Step-APO to learn critical plan steps. This approach enables models

to learn more diverse reasoning paths, thereby improving generalization across various tasks. Subsequently, the model iteratively trains the policy and value models to further enhance performance. During each iteration, the policy model generates plan steps and final solutions, while the value model evaluates the quality of intermediate steps. Training data, generated by MCTS, is used to update both the policy and value models.

Direct preference optimization. In the task of mathematical reasoning, directly employing the DPO [112] method for preference optimization yields suboptimal results due to the presence of lengthy reasoning steps in the preference data. Amini et al. [4] introduced ODPO, which refines DPO by taking into account the degree of preference between responses instead of treating all preference pairs as equal. ODPO has achieved significant improvements over DPO in mathematical reasoning tasks.

In summary, the primary challenge of training based on outcome rewards lies in distinguishing the correctness and importance of intermediate reasoning steps. Current methods, primarily based on Monte Carlo sampling or Monte Carlo Tree Search, offer advantages in estimating the significance of these intermediate steps, though the computational cost during search remains high. Existing work has primarily focused on mathematical or other reasoning problems, where the final solutions can be easily verified. These methods can be extended to a wider range of reasoning tasks, including those where the solutions are difficult to validate. A potential approach is to learn a reward model based on human annotation data, and use it to judge the quality of the final solution. Based on the final score provided by the reward model, Monte Carlo sampling or search techniques can then be employed to further improve performance.

4.4 Enhancing Multi-step Reasoning with Process Reward Model

Process Reward Model (PRM) based Reinforcement Learning represents a significant advancement in LLM reasoning, emphasizing the evaluation of intermediate steps rather than solely focusing on end-state outcomes. As shown in Figure 2 (c), the reward of PRM is distributed across each reasoning step, rather than being concentrated at the final outcomes. By providing nuanced feedback throughout the reasoning trajectory, PRM enables models to optimize behavior with greater alignment to human preferences and complex task requirements. This approach is crucial for tasks that involve sequential decision-making, where intermediate steps or decisions are of significance for the final goal. We explore PRMs' evolution and highlight their role in improving reasoning by providing step-level rewards during complex tasks.

Classic reinforcement learning A series of recent works apply PRMs for mathematical or logistic reasoning, since a seminal work from OpenAI [75] has proven the importance of process reward. SELF-EXPLORE [55] uses PRMs to enhance mathematical reasoning by identifying and addressing "first pits", which are the initial incorrect steps in problem-solving. By rewarding steps that correct such errors, PRMs enable self-supervised fine-tuning without requiring extensive human annotations. This model achieves significant improvements in accuracy on mathematical benchmarks like GSM8K and MATH by leveraging step-level fine-grained feedback. MATH-SHEPHERD [149] introduces a PRM framework designed for step-by-step verification and reinforcement in mathematical reasoning tasks. By automating process supervision through MCTS-inspired methods, MATH-SHEPHERD eliminates the need for human annotations while ensuring high accuracy in multi-step problem-solving. PRMs are employed to reinforce logical progression and correctness, resulting in improved performance on benchmarks like GSM8K and MATH. DeepSeekMath integrates PRMs via Group Relative Policy Optimization (GRPO) [128], a RL algorithm that optimizes step-level rewards. PRMs are used to enhance mathematical reasoning and reasoning consistency across domains. By focusing on intermediate reasoning steps, DeepSeekMath achieves state-of-the-art performance on several benchmarks, showcasing the power of PRMs in mathematical domains. Scaling Automated Process Verifiers introduces Process Advantage Verifiers (PAVs), a PRM variant, to evaluate step-level progress in problem-solving [123]. PAVs use step-level supervision to improve the efficiency and accuracy of search algorithms and reinforcement learning. By focusing on steps that make meaningful progress toward a correct solution, PAVs enable substantial gains in sample efficiency, compute efficiency, and reasoning accuracy compared to outcome reward models. This demonstrates the importance of fine-grained process rewards in scaling LLM reasoning capabilities.

Interactive process reward models. PRMs are also applied to interactive tasks, such as conversation and multi-turn question answering. ArCHer employs a hierarchical RL approach using PRMs to

train agents for multi-turn, long-horizon tasks [198]. It implements a dual-layer system: a high-level value function evaluates utterance-level rewards, while a low-level PRM optimizes token-by-token generation within each turn. This hierarchical structure ensures more effective credit assignment and allows for nuanced training of language models to handle multi-turn interactions and reasoning tasks. The use of PRMs enables ArCHer to scale efficiently, achieving significant gains in sample efficiency and performance across agent tasks. Multi-turn Reinforcement Learning from Preference Human Feedback [126] integrates PRMs into multi-turn reinforcement learning to optimize long-term objectives with human feedback. The Multi-turn Preference Optimization (MTPO) algorithm compares entire multi-turn interactions to generate preference signals, where PRMs are used to assign step-by-step rewards. This enables LLM agents to align behavior with long-term goals, improving overall performance in dynamic, multi-turn tasks such as conversations and strategic decision-making.

Direct preference optimization. Several recent studies leverage MCTS to enable the optimization of multi-step reasoning tasks through Direct Preference Optimization [165, 17, 183, 16]. For instance, SVPO [17] employs MCTS to automatically annotate step-level preferences for multi-step reasoning tasks. From the perspective of learning to rank, it trains an explicit value model to replicate the behavior of an implicit reward model. Furthermore, SVPO integrates the explicit value model with DPO, where the value model not only aids the policy model in navigating more efficient reasoning paths but also guides preference learning. However, these works primarily focus on first collecting preference data or training a reward model and then performing policy optimization based on static data and the pre-trained reward model. Xie et al. [165] advanced these approaches by integrating data collection and policy preference optimization into an iterative process. This method can be considered as an online version of Direct Preference Optimization, where the updated policy is iteratively utilized to collect preferences through MCTS.

The evolution of multi-step RL techniques for LLMs reflects a transition from sparse outcome-based feedback to detailed process-oriented supervision. PRMs now stand as the centerpiece of the progression of LLM's reasoning ability, offering nuanced, step-level rewards that drive substantial improvements in reasoning tasks. Future research may focus on refining these models and expanding their applicability across diverse task domains.

4.5 Reinforcement Fine-tuning

Reinforcement fine-tuning (RFT) [101] is a technique recently proposed by OpenAI for customizing expert LLMs tailored to specific vertical domains. Currently, RFT remains part of a research program and the technical details have not been fully released. Available information suggests RFT utilizes a small amount of preference data provided by users along with a grader model to evaluate LLM's output. This technique enables iterative optimization of the LLM's multi-steps reasoning capabilities. As a result, RFT technique can enhance LLM's strategy of reasoning through similar problems in the optimized domains.

The grader model. RFT introduces the concept of a grader model to assess the outputs of LLMs. Considering that reinforcement learning training typically requires a reward model to provide feedback, the grader is likely analogous to a reward model, transforming textual inputs (*e.g.*, questions and answers) into scalar values of reasoning quality. This suggests that the grader could act as a reward model trained on user-provided preference data, potentially operating as either an outcome reward model or a process reward model [76].

Data efficiency. In OpenAI's live sessions, it was mentioned that RFT can enable learning in new domains with as few as dozens of user preference data. This suggests that RFT facilitates the exploration of diverse reasoning paths to address tasks based on limited preference data. Such an approach demonstrates remarkably high sample efficiency while mitigating the risk of overfitting [56].

Training stability. The stability of reinforcement learning training is notoriously difficult problem that presents significant challenges to its broader application. Variations in random seeds or adjustments to certain hyperparameters can greatly impact the training outcomes of RL. In the context of the RFT project, OpenAI has announced plans to make this technology available to the public via APIs, enabling users to fine-tune domain-specific expert models using their own data. This claim potentially indicates that RFT has achieved a level of stability sufficient for reliably fine-tuning language models using RL techniques.

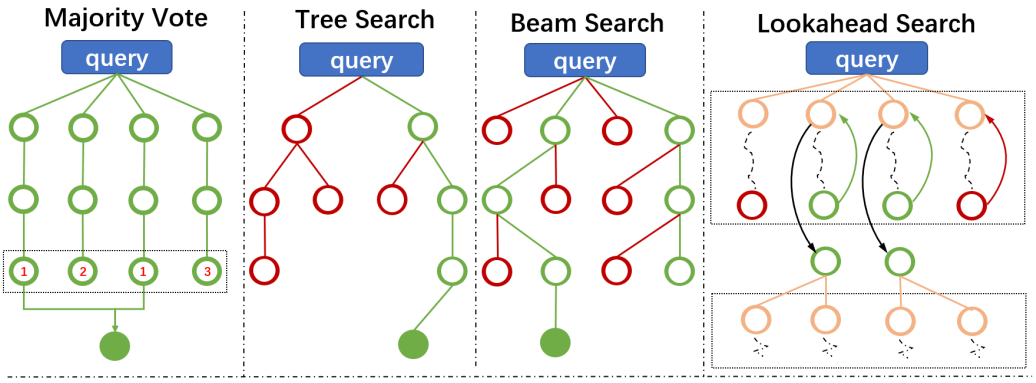

Figure 3: Diagrams of Different Search Algorithms for Test-time Reasoning Enhancement.

5 Test-time Scaling: from CoTs to PRM Guided Search

5.1 Elicit Deliberate Thinking with Prompts

Beyond train-time optimization through techniques such as reinforcement learning, researchers have discovered that test-time prompting techniques like Chain-of-Thought and Tree-of-Thoughts can further enhance LLMs' capabilities [160, 153]. While simply asking models for direct answers often yields suboptimal results, guiding them through explicit reasoning processes at test-time significantly improves their performance [62]. These prompting strategies have shown remarkable effectiveness across various domains, from mathematical reasoning to complex decision-making tasks [173, 196]. The emergence of structured prompting methods like ReAct and Least-to-Most Prompting has demonstrated that LLMs can benefit from explicit guidance in organizing their thought processes, leading to more reliable and interpretable outputs [189]. Although these approaches typically increase token consumption and computational overhead, they provide a compelling complement to train-time methods by enhancing LLMs' reasoning capabilities and solution accuracy without requiring model parameter modifications [172, 11]. This suggests a promising direction for improving LLM performance through sophisticated test-time interventions rather than solely relying on model architecture or training modifications.

