

Knowledge Augmented Complex Problem Solving with Large Language Models: A Survey

DA ZHENG^{*†}, Ant Group, China

LUN DU^{*}, Ant Group, China

JUNWEI SU, The University of Hong Kong, China

YUCHEN TIAN, Ant Group, China

YUQI ZHU, Zhejiang University, China

JINTIAN ZHANG, Zhejiang University, China

LANNING WEI, Ant Group, China

NINGYU ZHANG[†], Zhejiang University, China

HUAJUN CHEN[†], Zhejiang University, China

Problem-solving has been a fundamental driver of human progress in numerous domains. With advancements in artificial intelligence, Large Language Models (LLMs) have emerged as powerful tools capable of tackling complex problems across diverse domains. Unlike traditional computational systems, LLMs combine raw computational power with an approximation of human reasoning, allowing them to generate solutions, make inferences, and even leverage external computational tools. However, applying LLMs to real-world problem-solving presents significant challenges, including multi-step reasoning, domain knowledge integration, and result verification. This survey explores the capabilities and limitations of LLMs in complex problem-solving, examining techniques including Chain-of-Thought (CoT) reasoning, knowledge augmentation, and various LLM-based and tool-based verification techniques. Additionally, we highlight domain-specific challenges in various domains, such as software engineering, mathematical reasoning and proving, data analysis and modeling, and scientific research. The paper further discusses the fundamental limitations of the current LLM solutions and the future directions of LLM-based complex problems solving from the perspective of multi-step reasoning, domain knowledge integration and result verification.

Additional Key Words and Phrases: Large language models, reasoning, complex problem solving

ACM Reference Format:

Da Zheng, Lun Du, Junwei Su, Yuchen Tian, Yuqi Zhu, Jintian Zhang, Lanning Wei, Ningyu Zhang, and Huajun Chen. 2018. Knowledge Augmented Complex Problem Solving with Large Language Models: A Survey. *J. ACM* 37, 4, Article 111 (August 2018), 28 pages. <https://doi.org/XXXXXX.XXXXXXX>

^{*}Equal contribution

[†]Corresponding author

Authors' Contact Information: Da Zheng, Ant Group, Beijing, China, zhengda.zheng@antgroup.com; Lun Du, Ant Group, Beijing, China, dulun.dl@antgroup.com; Junwei Su, The University of Hong Kong, Hong Kong, China, junweisu@connect.hku.hk; Yuchen Tian, Ant Group, Beijing, China, wanglian.tyc@antgroup.com; Yuqi Zhu, Zhejiang University, Hangzhou, China, zhuyuqi@zju.edu.cn; Jintian Zhang, Zhejiang University, Hangzhou, China, zhangjintian@zju.edu.cn; Lanning Wei, Ant Group, Beijing, China, weilanning.wln@antgroup.com; Ningyu Zhang, Zhejiang University, Hangzhou, China, zhangningyu@zju.edu.cn; Huajun Chen, Zhejiang University, Hangzhou, China, huajunsir@zju.edu.cn.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM 1557-735X/2018/8-ART111

<https://doi.org/XXXXXX.XXXXXXX>

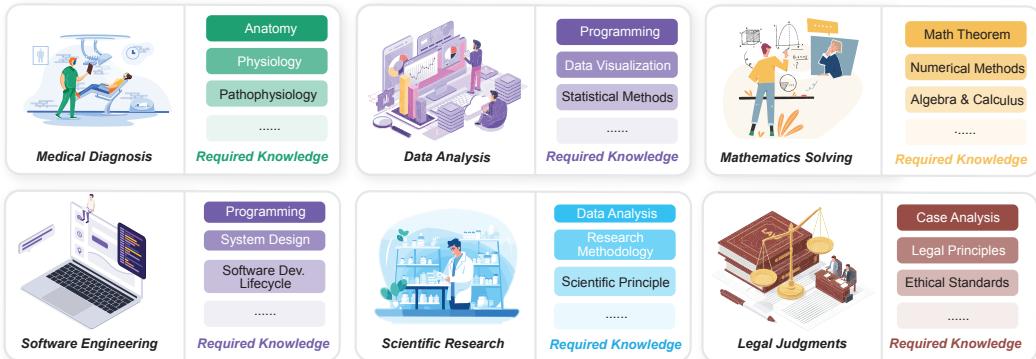


Fig. 1. Some scenarios for complex problem solving.

1 Introduction

The history of human civilization has been shaped by the ability of solving problems, ranging from constructing shelters in ancient times to unlocking the mysteries of the universe. For example, ancient astronomers calculated the Earth's size, while modern scientists predict weather using computational models. With technological advancements, humanity has gradually shifted from relying solely on individual or collective intellect to leveraging powerful tools like computers to address increasingly complex challenges. This transition marks a paradigm shift in problem solving, evolving from purely human-centered approaches to a synergy between human ingenuity and computational capability.

Today, LLM-based AI systems represent a groundbreaking advancement [77, 96, 171, 178]. Unlike traditional computers, which excel at precise calculations, LLMs simulate aspects of human reasoning, such as generating creative solutions and making contextual inferences. This positions LLMs as tools that combine computational power with an approximation of human thought to solve complex problems that are challenging to humans. Similar to human problem-solving, LLMs can directly solve problems and generate final results; LLMs can leverage computers to solve problems by writing and executing code to get results.

The scope of complex problem solving spans a wide range of domains, encompassing challenges that touch virtually every aspect of human society (Figure 1). For instance, designing robust software system architectures requires balancing scalability, reliability, and user needs, while proving mathematical theorems demands rigorous logical reasoning and abstraction. In the realm of data science, building accurate models to interpret vast datasets is essential for informed decision-making. Similarly, drug discovery involves navigating intricate molecular interactions to identify effective therapies, and constructing physical models enables us to simulate and understand natural phenomena. These examples highlight the diversity of complex problems humanity strives to solve, each requiring a blend of domain expertise, reasoning, and creativity.

Solving these real-world complex problems involves leveraging domain knowledge or experience and progressing through multiple reasoning steps to arrive at a final solution. In the community, mathematical reasoning is frequently studied as a representative form of complex problem-solving and current research predominantly focuses on the mathematical reasoning problems with definitive final answers. In contrast, mathematical theorem proving tasks – which are more representative of challenges encountered in higher education and research – are often overlooked because they typically lack a single final answer to verify. In practice, many real-world complex problems are

even more challenging than mathematical reasoning tasks. First, these problems are inherently difficult to verify. For instance, in data science, numerous modeling techniques can be applied to the same dataset, yet their performance may vary greatly. Moreover, the effectiveness of a model is highly context-dependent, differing across problems and datasets. This variability makes it difficult to determine the optimal solution for any given modeling task. Second, solving such real-world problems requires substantial domain expertise. Using data modeling again as an example, one must not only understand the nuances of the data but also be proficient in a wide range of modeling techniques to achieve strong performance.

Solving real-world complex problems requires three key components: *multi-step reasoning*, *domain knowledge*, and *result verification*. This problem-solving process presents multiple challenges when LLMs are applied to real-world problems.

- **Multi-step reasoning:** Solving a complex problem requires taking multiple steps to reach the final outcome. The size of the search space is largely determined by the number of steps needed to solve a complex problem, and can grow exponentially as the number of reasoning steps increases. This makes it challenging to identify the correct path to the final result. In addition, any errors that occur in the search process can propagate and lead to an incorrect result.
- **Domain knowledge:** Knowledge plays a crucial role in guiding the problem solver through the search space, helping to identify the next step or recognize when the solution has been reached. Domain-specific applications, such as machine learning tasks and mathematical theorem proving tasks, typically require to utilize long-tail domain knowledge while it is well-known that LLMs cannot master low-tail knowledge well [121]. This requires an LLM-based system to take extra care to master domain knowledge and reliably retrieve and apply the required knowledge to solve problems.
- **Result verification:** Each step must be carefully evaluated to assess whether it contributes to a correct solution or whether the entire solutions can solve the given problem. This evaluation can be particularly challenging in many applications where standard outcomes or predefined solution procedures are lacking. The difficulty is even greater for open-ended problems with ill-defined goals, such as those found in scientific research and data mining.