5.2 PRM Guided Search

As previously mentioned, PRM marks a significant shift from sparse outcome-based feedback to detailed process-oriented supervision. Moreover importantly, PRM can also be utilized during the test-time phase, where it can further boosts the model's reasoning capabilities. OpenAI o1 series models stand as a prominent example of the advanced application of PRM. The new test-time scaling laws suggest that inference capabilities can be effectively enhanced by increasing test-time compute, providing a clear direction for the future development of LLMs. We introduce some methods applied during the inference phase, as shown in Figure 3. Red hollow circles represent the reasoning paths discarded during the algorithm's exploration process in the inference phase, green hollow circles signify the reasoning paths adopted during exploration, and green solid circles mark the endpoints of the reasoning paths once the correct answer is identified.

Majority Vote: Majority vote is one of the most straight-forward strategy to generate one final answer from dense test-time compute. During inference, each inference trace produces a prediction for a given input. The basic idea is to select the answer which most inference traces accord with. The predictions from all the models are then aggregated, and the class that appears the most (the "majority vote") is selected as the final output: $f* = argmax_f \sum_y \mathbb{I}_{final\_ans(y)=f}$, where $\mathbb{I}$ is the indicator function and y is each evaluation trace.

Tree Search [15]: Tree Search is a classic algorithm that systematically explores different choices by recursively constructing a search tree. It is commonly used in complex decision-making problems, such as board games and planning tasks. Monte Carlo Tree Search (MCTS) is one of the most widely used tree search methods. It consists of four main steps: Selection, Expansion, Simulation,

and Backpropagation. By progressively expanding the search space, MCTS incrementally improves decision-making. Tree Search has already been applied in some LLM inference tasks, achieving notable success. For instance, the Tree-of-Thoughts framework [172] enables LLMs to consider multiple reasoning paths structured as a tree. It incorporates self-evaluation to make thoughtful decisions, determining the optimal course of action for the next step. This approach significantly enhances the performance of model inference.

Beam Search [133]: Beam Search is an improved version of greedy search, commonly used in generation tasks to select the optimal output sequence. The main idea is to retain the top-K highest-scoring paths (referred to as beams) at each time step from all candidate paths for further expansion. Unlike greedy search, Beam Search maintains multiple candidate paths, thereby expanding the search space and improving generation quality. Beam Search is widely applied in LLM inference. For example, BART [71] uses Beam Search as its primary inference strategy, demonstrating its outstanding effectiveness in text generation tasks.

Lookahead Search [134]: Lookahead Search is another promising method that has the potential to significantly enhance LLM inference. It modifies the scoring mechanism at each step of Beam Search. Instead of selecting the best candidates based solely on the scores of the current step, Lookahead Search performs forward simulations by rolling out up to k steps ahead. If a solution endpoint is reached during the forward simulation, the process halts early. During Lookahead Search, a pre-trained and frozen Predictive Reward Model is used to score each step of the simulation. The cumulative scores derived from the PRM over the k-step simulation are then used to determine whether to retain or discard a beam branch. This strategy improves decision-making by incorporating more context into each evaluation step. Compared to beam search, lookahead search increases the depth of the exploration space, allowing for the judgment of current decision-making based on the results of more distant simulated decisions. However, it also increases the demand for computational resources which also leads to poor performance when computational resources are limited.

6 Path toward Large Reasoning Model

6.1 Development of OpenAI o1 Series

In September 2024, OpenAI released o1, a groundbreaking language model that represents a significant advancement in AI reasoning capabilities, particularly excelling in complex tasks like mathematics, coding, and scientific problem-solving. On December 20, 2024, OpenAI opened testing applications for o3, an upgraded version of o1 [102], which is considered to have literal doctorate-level intelligence [7]. These model achieve remarkable results across various challenging benchmarks, including scoring at the gold medal level in International Mathematics Olympiad [73] and matching PhD-level performance in physics, chemistry, and biology questions [48]. Extensive evaluations show distinct reasoning patterns of o1 series through systematic analysis of its basic reasoning capabilities. We list the key findings of existing research as follows:

Effective knowledge integration. Initial comprehensive evaluations [194] demonstrate o1's structured analytical approach and knowledge integration in fundamental problem solving tasks, achieving 83.3% success rate in competitive programming through step-by-step logical deduction, where the model demonstrates clear ability to use their knowledge to decompose complex problems and follow formal derivation processes. The model's structured understanding and interconnected knowledge application is further evidenced in specialized fields like radiology and chip design, where accurate diagnosis and complex circuit analysis require integration of multiple domain concepts. Systematic assessments [68] quantitatively validate this pattern, showing 150% of human-level performance in structured analytical thinking and computational reasoning tasks. This advantage is particularly prominent in scenarios requiring knowledge integration across domains, such as applying physical principles to biological systems or combining statistical methods with domain-specific constraints, indicating a fundamental capability in knowledge synthesis and application.

Systematic problem decomposition. o1 maintains consistent performance across tasks of varying complexity levels, showing systematic problem decomposition in handling increased difficulty. In mathematical reasoning, detailed studies [27] show its systematic problem decomposition approach, achieving near-perfect scores on the Dutch Mathematics B exam through structured solution steps.

The model demonstrates ability to identify key mathematical principles, construct formal proofs, and verify solution validity step by step. This consistency extends to more complex scenarios, as validated by research [26] on 105 science and math problems of increasing difficulty, where the model maintains high accuracy even as problem complexity increases in terms of both conceptual depth and computational requirements. In programming tasks, this pattern is further demonstrated through systematic debugging [52] on the QuixBugs benchmark, where o1 maintains consistent performance across bugs of varying complexity through a structured three-step approach: error identification, root cause analysis, and targeted correction.

Reliable and coherent reasoning in complex tasks. The model's reasoning adapt effectively across different problem types, always showing consistence of reasoning chains in various tasks. In planning tasks, PlanBench evaluations [144] demonstrate its systematic handling of both deterministic and probabilistic scenarios, showing significant improvement in constraint satisfaction and state management. The model shows particular strength in handling problems with incomplete information and dynamic constraints, maintaining consistent performance in both standard and rare task variants [94]. This adaptability indicates robust generalization capabilities across different problem formulations. Studies on complex planning [146] further show o1's ability to maintain reasoning coherence in long-horizon tasks, effectively managing extended dependency chains and context transitions. This is evidenced by its performance in multi-step planning problems where intermediate goals must be correctly sequenced and dependencies carefully managed, demonstrating advanced capabilities in temporal reasoning and causal understanding.

New Scaling Laws for Large Reasoning Models. Empirical studies demonstrate o1's distinctive scaling patterns in both training and inference phases. During training, the model's large-scale reinforcement learning algorithm teaches it to think productively using chain of thought in a highly data-efficient process [103]. Research [134] shows that through optimized test-time computation strategies, the model achieves significant performance improvements across various reasoning tasks. Comprehensive evaluations [194, 68] reveal that o1's reasoning capabilities can be effectively enhanced through advanced computation allocation during inference, particularly in complex problem-solving scenarios. The constraints on scaling this approach differ substantially from those of LLM pretraining, with performance consistently improving with more time spent thinking [103]. This is evidenced in programming tasks, where allowing 10,000 submissions per problem enables the model to achieve significantly better results, scoring above the gold medal threshold even without test-time selection strategies. The model's ability to effectively utilize additional computation resources during both training and inference suggests a fundamental advancement in reasoning architecture, demonstrating particular strength in scenarios where traditional approaches might require significantly larger model sizes.

6.2 Open-source Attempts of Large Reasoning Models

Open-source frameworks have also made substantial strides in developing advanced reasoning capabilities for LLMs. These frameworks serve as invaluable references for researchers and developers aiming to replicate or approximate the reasoning strengths of proprietary models like OpenAI's o1. In this section, we introduce four significant open-source efforts, each of which employs distinct strategies to enhance LLM reasoning (summarized in Table 2). By exploring their unique implementations, we aim to provide insights into the diverse methodologies used to reinforce reasoning abilities in LLMs.

Table 2: Open-source Attempts of Large Reasoning Models: A Contribution Point of View.

	Data Construction	Pre-Training	Post-Training	Test-time Improvement
OpenR[145]	MCTS	-	GRPO	Best-of-N
Rest-MCTS*[183]	MCTS (in loop)	-	SFT	Tree Search
Journey Learning[110]	-	-	SFT	Tree Search
LLaMA-Berry[185]	MCTS	-	DPO	Borda Count

The OpenR project[145]. The project claimed that it is the first open-source framework to explore the core methods of OpenAI's o1 model with reinforcement learning techniques. The core of OpenR replication is to construct step-by-step reasoning data, where the more precise and fine-grained

feedback is obtained instead of purely final answers. The automated data augmentation algorithm OmegaPRM [85] is adopted by selecting reasoning trajectories from a constructed search tree. Based on the augmented process data with supervision on each reasoning step, a process reward model is further trained in a supervised learning scheme, based on a pre-trained Qwen2.5-Math-7B-Instruct model [168]. The PRM can be directly deployed during test-time compute, integrated with either majority-vote, best-of-N or beam search methods. It can also be utilized to finetune LLM within the post-training stage using RL. Experiments are conducted to demonstrate the effectiveness of the PRM in test-time compute and post-training each.