Recent development of LLMs have demonstrated their strong reasoning capabilities on some complex problems that have well-defined goals and whose results can be easily verified, making it ideal for tasks like mathematical reasoning and competitive coding challenges. Chain-of-Thought (CoT) reasoning is the major technique to solve multi-step reasoning [13, 136, 149, 175]. There is an inference scaling law in CoT reasoning that the likelihood of finding a correct solution improves significantly as the number of CoT paths increases [20] and it is often possible to generate correct solutions for many challenging problems with a sufficient number of CoT paths [10]. Because the target applications, such as mathematical reasoning and competitive coding, are easily verifiable, many works [23, 65] are using reinforcement learning to train LLMs to improve their reasoning capabilities for these applications [74]. The release of GPT-01 by OpenAI [99] and DeepSeek-R1 [23] showcases the potential of this CoT reasoning approach [161].

While CoT reasoning is an important technique to solve complex problems, it is necessary to adopt an agentic approach that enables access to external knowledge bases and the use of verification tools to further improve LLMs' capability in solving real-world complex problems. Previous studies have shown that LLMs have difficulty retaining long-tail knowledge [121], and domain-specific knowledge often falls into this category. It is essential to make external knowledge integration for knowledge-intensive tasks such as scientific discovery [1], mathematical theorem proving [138], and data science [44], where domain expertise is critical for accurate and informed

decision-making. Knowledge can be retrieved from documents through techniques like RAG [36] and GraphRAG [30, 48], or by leveraging knowledge graphs constructed from document collections [78]. Additionally, agents may interact with humans to acquire domain knowledge directly [26, 174]. Result verification is also essential to ensure valid solutions from large language models (LLMs), both during training and inference. Reasoning-focused LLMs are often trained with synthetic data, which requires a verifier to select high-quality data for model training [20]. During inference, the inference scaling law highlights the need for a verifier to identify the correct solution among multiple candidates [10]. Various types of verifiers can be employed for this purpose, including LLM-as-a-judge approaches [41], symbolic reasoning tools [19], and even experimental validation systems [76].

Despite the significant advancements in LLMs for complex problem solving, each domain presents its own unique challenges when applying LLMs to practical applications. Take some domains in Figure 1 as examples. In software engineering, LLMs are tasked with generating or modifying code within large code repositories for bug fixings and new feature implementations. This requires them not only to reason about code generation but also to have a comprehensive understanding of the entire codebase and project requirements [146]. Furthermore, software development demands not just code correctness but also optimization in terms of computational efficiency and memory usage [111], adding an additional layer of complexity to the evaluation process. Mathematics encompasses two primary types of tasks: calculations and provings. While extensive data are available for basic arithmetic and computational tasks, data scarcity remains a significant challenge in advanced mathematics, particularly in higher education and research [39]. To address this limitation, it is essential to leverage domain knowledge more effectively for data synthesis to mitigate the impact of data scarcity and utilize existing mathematical knowledge, such as theorems, to improve mathematical proving. In addition, mathematical theorem proving usually lack an effective way to verify the proving solution, making it difficult to train LLM models to generate rigorously correct mathematical reasoning solutions. Data science involves working with large datasets, yet task descriptions often lack sufficient details about the distribution of input data, making it challenging for LLMs to generate the most suitable solutions to model the large datasets well [12]. This also complicates the evaluation of LLM-generated outputs, necessitating multi-level assessments. Furthermore, leveraging a comprehensive knowledge base of data modeling techniques is crucial for developing more effective methods to tackle complex data science problems. Scientific research often involves open-ended problems, which prevents us from training LLMs to solve scientific problems directly. One potential solution is to involve humans in the process (human-LLM collaboration), allowing for iterative collaboration between humans and LLMs to explore existing scientific literature and human knowledge [8, 60, 85, 88, 109, 123], generate novel ideas [5, 112, 128, 131] and automate entire research pipelines [87]. These challenges highlight the need for further research into complex problem-solving that go beyond the current reasoning LLMs.

This paper provides a high-level overview of the current advancements in LLMs for solving complex problems and goes beyond the literature of reasoning LLMs. Our goal is to review the key techniques developed for LLMs and how these methods are being applied to address the challenges in different domains. The paper is structured into four sections to discuss the current LLM research:

- **Definition of complex problem solving:** We begin by formally defining complex problem solving from the perspectives of cognitive science and computational theory (Section 2).
- **Methodologies:** We examine the key methodologies in LLM research for solving complex problems, including multi-step reasoning, knowledge augmentations and result verifications (Section 3).

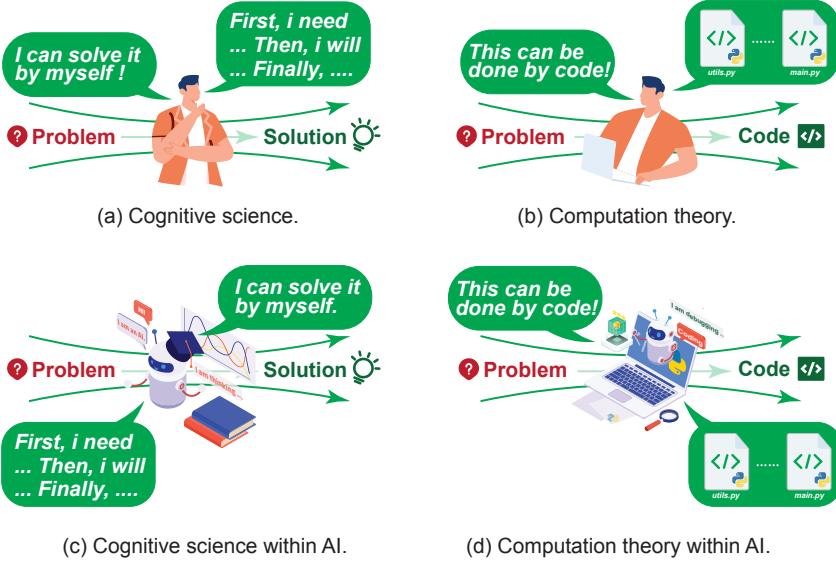


Fig. 2. Two paradigms of problem solving by humans and AI.

- **Domains:** We explore complex problem solving across four domains –software engineering, data science, mathematics, and scientific research –highlighting the unique challenges in each and the solutions developed to address them (Section 4).
- **Current limitations and future directions:** We discuss the limitations of current research and propose potential directions for future studies (Section 5).

2 Definition of Complex Problem Solving

We can define complex problem solving from two perspectives: **cognitive science** and **computational theory**. Cognitive science investigates how humans use their inherent abilities to solve problems. Computational theory, by contrast, explores how to leverage machines for problem solving, emphasizing the design of algorithms to automate intricate computations. When considering the role of LLMs in addressing complex problems, two potential paradigms emerge: (1) Direct problem solving: LLMs autonomously generate solutions akin to human experts.¹; (2) Leveraging computational systems for complex problems: LLMs extract and define the computational components of a problem, utilizing traditional computers to execute intensive calculations while focusing on designing solutions and orchestrating processes. With these paradigms in mind, this section will delve deeper into how CPS is defined in the frameworks of cognitive science and computational theory.

2.1 Definitions

DEFINITION 1 (PROBLEM). A problem $\Pi(X, \mathcal{Y}, \mathcal{P})$ is described by (1) a description of its parameters X , and (2) a statement (i.e., a predicate logic) $\mathcal{P}(Y; X)$ that characterizes the properties the solution must satisfy. Formally, the **goal** set is defined as $\mathcal{G} = \{Y \in \mathcal{Y} \mid \mathcal{P}(Y; X)\}$, where \mathcal{Y} is the space of final results, and \mathcal{P} is a predicate logic that means $\mathcal{P}(Y; X)$ represents the property that a final result

¹+wln+: generate plans and manage the tools to solve the problems like human experts. (These components will be mentioned in the following methodologies and domains.)

Y should satisfy when $X = X$. An instance π of the problem is obtained by specifying particular values for all the problem parameters, i.e., $\pi := \Pi(X = X)$.

A problem can be seen as a task that involves finding a solution from a set of possible candidates. The predicate $\mathcal{P}(Y; X)$ specifies the condition that an answer must satisfy to be considered valid. In different problems, the predicate $\mathcal{P}(Y; X)$ may either be well-defined or not. For instance, in the shortest path problem, the answer space \mathcal{Y} consists of all possible paths, and the predicate $\mathcal{P}(Y; X)$ is well-defined, specifying that a final result Y (a path) must satisfy the property of having the minimum total weight. In contrast, in data mining tasks, the goal is to discover insightful patterns within the data. However, what constitutes an “insightful” pattern is not clearly defined, making the predicate $\mathcal{P}(Y; X)$ more subjective and context-dependent.

Based on the definition of a problem, we can now formally define **problem solving** as the process of identifying a sequence of transformations that leads from an initial state to a goal state.

DEFINITION 2 (PROBLEM SOLVING). *Problem solving is the process of finding a solution trace $T(\pi) \in \mathcal{T}_{\text{feasible}} \subseteq \mathcal{T}$ for a problem instance π , where $\mathcal{T}_{\text{feasible}}$ is the set of all possible solution traces, formally defined as:*

$$\begin{aligned}\mathcal{T}_{\text{feasible}} := & \{X \rightarrow O_1 \rightarrow \dots \rightarrow O_\kappa \rightarrow Y | \\ & X \in \mathcal{X}, Y \in \mathcal{Y}, \kappa \in \mathbb{N}^+, \forall_{1 \leq i \leq \kappa} O_i \in \mathcal{O}\},\end{aligned}$$

, \mathcal{T} is the set of all possible traces:

$$\mathcal{T} := \{X \rightarrow O_1 \rightarrow \dots \rightarrow O_\kappa \rightarrow Y | X \in \mathcal{X}, Y \in \mathcal{Y}, \kappa \in \mathbb{N}^+, O_i \in \mathcal{O}\},$$

, and \mathcal{O} is the set of all possible intermediate states during the problem-solving process.

This definition emphasizes the iterative and state-dependent nature of problem solving, where intermediate states O_i capture the evolving understanding or partial solutions leading to the final result Y . However, the mechanisms driving state transitions and the constraints on feasible solution traces vary depending on the nature of the problem solver.

In a **human-centered** perspective, problem solving is inherently constrained by individual cognitive capabilities. The transition from one state to another is influenced not only by logical reasoning but also by domain knowledge, prior experience, and real-time feedback. As a result, different individuals may follow different paths within $\mathcal{T}_{\text{feasible}}$ based on their available cognitive resources. The formal definition is as follows:

DEFINITION 3 (HUMAN-CENTERED PROBLEM SOLVING). *Human-centered Problem solving is the process of finding a solution trace $T(\pi)$ for a problem instance π by a person with cognitive capabilities C , which include domain knowledge, logical reasoning, leveraging real-time feedback and other cognitive resources [67]. The transition from an intermediate state O_i to the next state O_{i+1} is governed by a cognition-guided transition function:*

$$\Gamma : \mathcal{O} \times C \rightarrow \mathcal{P}(\mathcal{O}),$$

where $\mathcal{P}(\mathcal{O})$ is the power set of \mathcal{O} , representing all possible next states, and the transition function $\Gamma(O_i, C)$ determines the set of feasible next states given the solver’s cognitive capacity.

Conversely, in a **computer-assisted** perspective, problem solving is approached from the lens of computational theory. Here, state transitions are governed by formal algorithms rather than cognitive capabilities.

DEFINITION 4 (COMPUTER-ASSISTED PROBLEM SOLVING). *Problem solving is the process of designing algorithms \mathcal{A} to solve a problem $\Pi(\mathcal{X}, \mathcal{Y}, P)$. An algorithm is a finite sequence of instructions*

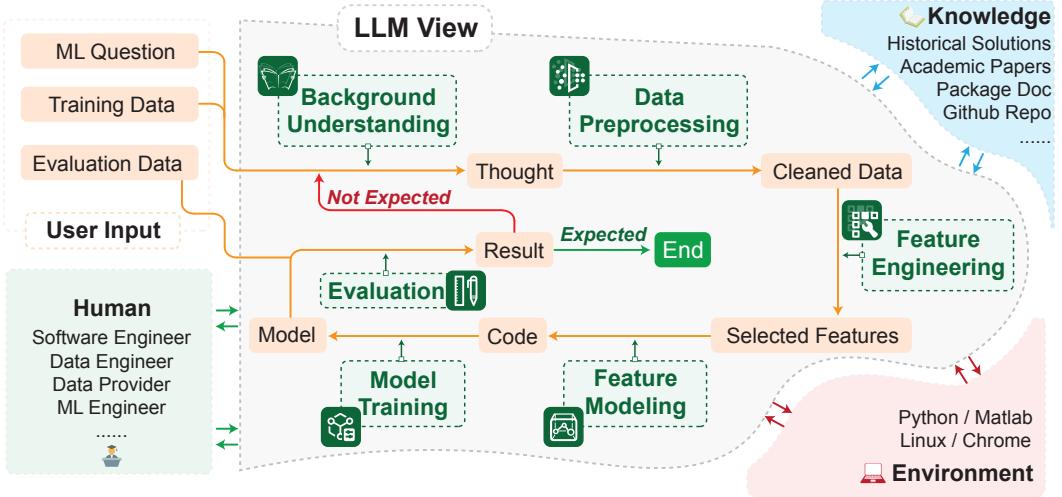


Fig. 3. Case Study: Example from the Field of Machine Learning

executable by a computer² to produce solutions. Formally, an algorithm is defined as a quintuple:

$$\mathcal{A} := (\mathcal{X}, \mathcal{Y}, \mathcal{O}, \delta, \sigma_0),$$

where \mathcal{X} is the input space, describing all possible parameters of the problem; \mathcal{Y} is the output space, representing all potential solutions; \mathcal{O} is the state space, containing all possible states during the execution of the algorithm; $\delta : \mathcal{X} \times \mathcal{O} \rightarrow \mathcal{O}$ is the state transition function, specifying how the algorithm transitions from one state to the next based on the input; $\sigma_0 \in \mathcal{O}$ is the initial state, representing the starting condition of the algorithm. An algorithm is said to solve a problem Π if, for any instance $\pi \in \Pi$, it guarantees to produce a solution $Y \in \mathcal{G}$ that satisfies the predicate $P(Y; X)$.

By comparing the two definitions, we observe that they share a fundamental similarity: both focus on finding the steps to solve a problem. However, they differ in their emphasis. **Human-centered Problem Solving** mainly focuses on the process of solving a specific problem instance π , while **Computer-assisted Problem Solving** emphasizes designing relatively general algorithms to address a class of problems Π .

2.2 Example

Let's use machine learning tasks as an example (Figure 3). Developing a high-quality machine learning model can be framed as a problem-solving process, where we seek a feasible solution trace $T(\pi) \in \mathcal{T}_{\text{feasible}}$. Each step in this process corresponds to a state transition $O_i \rightarrow O_{i+1}$, driven by reasoning, domain knowledge, and iterative evaluation. Initially, we define the problem by identifying the task and structuring it into a machine learning formulation. Then, we transition through intermediate states by analyzing data, applying preprocessing techniques, and performing feature engineering. Once the data is processed, we select suitable modeling techniques and develop the model for training. To refine these transitions, domain knowledge plays a crucial role, guiding the selection of appropriate models and training strategies. Knowledge may originate from historical approaches, theoretical research, or expert intuition, shaping the feasible state space $\mathcal{T}_{\text{feasible}}$. Developing effective machine learning models requires multiple rounds of evaluation for each

²Refers to a Turing machine or an equivalent computational architecture.