Rest-MCTS*[183]. Rather than training PRM and the finetuned policy model separately, they instead integrate these two updates within one mutual self-training loop. Process reward as supervision for PRM training and reasoning traces for policy model training are collected in advance, based on a similarly designed MCTS algorithm. Then the iterative training process starts based on initial policy π and initial PRM values V_θ. The policy further iteratively perform the MCTS and generate solutions, while the values influence the tree search process. The updates of them complement each other iteratively.

The o1 Replication Journey project[110]. Rather than thoroughly considering improvement implementation in both stages, the project aims to replicate OpenAI's o1 model's reasoning abilities by focusing on comprehensive training strategies. It emphasizes a structured training diagram that incorporates trial-and-error, reflection, and backtracking to build deep causal reasoning. A core aspect of the project is data generation, with high-quality training examples designed to model complex reasoning paths. Using a journey learning method, o1 Replication Journey exposes the model to varied logical sequences and corrections, encouraging exploration and adaptability within the training stage. However, the o1 Replication Journey is less sophisticated at the inference stage, lacking advanced post-training techniques, which limits its adaptability during real-time reasoning. This focus on training over inference highlights its foundational approach compared to models with dynamic inference optimizations.

LLaMA-Berry[185]. The project directs its focus on optimizing reasoning abilities at the inference stage, leveraging the LLaMA-3.1-8B architecture to deliver more sophisticated real-time reasoning adjustments. It employs a unique pairwise optimization approach that combines Monte Carlo Tree Search with Self-Refine (SR-MCTS), allowing the model to dynamically explore and refine solution paths during inference. This configuration grants LLaMA-Berry a high level of adaptability, enabling it to tackle complex, open-ended reasoning tasks efficiently and flexibly. A key component of this framework is the Pairwise Preference Reward Model (PPRM), which evaluates solution paths in pairs, ensuring that high-quality reasoning paths are prioritized. LLaMA-Berry's Enhanced Borda Count (EBC) then consolidates these preference rankings to guide the model's decision-making, further enhancing its inference-stage sophistication. This robust architecture positions LLaMA-Berry as a leading example of inference-focused reinforcement, distinguishing it from O1 Replication Journey's training-centric approach.

These four open-source frameworks, not only demonstrate distinct implementation strategies for reinforced reasoning but also play an essential role in improving the understanding of OpenAI's o1 model. Together, they expand the range of techniques available to the open-source community, advancing the collective goal of developing sophisticated, transparent, and adaptable reasoning models that bring proprietary-level capabilities to publicly accessible systems.

7 Other Test-time Enhancing Techniques

In addition to the PRM-guided search, there are numerous other techniques devised to enhance LLM reasoning abilities with more test-time compute. These techniques refine reasoning result dynamically without modifying the model itself. Approaches such as verbal reinforcement search, memory-based reinforcement, and agentic system search, depicted in Figure 4, demonstrate that substantial reasoning improvements can be achieved with the off-the-shelf LLMs alone. A selection of representative works exploring these methods is summarized in Table 3. While these methods do not leverage PRM, they offer a foundation for future research to explore hybrid models for further advancing reasoning capabilities.

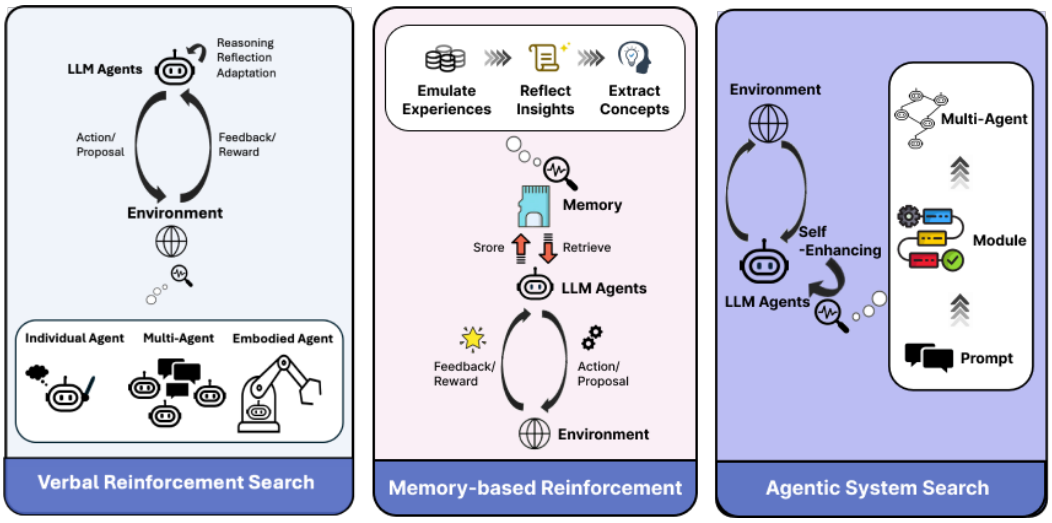

Figure 4: Typical training-free test-time enhancing methods: verbal reinforcement search, memory-based reinforcement, and agentic system search.

Table 3: A list of representative works of training-free test-time reinforcing.

Method	Category	Representative literature
Verbal Reinforcement Search	Individual Agent	Romera et al.[115], Shojaee et al.[130], Mysocki et al.[162],Ma et al.[88]
	Multi-Agent	Chen et al.[20],Zhou et al.[199], Le et al.[69] ,Yu et al.[176]
	Embodied Agent	Boiko et al.[13]
Memory-based Reinforcement	Experiential Learning	Zhang et al.[184],Gao et al. [39] Qian et al.[108]
	Reflective Learning	Shinn et al.[129],Sun et al.[138], Sun et al.[190]
	Concept Learning	Zhang et al.[188],Gao et al.[40] , Guan et al.[44]
Agentic System Search	Prompt Level	Madaan et al.[90], Fernando et al.[38] Yang et al.[169]
	Module Level	Shang et al.[125], Zhang et al.[186]
	Agent Level	Huot et al.[54],Zhuge et al.[200]

7.1 Verbal Reinforcement Search

Verbal Reinforcement Search (VRS) leverages the pre-trained reasoning and semantic capabilities of LLMs to explore and optimize solution spaces. Unlike traditional reinforcement learning or training-intensive approaches, VRS operates purely through test-time inference, using iterative feedback loops to refine solutions without requiring additional training. By drawing on the semantic knowledge encoded in LLMs and their ability to follow complex instructions, VRS provides a versatile approach for navigating diverse problem spaces. This inference-driven framework finds application across individual agents, multi-agent systems, and embodied agents, supporting a wide range of tasks, including programmatic optimization, collaborative decision-making, and interactions in real-world settings. This section analyzes VRS through these three key aspects, delving into the methodologies and unique insights presented within each category.

In **individual agent settings**, VRS relies on iterative reasoning and feedback mechanisms to refine solutions within structured problem spaces. This approach is well-suited for tasks like mathemati-

cal optimization, symbolic reasoning, and hypothesis-driven discovery, where systematic refinement significantly improves problem-solving outcomes. Research on mathematical discovery illustrates how VRS reshapes the problem-solving process into a dynamic iterative cycle. For example, studies on combinatorial problems, including the cap set and online bin-packing, highlight how programmatic solutions evolve through feedback-driven evaluation [115]. Similarly, symbolic regression research treats equations as dynamic constructs, iteratively generating, evaluating, and optimizing mathematical expressions [130]. These approaches show how VRS navigates constrained spaces, surpassing traditional optimization techniques in efficiency and accuracy. In scientific discovery, VRS has shown its utility in integrating reasoning with empirical data and simulations. Researchers have developed systems for biomedical hypothesis refinement by synthesizing diverse data sources. For instance, applications in oncology use iterative synthesis to address the complexity of multi-scale data [162]. In physical sciences, VRS is used to refine hypotheses through simulation feedback, advancing fields like molecular design and the discovery of physical laws [88]. These findings emphasize the role of VRS in connecting abstract reasoning with real-world validation, supporting tasks that are both data-intensive and hypothesis-driven. Reflective processes in heuristic optimization further showcase the flexibility of VRS. For example, researchers have explored the iterative generation and evaluation of strategies for solving combinatorial problems [174]. This approach focuses on creating adaptive hyper-heuristics that generalize effectively across different domains by continuously refining solutions through feedback cycles. Overall, VRS applies iterative reasoning and feedback to connect abstract problem-solving with real-world applications, addressing challenges in mathematics, science, and optimization with precision and adaptability.

In **multi-agent systems**, VRS facilitates collaboration between LLM-based agents through natural language communication. These systems leverage shared reasoning and iterative refinement to tackle complex solution spaces, allowing agents to exchange insights and achieve common goals. Meta-structure discovery in heterogeneous information networks (HINs) exemplifies how VRS is applied in multi-agent contexts. Recent research has combined LLM reasoning with evolutionary optimization to refine meta-structures, enhancing their explainability and predictive accuracy [20]. Similarly, in socioeconomic prediction, multi-agent systems integrate knowledge graphs and meta-path reasoning to extract cross-task insights for applications like population estimation and economic activity prediction. This approach facilitates collaboration between LLM agents and improves performance in multi-task environments [199]. Causal discovery also benefits from multi-agent frameworks enabled by VRS. For example, systems using LLMs as reasoning agents collaboratively debate and propose causal relationships. By incorporating statistical methods and natural language interactions, these frameworks generate accurate causal graphs while addressing ambiguities in causal relationships [69]. In financial decision-making, VRS enhances hierarchical collaboration. The FINCON framework employs a manager-analyst system to refine financial strategies using conceptual verbal reinforcement. By minimizing redundant communication and improving strategy refinement, FINCON demonstrates the utility of VRS in optimizing financial decision-making processes [176]. With iterative refinement and shared reasoning, VRS supports multi-agent systems in tackling complex tasks such as meta-structure refinement, socioeconomic prediction, and financial decision-making.