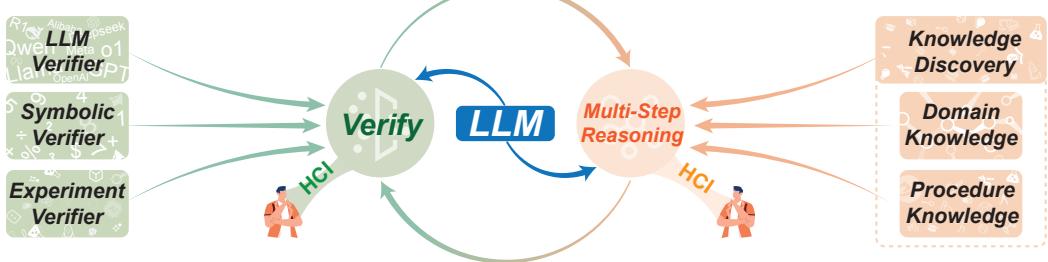


Fig. 4. The loops of complex problem solving.

method, including both human assessments and experimental evaluations. Since machine learning models rely on learning data distributions from training data to make predictions, assessing the quality of a solution solely by examining it is challenging. Instead, empirical validation through human assessments and experimental testing determines the effectiveness of a model before convergence to an optimal solution Y .

3 Methodology

Figure 4 illustrates LLM-based techniques for complex problem solving. Current Chain-of-Thought (CoT) LLMs are trained through data synthesis. The process begins with generating CoT data, followed by selecting the correct CoT samples using a verifier for model training. During inference, the LLM generates multiple CoT solutions, and a verifier is used to identify the correct one for the given task. There are multiple approaches to synthesizing data. One method involves having the LLM generate CoT data autonomously, which requires the base model to be well-trained. For applications with limited training data, knowledge mining can be conducted on existing datasets to synthesize data, while human expertise can also be incorporated. Additionally, the mined knowledge can be injected into the LLM during inference rather than being solely used for training. Certain applications produce results that are difficult to verify, such as machine learning tasks. In such cases, multiple verification methods can be employed. Besides using an LLM-based verifier, symbolic verification and experimental evaluations can be conducted. Additionally, human experts may also be involved in the verification process.

3.1 Multi-step reasoning

Chain-of-thought reasoning with LLM has been proven to be effective for solving complex problems. This line of research started from [136] that shows chain-of-thought prompting with a few examples of reasoning paths can enhance the reasoning capabilities of LLM. [64] later demonstrates that chain-of-thought reasoning can improve performance in a zero-shot setting by simply encouraging LLM to generate intermediate reasoning steps with the “Let’s think step by step” prompt. [132] shows that sampling multiple reasoning paths and using majority vote can further improve the performance of LLM in reasoning tasks. [155] introduces Tree of Thoughts (ToT) that allows LLMs to explore multiple reasoning paths over thoughts to improve the reasoning capabilities of LLMs.

We can utilize the architecture depicted in Figure 4 to improve chain-of-thought reasoning for tackling complex problems. When a question is posed, a *generator* powered by a large language model (LLM) produces multiple reasoning paths. These paths are then evaluated by a *verifier* to determine their accuracy. If some of the reasoning paths are validated as correct, they are used to formulate an answer to the question. However, if none of the paths are deemed correct, a *corrector* is employed to create new reasoning paths by modifying the incorrect ones and incorporating

additional feedback from the verifier. In this approach, enhancing the likelihood of getting a correct solution for any given question requires improving two key metrics:

- **coverage**: the percentage of problems that can be solved using at least one of the generated reasoning paths,
- **precision**: the probability of selecting the correct reasoning path from all generated paths.

To boost coverage, we need to fine-tune both the *generator* and the *corrector* to increase the chances of producing a valid reasoning path. To enhance precision, the *verifier* must be fine-tuned to more accurately identify the correct path.

Generator. To refine the generator, we must move beyond human-produced data and instead synthesize data with reasoning paths. [159] present an iterative procedure that generates multiple reasoning paths, selects the correct ones, and uses them to further fine-tune the LLM, progressively improving its ability to produce accurate reasoning. Additionally, they introduce a "rationalization" technique that leverages the problem's answer as a hint to enhance the generation of reasoning paths. [113] takes a similar iterative approach of generating reasoning paths to fine-tune LLM and using fine-tuned LLM to generate more reasoning paths. The main difference is that this work generates reasoning paths by using temperature sampling to generate multiple paths and use a binary reward function to score them, while [159] uses greedy decoding to generate reasoning paths. Both works show that the LLM model overfits quickly with the generated data. [7] shows that it is not necessary to use a strong LLM to generate high-quality synthetic data and the models fine-tuned on data generated by weaker LLMs can consistently outperform those trained on data generated by stronger models.

Self-correction. We can use the incorrect reasoning paths from the previous attempt and the feedbacks from the verifier to increase the probability of generating a correct one in the next iteration. This process is considered as *self-correction*. [52] shows that the existing LLM, such as GPT-4 and Llama-2, cannot increase the probability of generating right paths when being used for self-correction. They tend to decrease the probability of getting right solutions compared with standard prompting methods. This indicates that we need a specifically fine-tuned LLM for self-correction. Pair-SFT [139] trains an independent corrector model to refine the outputs of a generator model. They create a dataset consisting of response pairs (y, y') , where y is the initial response to a question and y' is the corrected version, to train the corrector. SCoRe [65] employs a reinforcement learning approach to train a single LLM that both generates the initial response and self-corrects it. They find that previous methods are ineffective due to distribution shifts or the amplification of biases from the base model. By using a single LLM for both response generation and correction, SCoRe avoids the distribution mismatch that arises between separate generator and corrector models.

Inference scaling law. It is costly to generate many reasoning paths and select a correct one. The more difficult a problem is, the more reasoning paths we may need to sample. A key research question here is how to use the compute resource wisely to find a right path for any given problem. [10] illustrates the scaling law for inference-time compute. They observe that the coverage grows nearly log-linearly with the number of samples generated from LLM and may reach to 100% coverage if many reasoning paths are generated. They further discover that it may be more cost-effective to generate more samples with a weaker model than using larger LLMs when solving some simpler problems; however, stronger LLMs are preferred when solving more difficult problems. [114] investigates the "compute-optimal" strategy for scaling inference-time compute in multiple aspects when generating a right reasoning path. When using a reward model to search for a good reasoning path, they evaluated different search strategies, including best-of-N search, beam

search and lookahead search, and conclude that beam search is preferable for harder problems and lower computational budgets, while best-of-N is more effective for easier problems and higher budgets. Another aspect is to update the proposal distribution of the generator model to increase the probability of generating a good reasoning path. An option is to generate multiple reasoning path in parallel, while the other is to use a fine-tuned LLM to iteratively revise their own answers, which results in a sequential test-time inference. They show that easier questions benefit from sequential inference while harder problems require some ratio of sequential to parallel inference. Compute resources can also be allocated to pre-training. To solve hard problems, some compute resources should be used for pre-training while for easier problems, we only need to use compute resources for inference.

3.2 Knowledge

Solving complex problems requires leveraging knowledge effectively. On one hand, many complex problem solving tasks are inherently domain-specific, and without specialized knowledge, solving them effectively becomes a challenge. On the other hand, the process of tackling these tasks involves multifaceted procedures, and existing large language models often lack the reliability and robustness required. Consequently, acquiring and enhancing this type of specialized knowledge represents a critical issue for effective complex problem solving.