In **embodied agent settings**, VRS is used to address real-world tasks by integrating reasoning with physical interactions, supporting activities such as experimental planning and execution in laboratory settings. These systems extend VRS into dynamic environments, combining semantic reasoning with practical experimentation. For example, autonomous chemical research has demonstrated the use of LLM-powered systems to independently design, execute, and refine experiments [13]. These agents integrate tools such as robotic liquid handlers, spectrometry devices, and web-based research modules to perform tasks like reaction optimization and compound synthesis. One application involves optimizing palladium-catalyzed cross-coupling reactions, where the system uses natural language prompts to determine conditions, calculate stoichiometries, and autonomously execute experiments. When faced with errors, such as incorrect module calls, the system revises its approach by referencing documentation and iterating on the task. This iterative process demonstrates how VRS supports adaptability and precision in experimental workflows. By combining reasoning and real-time feedback, embodied agents illustrate the capability of VRS to refine and optimize complex processes in dynamic environments. These systems reduce human intervention while accelerating scientific discovery, making them a valuable tool for real-world experimentation and innovation.

In general, previous studies have showcased the adaptability and effectiveness of VRS across individual agents, multi-agent systems, and embodied agents. Leveraging the semantic reasoning and iterative feedback capabilities of LLMs, VRS tackles a wide range of tasks without the need for additional training. From structured optimization in mathematical and scientific contexts to collaborative exploration in multi-agent frameworks, and dynamic experimentation in real-world applications, VRS provides a unified approach to problem-solving. VRS as a versatile framework, capable of addressing complex challenges across both computational and physical domains while driving advancements in diverse fields.

7.2 Memory-based Reinforcement

When applied to open-ended tasks such as creative writing, complex logical reasoning, and open-world gaming, the solution space tends to expand dramatically, often becoming unbounded or ill-defined. These tasks typically require continuous interaction with the environment to acquire relevant information, making simple solution space searches inefficient. To address these challenges, some studies incorporate an external memory module for LLM agents. This module stores information such as observations, successful and failed actions from past trials. Agents explore their environments iteratively, using memory as a foundation for verbal reinforcement learning. Through this process, they summarize experience, extract interpretable high-level insights of the solution space, and refine their actions in subsequent trials, thereby improving inference performance. These studies not only focus on exploring the external solution space but also emphasize the intrinsic ability of LLM agents to develop an understanding of the solution space from memory. As the agents accumulate memory through environmental exploration, their capabilities are progressively reinforced and generalized to unseen tasks. Specifically, we classify the studies in this area into the following three categories.

Experiential learning. Methods in this category encourage LLM agents to simply emulate favorable experiences stored in memory while avoiding unfavorable ones. REMEMBERER [184] introduces a semi-parametric RL-LLM agent that records past observation-action pairs in memory and uses a traditional off-policy Q-learning algorithm to dynamically maintain and update the Q-value (expected future reward) of each observation-action pair. When faced with a new task, the agent retrieves relevant actions with the highest and lowest Q-values from memory, incorporating these as encouraged and discouraged examples in the prompt. Memory Sharing [39] leverages concepts from multi-agent reinforcement learning to enhance learning efficiency. Multiple agents execute tasks concurrently in a shared environment and contribute high-quality prompt-answer pairs to a collective memory pool. Each agent can retrieve the most relevant examples from this pool to facilitate few-shot learning. Similarly, Experiential Co-Learning [108] employs a multi-agent framework in which Instructor and Assistant agents alternately provide instructions and solutions during multi-step code generation. This dynamic exchange helps extract shortcuts to reduce redundancy and prevent repetitive mistakes. When encountering new tasks, these agents retrieve relevant memories alternately to improve in-context learning.

Reflective learning. Although using memory as few-shot exemplars is straightforwardly effective, this approach does not fully exploit the semantic comprehension capabilities of LLMs. Some studies argue that LLM agents should reflect directly on the successes and failures stored in memory to summarize underlying causes explicitly, adopting these insights as guidelines. Reflexion [129] is a pioneering effort in this area, reflecting on the reasons behind success or failure semantically based on task feedback signals. It integrates reflective text and past trajectories into prompts to enhance decision-making in subsequent trials. ExpeL [190] combines imitation and reflection by retrieving the most relevant successful experiences from memory, summarizing patterns of successful trajectories, and identifying insights from comparisons of success-failure pairs. RAHL [138], inspired by hierarchical reinforcement learning, organizes memory into goal modules and sub-task modules, enabling reflection and experience summarization at different levels. For new tasks, it retrieves relevant experience to formulate high-level goals and low-level sub-tasks separately.

Concept learning. Explicit reflection significantly enhances the inference capabilities of LLMs. Building on this, some studies aim to enable LLM agents to develop generalized "concepts" that transcend specific tasks, facilitating a broader understanding of the environment and tasks. This generalization helps agents internalize cognitive abilities from memory and evolve continuously as memory grows. Agent-Pro [188], for example, enables agents to establish beliefs about them-

selves and their environments in card-based gaming. Instead of reflecting on individual actions, it evaluates the rationality and consistency of these beliefs, iteratively refining strategies. Similarly, Richelieu [44] equips agents with an understanding of the environment in military strategy games. It retrieves the most relevant states from memory to formulate plans and assess feasibility. By employing self-play, it collects experience autonomously, assuming the roles of all players to advance its knowledge. Self-Evolving GPT [40], inspired by human memory mechanisms, designs a memory-based autonomous learning framework for LLMs. It categorizes tasks to determine relevant memory retrievals and identifies differences between stored memories and the current task to extract shared general experience. Additionally, it generates unseen tasks for practice, consolidating its knowledge based on memory retrieval outcomes.

7.3 Agentic System Search

The design of agentic systems plays a crucial role in harnessing the power of LLMs for many downstream tasks. An important branch of test-time enhancing techniques is to leverage LLMs to search the agentic systems. Studies in this area can be classified into three levels of search: prompt level, module level, and agent level. Please note that this approach does not aim to directly search the solution space, rather it leverages the empirical data to optimize the agentic system itself, which is similar to a meta-learning problem. We summarize the related works in this area as follows.

Prompt level. The process of "verifying and correcting" improves the prompts by iteratively integrating useful feedback experience. The verification signal can come from external feedback [43], LLM's self evaluation [90], and other sources. On the other hand, prompts themselves are also worth searching and optimizing. Automated prompt engineering, such as evolutionary prompt optimization [38] and meta prompt iterations [169], can achieve better results than manual prompts, but it also introduces more token consumption.

Module level. Agentsquare [125] proposes to use LLM to search the modular design of agentic system, where the modules are essentially blocks of prompts that have specific functions of Planning, Reasoning, Tool use, and Memory. The basic units of these agentic modules have standard IO interface that allows them to collaborate well with each other. The advantage of module level search is it allows new agents to easily reused the classic agent design, such as CoT and ToT, through recombination of modules. Besides, Aflow[186] connects the different calling nodes of LLM through edges represented by code. In addition to the search method, it is necessary to evaluate the performance of the searched agents. The function used to evaluate the performance of agents can also be driven by LLMs to improve search efficiency while closely matching their actual performance.

Agent level. ADAS proposes to leverage LLMs to search the entire agentic systems defined in python code space [53]. Besides, multi-agent systems make decisions and achieve goals in a shared environment. In multi-agent level search, key aspects include agent creation, environmental perception, action, interaction, and system evolution. The search for multi-agent systems has achieved good results in downstream tasks such as long story creation [54]. The unified search and optimization mechanism for multi-agent systems is currently being explored. GPTSwarm [200] enhances the collaborative capability of agents through graph optimization.

Agentic System Search provides agents with the ability to self-improve, enabling them to optimize themselves to enhance their reasoning abilities without the need to make changes to the LLM structure. The above three levels of search have vast search spaces. The common challenge faced by these three search levels is to improve search efficiency, reduce search costs, and ensure automation while ensuring search rationality.

7.4 Summary

The test-time enhancing techniques review in this section currently are not incorporated in the implementations of large reasoning models. However, they have huge potential to further boost the reasoning capacities of LLMs through more comprehensive test-time "thinking", facilitating LLMs to strategically reason across the solution space, leverage past experiences and dynamically optimize agentic workflows. Therefore, training LLMs to master these test-time techniques represents a promising future research direction, with the potential to elevate LLMs from "reasoners" to fully functional "agents."

8 Evaluation Benchmarks

Designing a robust benchmark is important to document the improvement LLM's capabilities. It also plays a crucial role in selecting the promising research direction for further advancement. In this section, we systemically review the popular benchmarks for LLM reasoning, which are summarized with the taxonomy in Figure 5. Here, we discuss these benchmarks as follows.

8.1 Math Problems

Mathematical reasoning has become a crucial testbed for evaluating LLM's reasoning capabilities. The landscape of mathematical reasoning benchmarks spans from elementary arithmetic to advanced university-level mathematics, providing systematic ways to assess different aspects of mathematical understanding and problem-solving abilities.

In the realm of **mathematical word problems** (MWP), benchmarks progress from fundamental arithmetic operations to increasingly complex problem-solving scenarios. At the basic level, datasets like MATH-401 [177] evaluate pure arithmetic capabilities through 401 carefully structured expressions, while MultiArith [116] and AddSub [51] assess the ability to translate simple word problems into mathematical operations (such as addition or subtraction). Moving to elementary and high school levels, comprehensive datasets such as GSM8K [24] and MATH [50] present more sophisticated multi-step reasoning challenges, with GSM8K offering 8.5K grade school problems and MATH providing 12.5K problems across various mathematical domains with graduated difficulty levels.

The evaluation of **advanced mathematical capabilities** is primarily conducted through competition and specialized test datasets. Collections like CHAMP [92] and ARB [5] present competition-level problems that require sophisticated problem-solving strategies, while MATHQA [5] incorporates standardized test questions from GRE and GMAT examinations. At the highest level, datasets such as FIMO [78] challenge models with International Mathematical Olympiad problems, testing the limits of automated mathematical reasoning.