To acquire this type of knowledge, the simplest and most direct approach is domain-specific pre-training [115]. While LLMs acquire world knowledge during training, research has shown that they are unreliable in memorizing and applying such knowledge—particularly long-tail information [121]—to practical tasks, and multiple studies [38, 106] indicate that they are unable to acquire new factual knowledge through supervised fine-tuning (SFT) after pre-training. Unlike these approaches, prompt-based enhancement techniques—such as RAG [36], GraphRAG [30, 48], and KAG [78]—can directly embed domain knowledge into the context of a specific task. Building on this, many studies have explored acquiring such knowledge through methods like information extraction [141, 167, 168], constructing domain-specific knowledge graphs [24, 169, 181], or procedure extraction [158], as well as directly generating task-specific workflows with large language models [104, 165] and refining them through human interactive feedback [9]. The following sections introduce various works categorized by the type of knowledge they address.

Domain Knowledge. Domain knowledge is designed to provide prior information for complex tasks, offering comprehensive introductions, detailed descriptions, and relevant background [18, 42, 102]. [80] proposes a computational framework that enhances agents’ problem-solving abilities by integrating a goal-directed dynamic knowledge generation mechanism. [126] introduces Knowledge-Driven Chain-of-Thought (KD-CoT), a framework that leverages external knowledge to verify and refine reasoning traces, thereby mitigating hallucinations and reducing error propagation. [73] introduces Chain-of-Knowledge (CoK), an innovative framework that enhances large language models by dynamically integrating grounding information from diverse sources. [101] proposes Physics Reasoner, a knowledge-augmented framework that leverages large language models to solve physics problems.

Procedure Knowledge. Procedure Knowledge refers to workflows or cognitive patterns designed to address complex problems, typically used to standardize and guide the reasoning processes of large models. Techniques like MoT [72] leverage synthetic and extracted high-quality thought processes as external memory, providing the model with superior problem-solving examples. Further, the BoT [150] paradigm introduces meta-buffers that store cross-task cognitive templates encompassing general reasoning patterns and knowledge structures, which can be reused and instantiated across various specific problems, thereby enabling efficient reasoning. Furthermore, methods like

Expel [177] also involve collecting an experience pool through environmental interactions, wherein the model learns from similar experiences and contrasts successful and unsuccessful trajectories to gain new insights and improve task inference capabilities. [182] introduces KnowAgent, an approach that boosts the planning capabilities of LLMs by integrating explicit action knowledge. Additional research by [135, 165] utilize workflows that selectively guide agents for complex problem solving.

Human-computer Interaction. Even with an external knowledge base, LLMs can still struggle with nuanced or domain-specific information, often lacking the deep contextual understanding that human experts possess. To address this, LLMs can collaborate with humans to bridge this gap by enabling humans to provide critical insights, ensuring that the LLM focuses on relevant information and refines its interpretations based on specialized knowledge. For instance, in tasks like legal or medical research, humans can guide LLMs to prioritize certain references or nuances that the model might overlook [4, 55, 117, 140]. To enable such human-LLM collaboration, we need to design intuitive, user-friendly interfaces that facilitate effective communication and interaction between humans and LLMs [26, 174]. These interfaces should enable effective bidirectional communication, where users can provide feedback, clarify ambiguous inputs, and track the LLM’s reasoning in real-time. A well-designed interface fosters trust, enhances collaboration, and ensures that the LLM can be effectively used by both experts and non-experts alike.

3.3 Evaluation

When tackling complex problems, it is essential to evaluate the effectiveness of solutions to enhance the reliability of LLM-based systems and identify better approaches. Prior research [97, 110] has demonstrated that LLMs are easily distracted by irrelevant information in mathematical reasoning. This suggests that LLMs may not truly grasp mathematical concepts but rather rely on pattern matching to generate responses. Additionally, [93] highlights that LLMs perform worse on rare tasks than on more frequent ones, even when the tasks share the same level of complexity. Moreover, LLMs are sensitive to the probability distribution of inputs and outputs in their training data (Internet text), even for deterministic tasks. These challenges become even more pronounced when LLMs are applied to domain-specific problems that are less commonly found online. To comprehensively assess solutions, multiple evaluation criteria—such as correctness and efficiency—may be considered. Ensuring that a solution meets practical requirements necessitates the development and integration of diverse evaluation techniques to effectively analyze LLM-generated solutions.

To improve chain-of-thought reasoning, we need a verifier model for selecting a right reasoning path. This was first demonstrated by [20] in solving math problems in GSM8K. This work shows that training a verifier model to select a right solution among multiple solutions can significantly improve the test solve rate compared with just fine-tuning LLM. Therefore, a key problem here is how to train a reliable verifier model to determine the right reasoning path. [81] shows the effectiveness of using process supervision to train a process reward model (PRM). This method first generates multiple reasoning paths for a problem and has human labelers to annotate labels for each individual step of the reasoning paths. This approach requires many human resources to prepare for the training data. [81] adopts active learning to reduce human labeling efforts. [127] proposes a method that eliminates the need for human labeling when training a PRM. To assess the correctness of intermediate steps in a reasoning path, they use a fine-tuned LLM to generate multiple subsequent reasoning paths from a given step. The correctness score for that step is then determined by the number of paths that successfully lead to the correct answer. [166] trains a generative model as a verifier and shows that a generative model outperforms a discriminative

verifier. In addition, they show that training a single LLM for both generation and verification can outperform separate LLMs for generation and verification.

In addition to LLM-based verification, tools are employed to validate model outputs, mitigating hallucinations and enhancing accuracy. These verification methods can be broadly categorized into *symbolic verification* and *experimental verification*.

Symbolic verification uses formal methods to ensure the correctness of the outputs of LLMs. This includes generating executable code and verifying its syntax and semantics through compilation [14, 34]. Additionally, outputs are compared against knowledge bases or knowledge graphs to validate factual accuracy. These methods are especially effective for tasks that require logical consistency, such as mathematical proofs or domain-specific fact-checking [19]. PAL [34] uses symbolic reasoning to interpret natural language problems and generate programs as intermediate steps. These programs are verified in runtime environments, like Python interpreters, ensuring that the logic and structure of the generated code are valid and executable. In mathematical reasoning, tools like those in [162] provide specialized interfaces for numerical computation, equation solving, and expression transformation. These interfaces allow the model to verify and correct each step, ensuring the correctness of the reasoning process, much like symbolic theorem proving. Factool [19] provides a flexible, domain-independent framework designed to identify factual errors. It enhances fact verification across domains by utilizing multiple verification tools, including search engines, academic databases, and code interpreters.

In contrast, experimental verification involves validating models through real-world testing and empirical experiments [12, 40, 76]. This approach is useful when formal verification is impractical or when the goal is performance optimization. Models are tested in practical environments or simulations, with performance measured against benchmarks or competing solutions. In automated data science, frameworks like AutoKaggle [76] exemplify experimental verification. These models autonomously participate in Kaggle competitions, optimizing data analysis pipelines and achieving top-tier performance by iterating through real-world testing, model tuning, and comparative analysis. Grosnit et al. [40] orchestrates structured reasoning to automatically analyze and optimize solutions, while Li et al. [76] uses a multi-agent framework to generate, test, and refine models.

For critical applications, ensuring safety and robustness is critical when applying LLMs in high-stakes or unpredictable environments, where incorrect outputs can lead to severe consequences. LLMs, while powerful, can generate unreliable or unsafe responses due to hallucinations, misinterpretations, or unexpected inputs. In this case, we should introduce human oversight to validate and correct outputs, ensuring safer and more reliable decision-making. For instance, in medical diagnosis, human experts can verify AI-generated treatment recommendations to avoid misdiagnoses or unsafe prescriptions [47?].

4 Domains

This paper examines four domains of real-world applications where LLMs can be applied to solve complex problems in these domains: software engineering, mathematics, data science, and scientific research. We will discuss the challenges in these applications from the perspective of multi-step reasoning, knowledge integration and result verifications.