Geometric reasoning represents a distinct category requiring spatial understanding and formal mathematical proofs. Datasets like Geometry3K [82] and GEOQA [19] provide specialized geometric problems, while UniGEO [18] offers a unified framework for geometric reasoning tasks focusing on calculation and proving. These benchmarks are particularly valuable in assessing models' abilities to connect visual and mathematical reasoning.

The field of **theorem proving** and formal mathematics has evolved to include rigorous evaluation frameworks. MINIF2F [193] and LeanDojo [170] focus on formal mathematical proofs related to Lean Theorem, while THEOREMQA-MATH [23] examines understanding of mathematical theorems. Specialized datasets like TRIGO [166] and PISA [57] address specific areas of mathematical reasoning, such as trigonometry and formal proof systems.

Lastly, **cross-modal mathematical reasoning** has emerged as a crucial area, reflecting the diverse ways mathematical problems are presented in real-world scenarios. MATHVISTA [81] and CHARTQA [93] evaluate visual mathematical reasoning through diagrams and charts, while TABMWP [83] and MultiHiertt [192] assess the ability to reason with tabular and textual data. SciBench [151] bridges the gap between pure mathematics and scientific applications, testing mathematical reasoning in broader scientific contexts.

8.2 Logical Problems

Building upon mathematical reasoning capabilities, the ability to engage in systematic logical reasoning stands as another fundamental criterion for evaluating LLM's cognitive abilities. While mathematical reasoning focuses on quantitative operations and formal proofs, logical reasoning encompasses the broader capacity to draw valid conclusions, recognize patterns, and generate rational explanations across diverse contexts. According to Luo et al. [86], logical reasoning can be classified into three main types: Deductive, Inductive, and Abductive reasoning. Each type represents a distinct cognitive process essential for comprehensive logical analysis while maintaining interconnections in cognitive assessment.

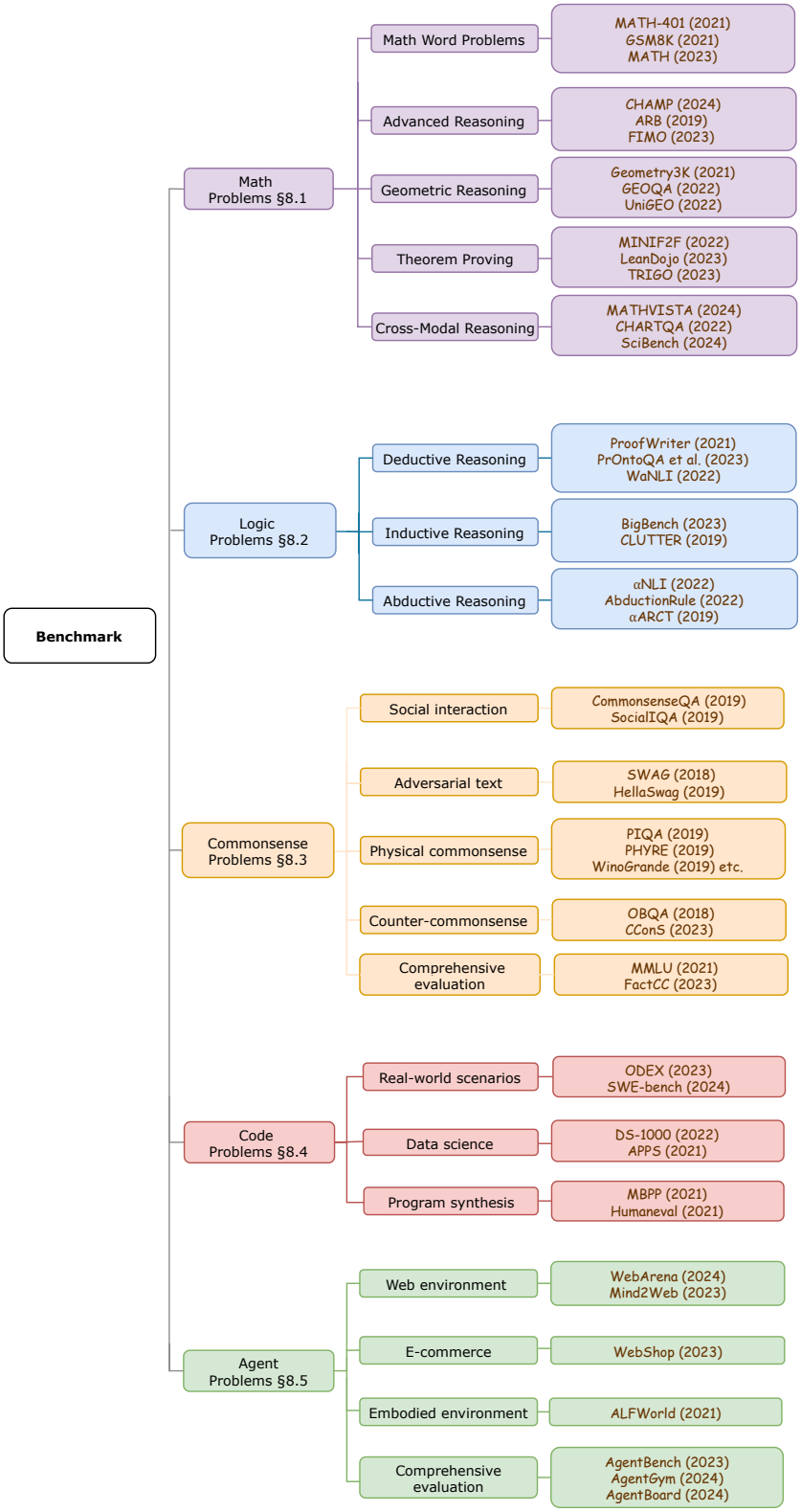

Figure 5: A Taxonomy for LLM Reasoning Benchmarks.

Deductive reasoning also known as premise-based reasoning, involves deriving specific conclusions from general principles with absolute certainty. For example, given a set of rules about relationships between entities, a model must determine what specific relationships must be true. ProofWriter [140] exemplifies this category, requiring models to construct explicit logical derivations from given premises. Other benchmarks, such as FOLIO [46] and PrOntoQA [119] evaluates first-order logic reasoning in natural contexts, WaNLI [77] have introduced increasingly sophisticated evaluation criteria with 107,885 examples.

Inductive reasoning emphasizes pattern recognition and generalization from specific observations to broader principles [47]. This involves identifying underlying regularities and extending them to new situations, dealing with probability rather than certainty. BigBench [136] with numerous specialized components that examine advanced pattern inference capabilities. Also, CLUTTR [132] benchmark series evaluates this capability through relationship patterns of varying complexity.

Abductive reasoning, also termed explanatory reasoning, refers to the process of forming the most likely explanation for a set of observations or facts, even though the conclusion is not guaranteed to be certain [34]. This type of reasoning tests how models handle scenarios with incomplete information by generating reasonable explanations. The αNLI [99] benchmark implements this through narrative completion tasks, where models must select the most likely explanation for given situations. The AbductionRule [175] series offers structured evaluation frameworks across different domains, with specific variants for animal-related and person-related reasoning scenarios. αARCT [100] specifically examines the ability to select and justify plausible explanations and argument comprehension.

8.3 Commonsense Problems

Commonsense reasoning remains a significant challenge in NLP, as it aims to evaluate LLM's ability to understand and apply everyday commonsense knowledge. There are various benchmarks targeting different dimensions of commonsense reasoning tasks. For instance, CommonsenseQA [141] requires models to answer reasoning questions grounded in commonsense knowledge bases.

SocialIQA [118] focus on **social interaction** commonsense reasoning, which revolves around causal reasoning in social scenarios. In contrast, datasets like SWAG [178] and HellaSwag [179] introduce **adversarial text** reasoning tasks, where models must predict the most plausible continuation of events based on contextual clues, thereby increasing task complexity. For **physical commonsense** reasoning, benchmarks such as PIQA [12] and PHYRE [10] concentrate on evaluating models' understanding of everyday physical tasks and interactive reasoning scenarios. PIQA primarily uses question-answering tasks, while PHYRE emphasizes interactive physical simulations. Similarly, WinoGrande [117] builds upon the Winograd Schema Challenge by introducing a larger-scale dataset and mo re complex disambiguation tasks to test semantic understanding and conference resolution capabilities.

Other works, such as OBQA [95] and CConS [63], explore model performance in **counter-commonsense** contexts, highlighting the challenges faced by current models in implicit reasoning and background knowledge utilization. More recently, **comprehensive benchmarks** like MMLU [49] and critical studies such as FactCC [66] have further analyzed LLM's commonsense reasoning and factual reasoning. These benchmarks offer valuable perspectives on the generalization abilities of language models and serve as valuable tools for evaluating and improving their performance across diverse commonsense reasoning tasks.

8.4 Coding Problem

The development of code generation benchmarks has been instrumental in evaluating the reasoning capabilities of LLMs in programming tasks. These benchmarks assess models' proficiency in generating accurate, efficient, and reliable code across various domains. For example, ODEX [155] introduces an execution-driven evaluation framework for open-domain code generation, emphasizing the importance of running generated code to verify its correctness and functionality.

As for the **real-world scenarios**, SWE-bench [58] focuses on real GitHub issues, challenging models to resolve practical software engineering problems. In the realm of **data science**, DS-1000 [67] presents a benchmark featuring authentic and dependable data science code generation tasks, en-

abling the assessment of models' abilities to handle complex data manipulations and analyses. Besides, the APPS benchmark [49] measures coding challenge competence by evaluating models on a diverse set of programming problems, reflecting the challenges encountered in competitive programming and technical interviews.

MBPP [6] focuses on **program synthesis** problems, assessing models' abilities to generate correct and efficient code based on given specifications, thereby contributing to the understanding of LLMs' capabilities in automated code generation. The HumanEval [21] evaluates LLMs trained on code by providing a set of Python programming problems, each provided with a function definition and accompanying documentation, requiring models to generate correct and functional code solutions.

8.5 Agent Problems

The emergence of agent-based benchmarks has revolutionized our ability to assess LLMs as independent agents within interactive environments. These sophisticated evaluation frameworks assess crucial capabilities including decision-making, reasoning, and environmental interaction across diverse scenarios.

WebArena [197] provides a practical **web environment** for building and testing autonomous agents, enabling the evaluation of LLMs' web navigation and interaction skills. Similarly, Mind2Web [28] aims to develop generalist agents capable of operating across diverse web tasks, emphasizing adaptability in dynamic online environments.

In **E-commerce** settings, WebShop [171] introduces a platform for scalable real-world web interaction, focusing on grounded language agents that can perform tasks such as online shopping, thereby testing models' practical application abilities. To bridge textual and embodied environments, ALFWorld [131] aligns text-based inputs with interactive learning scenarios, facilitating the assessment of models' abilities to transfer knowledge between different modalities.

Comprehensive evaluation frameworks like AgentBench [79] and AgentGym [164] have been developed to systematically assess LLMs functioning as agents. AgentBench includes diverse environments to assess reasoning and decision-making skills, while AgentGym focuses on evolving LLM-based agents across diverse settings, emphasizing adaptability and learning efficiency. Furthermore, AgentBoard [87] offers an analytical platform for evaluating multi-turn LLM agents, providing insights into their performance over extended interactions and highlighting areas for improvement in sustained reasoning tasks.

9 Discussion

9.1 Inspirations from the recent advances

Scaling law of post-training phases. The inspiration from OpenAI o1 series leads to a new understanding of the pre-training/post-training/inference stages. Particularly, it involves the introduction of self-play reinforcement learning and the process reward learning of high-quality Chain-of-Thought labeled data during the post-training stage. Further, it extends to the scaling law in the post-training stage, which provides inspiration for the difficulties in the further development of the Scaling Law in the training stage. As we know, the scaling law in the pre-training and training stages has led to the success of popular LLMs, with the huge investment of training data and computation resources. However, it now reaches the bottleneck, and thus, the scaling law of post-training phases may be the driving force for the next period of development of large language models. Furthermore, LLM-driven agents [163] has also shown great potential with carefully-designed workflow even if the reasoning abilities have not been reinforced. Therefore, it is still an open question whether there will also be a similar scaling law regarding resource consumption and performance in LLM agents, which could be the potential to further enhance LLM in real-world applications. Lastly, there may be a relationship between the currently exhibited test-time scaling law and the model's ability to follow instructions; that is, it must have a sufficiently strong instruction following ability to demonstrate test-time scaling laws. For example, the success of verbal reinforcement search techniques require the LLMs to have the basic ability to follow instructions. Thus, if the LLMs cannot accurately follow instructions, the complicated post-training techniques might not work properly.

Generating high-quality data through searching. Both the technical ideas of OpenAI o1 series disclosed by its core technical personnel and the open-source works attempting to reproduce OpenAI o1 currently regard the generation of high-quality data (including CoT data) as the key point, although different approaches such as Monte Carlo Tree Search, LLM generation, and others have been adopted. That is, the development of large reasoning models reaches a stage where high-quality process reward data is more important than general pre-training data size. Similarly, as discussed above, it may inspire us to refer to these related approaches in LLM agents as well, first to conduct high-quality data generation and then enhance the learning of slow reasoning as well as the acquisition of capabilities.

9.2 Slow-thinking and reasoning

Even if the breakthrough of OpenAI o1 series at the engineering level remains unknown, theoretically and technically, its breakthrough currently seems to mainly lie in the post-training learning of slow-thinking data. Also, the human cognitive science of "System 1 + System 2" has been repeatedly mentioned, but the ideas on how to implement it based on large models have been constantly updated, mainly still staying at the stage of drawing on the concept of slow thinking. That is, the mechanism named "System 1 + System 2" of human brains has guided the design of LLMs, but the guidance is still very limited. In other words, the imitation of the human brain is only about system-level design rather than very detailed techniques. The complex mechanisms of human slow thinking and their benefits still show high potential to support the next-level reasoning abilities of LLMs. To accomplish it, the domain knowledge of slow thinking should be used in the related designs such as reasoning data generation, reward functions, learning process, etc.

There has been no truly significant and representative work on the theoretical analysis of slow-thinking of LLMs up to now. The generative artificial intelligence is so mysterious that understanding LLMs also requires some tricks or special techniques such as new metrics for understanding hallucination of LLM [37]. To understand the slow-reasoning abilities, we may need to also step into the theoretical analysis. Taking the two different versions, OpenAI o1 Preview and OpenAI o1 Mini, as examples, the main difference lies in the cost and depth of thinking in the CoT inference stage, yet they show significant differences in tasks such as text generation, code generation, and mathematical problem-solving. The special characteristics of reasoning shown by LLMs also inspire us to design task-adaptive usage and applications. Specifically, it may support more interesting insights to link the reasoning mechanism and the performance in different tasks.

9.3 Downstream applications and open problems

As pointed out throughout this paper, the progress of reasoning enhancement technologies is rapid. The reasoning abilities are not limited to the tasks in these popular benchmark tasks, but also in more general tasks in downstream applications. For example, the FunSearch work [115] has shown the general ability for tasks difficult to provide solutions, but verification is fast. There could be a bundle of tasks with similar features in various domains, such as urban planning, logistics scheduling, etc. An interesting question is whether there may be many complementary problems in current research, which is difficult to verify, but the reasoning process is easier. It may be possible to further verify the quality of some answers by combining LLMs and external evaluators, or we can use these answers with evaluated scores to train the reward model.

10 Conclusion

The recent evolution of LLMs has significantly advanced their human-like reasoning capabilities. The innovations of introducing concepts like "thought" as intermediate steps, leveraging reinforcement learning techniques for train-time scaling, and using search algorithms for test-time scaling have laid the groundwork for large reasoning models, which can address increasingly complex cognitive tasks, as exemplified by OpenAI's o1 series. The ongoing progress in this field promises to reshape both our understanding of language and the application of AI in solving real-world problems.

References

[1] Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[4] Afra Amini, Tim Vieira, and Ryan Cotterell. Direct preference optimization with an offset. *arXiv preprint arXiv:2402.10571*, 2024.

[5] Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms, 2019.

[6] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[7] AXIOS. Openai's new o3 model freaks out computer science majors. 2025.

[8] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[9] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[10] Anton Bakhtin, Laurens van der Maaten, Justin Johnson, Laura Gustafson, and Ross Girshick. Phyre: A new benchmark for physical reasoning. *Advances in Neural Information Processing Systems*, 32, 2019.

[11] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690, 2024.

[12] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

[13] Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

[14] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[15] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[16] Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Alphamath almost zero: process supervision without process. *arXiv preprint arXiv:2405.03553*, 2024.

[17] Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Step-level value preference optimization for mathematical reasoning. *arXiv preprint arXiv:2406.10858*, 2024.

[18] Jiaqi Chen, Tong Li, Jinghui Qin, Pan Lu, Liang Lin, Chongyu Chen, and Xiaodan Liang. Unigeo: Unifying geometry logical reasoning via reformulating mathematical expression, 2022.

[19] Jiaqi Chen, Jianheng Tang, Jinghui Qin, Xiaodan Liang, Lingbo Liu, Eric P. Xing, and Liang Lin. Geoqa: A geometric question answering benchmark towards multimodal numerical reasoning, 2022.

[20] Lin Chen, Fengli Xu, Nian Li, Zhenyu Han, Meng Wang, Yong Li, and Pan Hui. Large language model-driven meta-structure discovery in heterogeneous information network. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 307–318, 2024.

[21] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[22] Nuo Chen, Yan Wang, Haiyun Jiang, Deng Cai, Yuhan Li, Ziyang Chen, Longyue Wang, and Jia Li. Large language models meet harry potter: A dataset for aligning dialogue agents with characters. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8506–8520, 2023.

[23] Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. Theoremqa: A theorem-driven question answering dataset, 2023.

[24] 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. Training verifiers to solve math word problems, 2021.

[25] Julian Coda-Forno, Marcel Binz, Zeynep Akata, Matt Botvinick, Jane Wang, and Eric Schulz. Meta-in-context learning in large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[26] Ernest Davis. Testing gpt-4-o1-preview on math and science problems: A follow-up study. *arXiv preprint arXiv:2410.22340*, 2024.

[27] Joost CF de Winter, Dimitra Dodou, and Yke Bauke Eisma. System 2 thinking in openai's o1-preview model: Near-perfect performance on a mathematics exam. *Computers*, 13(11):278, 2024.

[28] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36, 2024.

[29] Xiang Deng, Yu Su, Alyssa Lees, You Wu, Cong Yu, and Huan Sun. Reasonbert: Pre-trained to reason with distant supervision. *arXiv preprint arXiv:2109.04912*, 2021.

[30] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[31] Bosheng Ding, Chengwei Qin, Ruochen Zhao, Tianze Luo, Xinze Li, Guizhen Chen, Wenhan Xia, Junjie Hu, Anh Tuan Luu, and Shafiq Joty. Data augmentation using llms: Data perspectives, learning paradigms and challenges. *arXiv preprint arXiv:2403.02990*, 2024.

[32] Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*, 2023.

[33] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

[34] Igor Douven. Abduction. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2021 edition, 2021.

[35] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[36] Tom Duenas and Diana Ruiz. The path to superintelligence: A critical analysis of openai's five levels of ai progression. *ResearchGate, 2024b. doi*, 10, 2024.