4.1 Software Engineering

This involves enabling LLMs to perform complex software engineering tasks with minimal human intervention. The core tasks in this domain are generally categorized into two major areas: code generation and code understanding. Code generation encompasses program synthesis [58, 69, 71, 124, 173], code translation [16, 100, 145], automatic program repair [53, 59, 107], and code optimization [28, 144], where LLMs must produce functionally correct and efficient code that

satisfies diverse specifications. Code understanding, on the other hand, focuses on analyzing and interpreting existing code, involving tasks such as source code summarization [70, 133, 163], code review [154], and code search [27, 122]. Although these tasks differ in goals, they both demand that LLMs deeply understand the syntax, semantics, and structure of codebases and reason across multiple levels of abstraction.

Solving complex software engineering tasks using LLMs presents several unique challenges. First, these tasks require multi-step reasoning, as software development often involves decomposing problems, maintaining contextual consistency across files or functions, and iteratively refining code. Second, knowledge integration is essential—LLMs must possess foundational programming knowledge (e.g., syntax, algorithms), domain-specific practices (e.g., tool usage, design patterns) as well as the large code repository. Third, result verification is nontrivial: generating syntactically correct code is insufficient; it must also compile, execute correctly, and meet performance goals. Unlike natural language tasks, software correctness can be formally tested, creating both an opportunity and a challenge for using execution feedback effectively.

To address these challenges, a variety of models and frameworks have been proposed. For program synthesis, methods such as Code Evol-Instruct [90] and OSS-INSTRUCT [137] enhance LLM capabilities through synthetic data generation and fine-tuning, while approaches like GraphCoder [84] and GALLa [176] inject structural representations (e.g., code graphs) to improve syntactic and semantic understanding. Feedback-based mechanisms—such as Self-Debugging [15], LDB [180], and RLTf [82]—use runtime outputs, compiler errors, or test cases to iteratively guide model refinement. In repository comprehension, tools like StarCoder2 [86], DeepSeek-Coder [43], SWE-GPT [91], and RepoCoder [164], CoCoMIC[25] and RepoFuse[79] utilize repository-level information, dependency graphs, and retrieval-augmented generation (RAG) to help models navigate large and interdependent codebases. For code optimization, frameworks like PIE-Problem [157] and SBLLM [35] introduce multi-programmer solution sets and evolutionary search strategies to help LLMs learn from diverse optimization techniques and refine code based on execution metrics.

Future work in automating software engineering will likely focus on three directions. First, building stronger reasoning-aware models that can generate and revise code through intermediate abstractions, such as pseudocode or symbolic plans. Second, enhancing long-context and memory mechanisms to handle complex repositories and cross-file reasoning. Third, incorporating closed-loop feedback systems that integrate automated test generation, runtime profiling, and formal verification into the code generation process. By combining these approaches, we can expect LLM-based agents to evolve from basic code assistants into capable autonomous software engineers.

4.2 Mathematics

Mathematical reasoning has emerged as a crucial benchmark for evaluating the capabilities of LLMs, as it requires not only natural language understanding but also precise logical reasoning, symbolic manipulation, and deep domain knowledge [108, 120]. The main tasks in this field include arithmetic computation problems [45, 56, 83, 151], math word problems (MWPs) [37, 49, 63, 116], and automated theorem proving (ATP) [2, 152]. These tasks test core competencies such as computational accuracy, deductive reasoning, the ability to model real-world scenarios mathematically, and the application of formal mathematical knowledge. Together, they serve as a rigorous framework for assessing whether LLMs can go beyond surface-level language generation to engage in structured, rule-based problem solving.

However, solving mathematical problems presents unique challenges that distinguish it from other complex domains. One major challenge is multi-step reasoning, as many mathematical tasks require sequential and logically dependent operations where one misstep can derail the entire solution. Another critical challenge is knowledge integration—LLMs must not only understand

abstract principles (e.g., induction), but also domain-specific concepts and theorems, and recognize when and how to apply them, especially in the graduate level and research. This requires retrieving and manipulating domain-specific knowledge that is usually long-tail knowledge for LLMs. A third challenge is result verification, especially in settings like theorem proving, where the correctness of a result can only be confirmed through human evaluations or rigorous formal checking. Recent studies [92] have shown that the current state-of-the-art LLMs generate correct final results but incorrect solutions in mathematical competitions. These challenges demand more than just fluent text generation—they require models to reason with precision, incorporate external tools or knowledge bases, and verify the correctness of multi-step solutions.

To address these challenges, recent research has introduced a range of specialized strategies and systems. For computational ability, models such as MathGLM [153] are pre-trained on progressively complex mathematical problems using curriculum learning, achieving superior accuracy even compared to larger general-purpose models. Prompting-based methods like MathPrompter [56] improve accuracy in arithmetic by generating and cross-verifying multiple solution paths. In reasoning tasks, symbolic integrations [151] with Prolog or proof assistants like Lean (e.g., LeanDojo [148], AlphaProof [22]) help bridge the gap between informal reasoning and formal logic to verify the mathematical reasoning generated by LLMs. In modeling and abstraction, efforts such as symbolic solvers for MWPs and autoformalization benchmarks (e.g., LeanEuclid) [98] illustrate how LLMs can map real-world problems or geometric reasoning into formal mathematical representations. Moreover, retrieval-augmented systems and knowledge-grounded toolkits like DOCMATH-EVAL [179] and LeanDojo [148] show that integrating structured mathematical knowledge significantly boosts performance in tasks that require prior theorems or domain-specific reasoning strategies.

Looking forward, future work in LLM-based mathematical reasoning may focus on deepening the model’s ability to conduct formal reasoning with external feedback and process supervision. Developing hybrid frameworks that combine LLMs with theorem provers, symbolic execution engines, or even formal verification compilers could further enhance result correctness and logical soundness. Additionally, enriching LLMs with structured mathematical knowledge bases, improving their ability to retrieve relevant prior knowledge, and training them on fine-grained proof steps could enhance their capacity for advanced mathematical reasoning. Ultimately, achieving generalizable, verifiable, and domain-aware mathematical reasoning will be key to pushing LLMs closer to human-level mathematical understanding.

4.3 Data Science

This is a field where we perform data analysis and data modeling on a large amount of data [172]. The main tasks in data science revolve around a complex, multi-stage pipeline that includes task understanding, data exploration and analysis, feature engineering, model selection, model training and evaluation. Each of these stages is interrelated, requiring not only technical execution but also careful reasoning and adaptation based on the input data. Unlike domains where problems are well-defined and static, data science demands continuous adjustments to explore the input data.

The unique challenges in this domain stem from its dynamic and data-dependent nature. First, multi-step reasoning is essential, as decisions made in early stages (e.g., feature extraction) significantly affect later ones (e.g., model performance). Second, effective solutions often require domain-specific knowledge that is not easily captured by general-purpose LLMs; integrating such knowledge is vital to handle real-world complexity. Third, verifying the quality of a solution is particularly difficult because the performance depends heavily on the input data rather than just problem descriptions. This makes it challenging to assess modeling strategies.

Current research efforts have made significant progress in addressing these challenges through the development of agent-based systems. Data Interpreter [50] introduces a graph-based agent

that models dependencies between pipeline stages and automates code generation and refinement accordingly. AutoKaggle [76] employs a multi-agent framework—featuring specialized agents such as the Planner, Developer, and Reviewer—to provide end-to-end solutions for tabular data tasks, complete with iterative debugging and testing. Agent K [40] optimizes performance through learned memory mechanisms, using reinforcement signals to retain useful strategies for future tasks. Meanwhile, DS-Agent [44] takes a knowledge-based approach by building a repository of expert insights derived from Kaggle competitions and applying case-based reasoning to generate better solutions. These systems are benchmarked using platforms like DS-Bench [61], MLE-Bench [12], and MLAgentBench [54], which provide structured tasks rooted in real-world ML challenges to evaluate performance across the entire modeling pipeline.