[37] Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

[38] Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*, 2023.

[39] Hang Gao and Yongfeng Zhang. Memory sharing for large language model based agents. *arXiv preprint arXiv:2404.09982*, 2024.

[40] Jinglong Gao, Xiao Ding, Yiming Cui, Jianbai Zhao, Hepeng Wang, Ting Liu, and Bing Qin. Self-evolving gpt: A lifelong autonomous experiential learner. *arXiv preprint arXiv:2407.08937*, 2024.

[41] Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Taco Cohen, and Gabriel Synnaeve. Rlef: Grounding code llms in execution feedback with reinforcement learning. *arXiv preprint arXiv:2410.02089*, 2024.

[42] Akshay Goel, Almog Gueta, Omry Gilon, Chang Liu, Sofia Erell, Lan Huong Nguyen, Xiaohong Hao, Bolous Jaber, Shashir Reddy, Rupesh Kartha, et al. Llms accelerate annotation for medical information extraction. In *Machine Learning for Health (ML4H)*, pages 82–100. PMLR, 2023.

[43] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*, 2023.

[44] Zhenyu Guan, Xiangyu Kong, Fangwei Zhong, and Yizhou Wang. Richelieu: Self-evolving llm-based agents for ai diplomacy. *arXiv preprint arXiv:2407.06813*, 2024.

[45] Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

[46] Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alex Wardle-Solano, Hannah Szabo, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander R. Fabbri, Wojciech Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. Folio: Natural language reasoning with first-order logic, 2024.

[47] James Hawthorne. Inductive Logic. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2024 edition, 2024.

[48] Kadhim Hayawi and Sakib Shahriar. A cross-domain performance report of open ai chatgpt o1 model. 2024.

[49] Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, et al. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*, 2021.

[50] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021.

[51] Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar, October 2014. Association for Computational Linguistics.

[52] Haichuan Hu, Ye Shang, Guolin Xu, Congqing He, and Quanjun Zhang. Can gpt-o1 kill all bugs? an evaluation of gpt-family llms on quixbugs. *arXiv e-prints*, pages arXiv–2409, 2024.

[53] Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*, 2024.

[54] Fantine Huot, Reinald Kim Amplayo, Jennimaria Palomaki, Alice Shoshana Jakobovits, Elizabeth Clark, and Mirella Lapata. Agents' room: Narrative generation through multi-step collaboration. *arXiv preprint arXiv:2410.02603*, 2024.

[55] Hyeonbin Hwang, Doyoung Kim, Seungone Kim, Seonghyeon Ye, and Minjoon Seo. Self-explore to avoid the pit: Improving the reasoning capabilities of language models with fine-grained rewards. *arXiv preprint arXiv:2404.10346*, 2024.

[56] interconnects.ai. blob reinforcement fine-tuning. (Accessed: 2025-12-6).

[57] Albert Q. Jiang, Wenda Li, Jesse Michael Han, and Yuhuai Wu. Lisa: Language models of isabelle proofs. *6th Conference on Artificial Intelligence and Theorem Proving*, 2021.

[58] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

[59] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[60] Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. Vineppo: Unlocking rl potential for llm reasoning through refined credit assignment. *arXiv preprint arXiv:2410.01679*, 2024.

[61] Hannah Kim, Kushan Mitra, Rafael Li Chen, Sajjadur Rahman, and Dan Zhang. Meganno+: A human-llm collaborative annotation system. *arXiv preprint arXiv:2402.18050*, 2024.

[62] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[63] Kazushi Kondo, Saku Sugawara, and Akiko Aizawa. Probing physical reasoning with counter-commonsense context. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 603–612, 2023.

[64] Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

[65] Teyun Kwon, Norman Di Palo, and Edward Johns. Language models as zero-shot trajectory generators. *IEEE Robotics and Automation Letters*, 2024.

[66] Philippe Laban, Wojciech Kryściński, Divyansh Agarwal, Alexander R Fabbri, Caiming Xiong, Shafiq Joty, and Chien-Sheng Wu. Llms as factual reasoners: Insights from existing benchmarks and beyond. *arXiv preprint arXiv:2305.14540*, 2023.

[67] Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. Ds-1000: A natural and reliable benchmark for data science code generation. In *International Conference on Machine Learning*, pages 18319–18345. PMLR, 2023.

[68] Ehsan Latif, Yifan Zhou, Shuchen Guo, Yizhu Gao, Lehong Shi, Matthew Nayaaba, Gyeonggeon Lee, Liang Zhang, Arne Bewersdorff, Luyang Fang, et al. A systematic assessment of openai o1-preview for higher order thinking in education. *arXiv preprint arXiv:2410.21287*, 2024.

[69] Hao Duong Le, Xin Xia, and Zhang Chen. Multi-agent causal discovery using large language models. *arXiv preprint arXiv:2407.15073*, 2024.

[70] Nicholas Lee, Thanakul Wattanawong, Sehoon Kim, Karttikeya Mangalam, Sheng Shen, Gopala Anumanchipalli, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. Llm2llm: Boosting llms with novel iterative data enhancement. *arXiv preprint arXiv:2403.15042*, 2024.

[71] M Lewis. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[72] Chengpeng Li, Guanting Dong, Mingfeng Xue, Ru Peng, Xiang Wang, and Dayiheng Liu. Dotamath: Decomposition of thought with code assistance and self-correction for mathematical reasoning. *arXiv preprint arXiv:2407.04078*, 2024.

[73] Leo Li, Ye Luo, and Tingyou Pan. Openai-o1 ab testing: Does the o1 model really do good reasoning in math problem solving? *arXiv preprint arXiv:2411.06198*, 2024.

[74] Minzhi Li, Taiwei Shi, Caleb Ziems, Min-Yen Kan, Nancy F Chen, Zhengyuan Liu, and Diyi Yang. Coannotating: Uncertainty-guided work allocation between human and large language models for data annotation. *arXiv preprint arXiv:2310.15638*, 2023.

[75] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[76] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023.

[77] Alisa Liu, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. Wanli: Worker and ai collaboration for natural language inference dataset creation, 2022.

[78] Chengwu Liu, Jianhao Shen, Huajian Xin, Zhengying Liu, Ye Yuan, Haiming Wang, Wei Ju, Chuanyang Zheng, Yichun Yin, Lin Li, Ming Zhang, and Qun Liu. Fimo: A challenge formal dataset for automated theorem proving, 2023.

[79] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023.

[80] Elita Lobo, Chirag Agarwal, and Himabindu Lakkaraju. On the impact of fine-tuning on chain-of-thought reasoning. *arXiv preprint arXiv:2411.15382*, 2024.

[81] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts, 2024.

[82] Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning, 2021.

[83] Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning, 2023.

[84] Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

[85] Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

[86] Man Luo, Shrinidhi Kumbhar, Ming shen, Mihir Parmar, Neeraj Varshney, Pratyay Banerjee, Somak Aditya, and Chitta Baral. Towards logiglue: A brief survey and a benchmark for analyzing logical reasoning capabilities of language models, 2024.

[87] Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. *arXiv preprint arXiv:2401.13178*, 2024.

[88] Pingchuan Ma, Tsun-Hsuan Wang, Minghao Guo, Zhiqing Sun, Joshua B Tenenbaum, Daniela Rus, Chuang Gan, and Wojciech Matusik. Llm and simulation as bilevel optimizers: A new paradigm to advance physical scientific discovery. *arXiv preprint arXiv:2405.09783*, 2024.

[89] Carey Maas, Saatchi Wheeler, Shamash Billington, et al. To infinity and beyond: Show-1 and showrunner agents in multi-agent simulations. *To infinity and beyond: Show-1 and showrunner agents in multi-agent simulations*, 2023.

[90] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

[91] Utkarsh Mall, Cheng Perng Phoo, Meilin Kelsey Liu, Carl Vondrick, Bharath Hariharan, and Kavita Bala. Remote sensing vision-language foundation models without annotations via ground remote alignment. *arXiv preprint arXiv:2312.06960*, 2023.

[92] Yujun Mao, Yoon Kim, and Yilun Zhou. Champ: A competition-level dataset for fine-grained analyses of llms' mathematical reasoning capabilities, 2024.

[93] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning, 2022.

[94] R Thomas McCoy, Shunyu Yao, Dan Friedman, Mathew D Hardy, and Thomas L Griffiths. When a language model is optimized for reasoning, does it still show embers of autoregression? an analysis of openai o1. *arXiv preprint arXiv:2410.01792*, 2024.

[95] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, 2018.

[96] Marie Mikulová, Milan Straka, Jan Štěpánek, Barbora Štěpánková, and Jan Hajič. Quality and efficiency of manual annotation: Pre-annotation bias. *arXiv preprint arXiv:2306.09307*, 2023.

[97] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.

[98] Arbi Haza Nasution and Aytug Onan. Chatgpt label: Comparing the quality of human-generated and llm-generated annotations in low-resource language nlp tasks. *IEEE Access*, 2024.

[99] Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial nli: A new benchmark for natural language understanding, 2020.

[100] Timothy Niven and Hung-Yu Kao. Probing neural network comprehension of natural language arguments, 2019.

[101] OpenAI. Reinforcement fine-tuning. (Accessed: 2025-12-6).

[102] OpenAI. Early access for safety testing. 2024.

[103] OpenAI. Learning to reason with llms. 2024.

[104] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[105] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22, 2023.

[106] Raul Puri, Ryan Spring, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Training question answering models from synthetic data. *arXiv preprint arXiv:2002.09599*, 2020.

[107] Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2023.

[108] Chen Qian, Yufan Dang, Jiahao Li, Wei Liu, Zihao Xie, Yifei Wang, Weize Chen, Cheng Yang, Xin Cong, Xiaoyin Che, et al. Experiential co-learning of software-developing agents. *arXiv preprint arXiv:2312.17025*, 2023.

[109] Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Eleanor Jiang, Chengfei Lv, and Huajun Chen. Autoact: Automatic agent learning from scratch via self-planning. *arXiv preprint arXiv:2401.05268*, 2024.

[110] Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, et al. O1 replication journey: A strategic progress report–part 1. *arXiv preprint arXiv:2410.18982*, 2024.

[111] Alec Radford. Improving language understanding by generative pre-training. 2018.

[112] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

[113] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[114] Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. *arXiv preprint arXiv:2210.01241*, 2022.

[115] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

[116] Subhro Roy and Dan Roth. Solving general arithmetic word problems, 2016.

[117] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740, 2020.

[118] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. 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 4463–4473, 2019.

[119] Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought, 2023.

[120] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2024.

[121] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[122] Bilgehan Sel, Ahmad Al-Tawaha, Vanshaj Khattar, Ruoxi Jia, and Ming Jin. Algorithm of thoughts: Enhancing exploration of ideas in large language models. *arXiv preprint arXiv:2308.10379*, 2023.

[123] Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

[124] Murray Shanahan, Kyle McDonell, and Laria Reynolds. Role play with large language models. *Nature*, 623(7987):493–498, 2023.

[125] Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic llm agent search in modular design space. *arXiv preprint arXiv:2410.06153*, 2024.

[126] Lior Shani, Aviv Rosenberg, Asaf Cassel, Oran Lang, Daniele Calandriello, Avital Zipori, Hila Noga, Orgad Keller, Bilal Piot, Idan Szpektor, et al. Multi-turn reinforcement learning from preference human feedback. *arXiv preprint arXiv:2405.14655*, 2024.

[127] Chenyang Shao, Fengli Xu, Bingbing Fan, Jingtao Ding, Yuan Yuan, Meng Wang, and Yong Li. Beyond imitation: Generating human mobility from context-aware reasoning with large language models. *arXiv preprint arXiv:2402.09836*, 2024.

[128] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[129] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[130] Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K Reddy. Llm-sr: Scientific equation discovery via programming with large language models. *arXiv preprint arXiv:2404.18400*, 2024.

[131] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

[132] Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. Clutrr: A diagnostic benchmark for inductive reasoning from text, 2019.

[133] W Smoke and E Dubinsky. A program for the machine translation of natural languages. *Mech. Transl. Comput. Linguistics*, 6:2–10, 1961.

[134] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[135] Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization for llm agents. *arXiv preprint arXiv:2403.02502*, 2024.

[136] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

[137] Theodore R Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*, 2023.

[138] Chuanneng Sun, Songjun Huang, and Dario Pompili. Retrieval-augmented hierarchical in-context reinforcement learning and hindsight modular reflections for task planning with llms, 2024.

[139] Richard Sutton. The bitter lesson. *Incomplete Ideas (blog)*, 13(1):38, 2019.

[140] Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language, 2021.

[141] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 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, 2019.

[142] Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. Large language models for data annotation: A survey. *arXiv preprint arXiv:2402.13446*, 2024.

[143] Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. Reft: Reasoning with reinforced fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7601–7614, 2024.

[144] Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. Llms still can't plan; can lrms? a preliminary evaluation of openai's o1 on planbench. *arXiv preprint arXiv:2409.13373*, 2024.

[145] Jun Wang, Meng Fang, Ziyu Wan, Muning Wen, Jiachen Zhu, Anjie Liu, Ziqin Gong, Yan Song, Lei Chen, Lionel M Ni, et al. Openr: An open source framework for advanced reasoning with large language models. *arXiv preprint arXiv:2410.09671*, 2024.

[146] Kevin Wang, Junbo Li, Neel P Bhatt, Yihan Xi, Qiang Liu, Ufuk Topcu, and Zhangyang Wang. On the planning abilities of openai's o1 models: Feasibility, optimality, and generalizability. *arXiv preprint arXiv:2409.19924*, 2024.

[147] Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*, 2023.

[148] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, 2024.

[149] Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Y Wu, and Zhifang Sui. Math-shepherd: A label-free step-by-step verifier for llms in mathematical reasoning. *arXiv preprint arXiv:2312.08935*, 2023.

[150] Tianlong Wang, Xueting Han, and Jing Bai. Cpl: Critical planning step learning boosts llm generalization in reasoning tasks. *arXiv preprint arXiv:2409.08642*, 2024.

[151] Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R. Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. Scibench: Evaluating college-level scientific problem-solving abilities of large language models, 2024.

[152] Xinru Wang, Hannah Kim, Sajjadur Rahman, Kushan Mitra, and Zhengjie Miao. Human-llm collaborative annotation through effective verification of llm labels. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–21, 2024.

[153] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

[154] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024.

[155] Zhiruo Wang, Shuyan Zhou, Daniel Fried, and Graham Neubig. Execution-based evaluation for open-domain code generation. *arXiv preprint arXiv:2212.10481*, 2022.

[156] Zihan Wang, Yunxuan Li, Yuexin Wu, Liangchen Luo, Le Hou, Hongkun Yu, and Jingbo Shang. Multi-step problem solving through a verifier: An empirical analysis on model-induced process supervision. *arXiv preprint arXiv:2402.02658*, 2024.

[157] Taylor Webb, Keith J Holyoak, and Hongjing Lu. Emergent analogical reasoning in large language models. *Nature Human Behaviour*, 7(9):1526–1541, 2023.

[158] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

[159] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[160] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[161] Jerry Wei, Da Huang, Yifeng Lu, Denny Zhou, and Quoc V Le. Simple synthetic data reduces sycophancy in large language models. *arXiv preprint arXiv:2308.03958*, 2023.

[162] Oskar Wysocki, Magdalena Wysocka, Danilo Carvalho, Alex Teodor Bogatu, Danilo Miranda Gusicuma, Maxime Delmas, Harriet Unsworth, and Andre Freitas. An llm-based knowledge synthesis and scientific reasoning framework for biomedical discovery. *arXiv preprint arXiv:2406.18626*, 2024.

[163] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

[164] Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, et al. Agentgym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*, 2024.

[165] Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*, 2024.

[166] Jing Xiong, Jianhao Shen, Ye Yuan, Haiming Wang, Yichun Yin, Zhengying Liu, Lin Li, Zhijiang Guo, Qingxing Cao, Yinya Huang, Chuanyang Zheng, Xiaodan Liang, Ming Zhang, and Qun Liu. TRIGO: Benchmarking Formal Mathematical Proof Reduction for Generative Language Models, October 2023. arXiv:2310.10180 [cs].

[167] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023.

[168] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[169] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024.

[170] Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. Leandojo: Theorem proving with retrieval-augmented language models, 2023.

[171] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

[172] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[173] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[174] Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. Reevo: Large language models as hyper-heuristics with reflective evolution. *arXiv preprint arXiv:2402.01145*, 2024.

[175] Nathan Young, Qiming Bao, Joshua Bensemann, and Michael Witbrock. Abductionrules: Training transformers to explain unexpected inputs, 2022.

[176] Yangyang Yu, Zhiyuan Yao, Haohang Li, Zhiyang Deng, Yupeng Cao, Zhi Chen, Jordan W Suchow, Rong Liu, Zhenyu Cui, Zhaozhuo Xu, et al. Fincon: A synthesized llm multi-agent system with conceptual verbal reinforcement for enhanced financial decision making. *arXiv preprint arXiv:2407.06567*, 2024.

[177] Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. How well do large language models perform in arithmetic tasks?, 2023.

[178] Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, 2018.

[179] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, 2019.

[180] Qingbin Zeng, Qinglong Yang, Shunan Dong, Heming Du, Liang Zheng, Fengli Xu, and Yong Li. Perceive, reflect, and plan: Designing llm agent for goal-directed city navigation without instructions. *arXiv preprint arXiv:2408.04168*, 2024.

[181] Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. Evaluating large language models at evaluating instruction following. In *International Conference on Learning Representations (ICLR)*, 2024.

[182] Chenhui Zhang and Sherrie Wang. Good at captioning, bad at counting: Benchmarking gpt-4v on earth observation data. *arXiv preprint arXiv:2401.17600*, 2024.

[183] Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search. *arXiv preprint arXiv:2406.03816*, 2024.

[184] Danyang Zhang, Lu Chen, Situo Zhang, Hongshen Xu, Zihan Zhao, and Kai Yu. Large language models are semi-parametric reinforcement learning agents. *Advances in Neural Information Processing Systems*, 36, 2024.

[185] Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, et al. Llama-berry: Pairwise optimization for o1-like olympiad-level mathematical reasoning. *arXiv preprint arXiv:2410.02884*, 2024.

[186] Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*, 2024.

[187] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*, 2023.

[188] Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. Agent-pro: Learning to evolve via policy-level reflection and optimization. *arXiv preprint arXiv:2402.17574*, 2024.

[189] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022.

[190] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642, 2024.

[191] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[192] Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. Multihiertt: Numerical reasoning over multi hierarchical tabular and textual data, 2022.

[193] Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. Minif2f: a cross-system benchmark for formal olympiad-level mathematics, 2022.

[194] Tianyang Zhong, Zhengliang Liu, Yi Pan, Yutong Zhang, Yifan Zhou, Shizhe Liang, Zihao Wu, Yanjun Lyu, Peng Shu, Xiaowei Yu, et al. Evaluation of openai o1: Opportunities and challenges of agi. *arXiv preprint arXiv:2409.18486*, 2024.

[195] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36, 2024.

[196] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

[197] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

[198] Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn rl. *arXiv preprint arXiv:2402.19446*, 2024.

[199] Zhilun Zhou, Jingyang Fan, Yu Liu, Fengli Xu, Depeng Jin, and Yong Li. Synergizing llm agents and knowledge graph for socioeconomic prediction in lbsn. *arXiv preprint arXiv:2411.00028*, 2024.

[200] Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Gptswarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*.