Looking ahead, future research in this domain should focus on enhancing LLMs' ability to reason, adapt, and learn from data-driven experimentation. One key direction is the development of knowledge-enriched modeling agents that can incorporate advanced, domain-specific techniques beyond commonly used libraries. Another promising area is the integration of experiment-driven reasoning, enabling agents to iteratively test, evaluate, and refine their modeling strategies based on actual performance metrics. Finally, training LLMs with chain-of-thought (CoT) mechanisms that incorporate feedback loops from experiment results and domain-specific cues may offer a path toward more intelligent and adaptive data science agents.

4.4 Scientific Research

Artificial intelligence (AI) are increasingly playing a transformative role in scientific research, supporting tasks such as data analysis, simulation, literature review, and idea generation. Their application spans numerous domains, including biology, where tools like AlphaFold [62] and RoseTTAFold [6] revolutionized protein structure prediction; physics, where AI assist in accelerating particle simulations [66]; and astronomy, where they aid in exoplanet detection [94]. In these contexts, LLMs are primarily used in scientific research in two ways: as tools that enhance human research capabilities and as co-creators that propose novel scientific hypotheses or ideas.

Despite these advances, the use of LLMs in scientific discovery presents several notable challenges. First, scientific research typically involves open-ended problems with unclear goals, making it difficult to apply LLMs in ways that guarantee accurate or verifiable solutions. Moreover, scientific research requires deep domain-specific knowledge, and LLMs must effectively leverage this expertise to make reliable predictions. These challenges make it difficult for LLMs to autonomously navigate the full research cycle, especially when the tasks involve open-ended reasoning, abstract synthesis, or interdisciplinary knowledge.

Due to the challenges of scientific research, LLMs are mainly used as tools to assist scientific tasks. For example, LLMs have been employed to accelerate data interpretation in fields like biomedicine and environmental science, where pre-trained models such as BioBERT and SciBERT help contextualize domain-specific data [8, 57, 68, 85]. In simulation and predictive modeling, LLMs have been applied to climate forecasting and molecular modeling, leveraging their world knowledge to support scenarios where traditional simulations may be limited [11]. For literature review and synthesis, LLMs help researchers uncover trends and identify knowledge gaps by summarizing extensive textual corpora [8, 60, 85, 88, 109, 123]. More experimental efforts use LLMs for research idea generation—some studies show that LLMs can generate novel scientific ideas, but also highlight the difficulties of evaluating and selecting high-quality ideas, especially as LLMs themselves are not reliable evaluators [5, 112, 128, 131]. Additionally, agent-based systems like AI Scientist [87] and HEADS [118] demonstrate the feasibility of automating entire research pipelines, from idea generation to simulated peer review, though they fall short of validating these pipelines in solving truly difficult, real-world scientific problems.

Future research will likely focus on improving the reliability and impact of LLMs in scientific discovery by integrating more rigorous evaluation mechanisms and enabling deeper domain-specific reasoning. One key direction is building multi-agent collaboration frameworks that mimic scientific team dynamics to diversify and refine generated ideas. Another is combining LLMs with external tools—such as experiment databases, simulation engines, or formal verification systems—to support result verification and reduce hallucinations. Finally, improving the feedback loop between LLM-generated outputs and human or experimental validation will be critical for realizing LLMs as trusted collaborators in the scientific process. These developments will help move from speculative generation toward verifiable, impactful contributions to scientific research.

5 Discussions and Future Directions

Although there has been notable progress in LLM research for solving complex problems, significant challenges persist. To further enhance LLMs' ability to tackle complex problems, we should focus on improving them from the three key perspectives: multi-step reasoning, knowledge and verification.

Multi-step reasoning. There are two major problems in training LLMs for multi-step reasoning: *data scarcity* and *high computation cost*.

CoT LLMs are typically pre-trained on vast amounts of Internet data, with further enhancement achieved through synthetic data generated by LLMs. However, data scarcity is still a challenge in many specialized domains. For instance, while widely-used programming languages like Python have large code corpora available online, lesser-known languages such as Lean [21] suffer from limited data. Although synthetic data generation via LLMs can improve LLMs, it relies on the base LLMs being well pre-trained for the specific domain. As a result, using data synthesis to improve an LLM's ability to generate code in languages like Lean remains a significant challenge. Similar issues arise in other fields, including mathematics and science. One approach to tackle the data scarcity problem is to develop agents that combine LLMs with custom models specifically trained for the target applications. For example, in formal theorem proving, where data is limited, custom models can help determine the applicability of mathematical strategies (tactics) and assess whether the proof has progressed towards the goal after each step [152]. These models guide LLMs in making informed decisions with reinforcement learning [103, 125, 134], enhancing their reasoning capabilities even in domains with sparse data.

Another problem is *high computation cost*. The inference scaling law has been identified as a way to enhance LLMs' ability to tackle complex problems [17, 32, 119, 129]. By generating numerous reasoning paths, LLMs are more likely to find a path that leads to a solution for highly complex problems at the cost of increased computation. For example, GPT-01 and its successor GPT-03 exhibit significantly higher inference costs compared to GPT-4. Therefore, it is crucial to reduce inference costs. We can address the computational challenges from several angles. First, we can train better LLMs for generation and self-correction to reduce the number of trials to generate reasoning paths/tokens [?]. Second, we should explore various search algorithms to generate reasoning paths more efficiently. Besides best-of-N, we should also explore beam search and Monte Carlo Tree Search. Third, we can reduce the model size of LLMs to speed up inference, which includes techniques like distilling LLMs into smaller models and decoupling knowledge from LLMs to create more compact versions, thus reducing the computational requirements.

Knowledge. Knowledge is fundamental to solving complex problems. Currently, LLMs acquire world knowledge through next-token prediction on massive data during pre-training, which leads to several challenges. LLMs may not reliably memorize world knowledge, particularly long-tail knowledge [121]. As a result, the current LLM cannot work well on domains where training data are scarce. Even if LLMs retain knowledge, they may struggle to recall relevant information when

solving complex problems or LLMs may lack the ability to correctly apply knowledge to solve complex problems.

To effectively leverage knowledge in solving complex problems, one approach is to construct comprehensive knowledge graphs that go beyond traditional triplet-based structures, which contain only entities and relations. In the context of machine learning, a specialized knowledge graph should include not only verbal descriptions of techniques but also their mathematical formulations and corresponding implementation code. Additionally, it should capture relationships between different techniques to facilitate the exploration of diverse approaches and foster innovation in problem-solving. Such a knowledge graph can be systematically built by extracting information from academic papers, technical reports, and textbooks, with careful validation and verification [89]. Once established, this knowledge graph can be utilized in two key ways. First, it can be used to synthesize data for model training, addressing data scarcity challenges. Second, it can support problem-solving during inference through a retrieval-augmented generation (RAG) approach, enabling the model to access and apply relevant knowledge in real time [46].

However, large language models still face challenges in representing and discovering knowledge. Their reliance on chain-of-thought reasoning for complex tasks is hindered by current serialized techniques, which struggle to structurally capture domain-specific knowledge and logic (workflow) while providing limited support for human intervention [95]. Additionally, large language models encounter difficulties in balancing innovative knowledge discovery with logical credibility, often leading to hallucinatory outputs. Compounding these issues, the large language model's dynamic adaptation capabilities are insufficient to keep pace with rapidly changing environments, as delayed knowledge updates can render decision-making strategies ineffective. These interconnected challenges underscore the need for further research into improving thought process modeling, enhancing domain knowledge discovery, updating (editing) [29, 130, 156, 170], and developing more robust adaptation mechanisms for complex problem solving [33, 143].

Evaluation. The current LLM research, such as OpenAI o1, focuses on complex problems whose final results can be easily verified, such as competitive coding and mathematical reasoning. However, practical applications have much more complex requirements that complicate the verification of final results. First, some applications not only require correctness of solutions but also to achieve efficiency or yield better accuracy. In machine learning tasks, for example, while baseline methods like random forecasts or multilayer perceptrons can be considered as "correct" solutions, they may not meet the desired performance, and more effective solutions are preferred. Furthermore, problems in many applications are difficult to define comprehensively. Again using machine learning tasks as an example. Both the task description and the distribution of input data are essential for designing an effective solution. However, conveying the data distribution of the input data to an LLM is challenging. In addition, in certain scientific fields, such as drug discovery, climate modeling, or social sciences, validation of results often requires extensive experimental testing, replication, or further theoretical analysis to confirm their accuracy and reliability.

These challenges emphasize the need for robust evaluation frameworks and the integration of domain-specific expertise to ensure the reliability of LLM-generated outputs. To enhance the credibility of LLM outputs, it is crucial to employ multiple evaluation approaches. Consider machine learning tasks as an example—there are several ways to assess the effectiveness of a machine learning algorithm: First, the algorithm's performance can be evaluated by comparing it with previously published results, such as academic papers and technical reports. Secondly, an LLM-based evaluator can be utilized to assess the quality of a solution. To improve its accuracy, data analysis should be conducted to extract comprehensive insights from the input data and feed them to LLM. Thirdly, implementing the machine learning algorithm and conducting experiments provides an empirical

assessment of its effectiveness. Fourth, for certain machine learning algorithms, we can perform some theoretical analysis on the algorithm, which is further verified by symbolic verification tools like Lean, ensuring rigorous validation of the algorithm's correctness and effectiveness. By combining all these different evaluation approaches, we can potentially perform thorough evaluation of machine learning algorithms. We believe similar evaluation principles (the combinations of LLM-based evaluations, empirical experiments, theoretical evaluations) can also be applied to other domains.

6 Related Works

Several survey papers have addressed LLM-based reasoning. Early works, such as Qiao et al. [105] and Huang and Chang [51], provide an overview of LLM-based reasoning, which is crucial for complex problem-solving. However, these surveys primarily focus on the initial developments in this area. With the release of GPT-01 [99], the effectiveness of LLM-based reasoning was significantly demonstrated. Following this, numerous studies have explored the potential mechanisms behind GPT-01. For example, Zeng et al. [160] and Xu et al. [142] delve into techniques that could enable 01-like reasoning, particularly through reinforcement learning. In contrast, this work takes a broader perspective, addressing all the different capabilities needed for complex problem-solving, rather than focusing solely on reasoning.

Numerous survey papers focus on specific domains of LLM-based reasoning. For instance, Yang et al. [147] examines the progress, challenges, and future directions in formal mathematical reasoning. Eger et al. [31] explores recent advances in using LLMs to support scientific research, covering applications such as literature search, idea generation, text and multimodal content (e.g., scientific figures and diagrams) generation, and AI-based peer review. Ahn et al. [3] provides an overview of various types of mathematical reasoning using LLMs. However, this work does not address 01-like techniques. Meanwhile, Li et al. [75] focuses on theorem proving within mathematical reasoning. Instead of solely relying on LLMs, this survey breaks down theorem proving into multiple components and discusses the application of various deep learning techniques for each aspect.

7 Conclusions

In this survey paper, we define complex problem-solving from the perspectives of both cognitive science and computational theory, and analyze the characteristics of different complex problems. We then investigate significant advancements in large language models (LLMs), with a focus on chain-of-thought reasoning and agent-based approaches in the context of complex problem-solving. We discuss how data synthesis and reinforcement learning have enhanced LLMs' abilities for multi-step reasoning. Additionally, we explore how the agent-based approach enables AI systems to harness external knowledge, tools for execution and result verification. However, we also examine the limitations of these methods when applied to different types of complex problems.

References

- [1] 2025. <https://openai.com/index/introducing-deep-research/>
- [2] Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large Language Models for Mathematical Reasoning: Progresses and Challenges. *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop* (2024), 225–237. St. Julian’s, Malta.
- [3] Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157* (2024).
- [4] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* (2016).

- [5] Jinheon Baek, Sujay Kumar Jauhar, Silviu Cucerzan, and Sung Ju Hwang. 2024. ResearchAgent: Iterative Research Idea Generation over Scientific Literature with Large Language Models. arXiv:2404.07738 [cs.CL] <https://arxiv.org/abs/2404.07738>
- [6] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N. Kinch, R. Dustin Schaeffer, Claudia Millán, Hahnbeom Park, Carson Adams, Caleb R. Glassman, Andy DeGiovanni, Jose H. Pereira, Andria V. Rodrigues, Alberdina A. van Dijk, Ana C. Ebrecht, Diederik J. Opperman, Theo Sagmeister, Christoph Buhlheller, Tea Pavkov-Keller, Manoj K. Rathinaswamy, Udit Dalwadi, Calvin K. Yip, John E. Burke, K. Christopher Garcia, Nick V. Grishin, Paul D. Adams, Randy J. Read, and David Baker. 2021. Accurate prediction of protein structures and interactions using a three-track neural network. *Science* 373, 6557 (2021), 871–876. <https://doi.org/10.1126/science.abj8754>
- [7] Hritik Bansal, Arian Hosseini, Rishabh Agarwal, Vinh Q. Tran, and Mehran Kazemi. 2024. Smaller, Weaker, Yet Better: Training LLM Reasoners via Compute-Optimal Sampling. arXiv:2408.16737 [cs.CL]
- [8] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676* (2019).
- [9] Léonard Boussoux, Jacqueline N Lane, Miaomiao Zhang, Vladimir Jacimovic, and Karim R Lakhani. 2024. The crowdless future? Generative AI and creative problem-solving. *Organization Science* 35, 5 (2024), 1589–1607.
- [10] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large Language Monkeys: Scaling Inference Compute with Repeated Sampling. arXiv:2407.21787 [cs.LG]
- [11] Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).
- [12] Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lillian Weng, and Aleksander Madry. 2024. MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering. *CoRR* abs/2410.07095 (2024). <https://doi.org/10.48550/ARXIV.2410.07095> arXiv:2410.07095
- [13] Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wangxiang Che. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567* (2025).
- [14] Wenhui Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks. *Trans. Mach. Learn. Res.* 2023 (2023). <https://openreview.net/forum?id=YfZ4ZPt8zd>
- [15] Xinyun Chen, Maxwell Lin, Nathanael Schärlí, and Denny Zhou. 2023. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128* (2023).
- [16] Xinyun Chen, Chang Liu, and Dawn Song. 2018. Tree-to-tree neural networks for program translation. *Advances in neural information processing systems* 31 (2018).
- [17] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuizhi Liu, Mengfei Zhou, Zhusongsheng Zhang, et al. 2024. Do not think that much for 2+3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187* (2024).
- [18] Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *Proceedings of the ACM Web conference 2022*. 2778–2788.
- [19] I-Chun Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, and Pengfei Liu. 2023. FacTool: Factuality Detection in Generative AI - A Tool Augmented Framework for Multi-Task and Multi-Domain Scenarios. *CoRR* abs/2307.13528 (2023). <https://doi.org/10.48550/ARXIV.2307.13528> arXiv:2307.13528
- [20] 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. 2021. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168 [cs.LG]
- [21] Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. 2015. The Lean Theorem Prover (System Description). In *Automated Deduction - CADE-25*, Amy P. Felty and Aart Middeldorp (Eds.). Springer International Publishing, Cham, 378–388.
- [22] DeepMind. 2024. AI solves IMO problems at silver medal level. <https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/> Accessed: 2024-06-17.
- [23] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu,

- Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhenwen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 [cs.CL] <https://arxiv.org/abs/2501.12948>
- [24] Linyi Ding, Sizhe Zhou, Jinfeng Xiao, and Jiawei Han. 2024. Automated construction of theme-specific knowledge graphs. *arXiv preprint arXiv:2404.19146* (2024).
- [25] Yangruibo Ding, Zijian Wang, Wasi Uddin Ahmad, Murali Krishna Ramanathan, Ramesh Nallapati, Parminder Bhatia, Dan Roth, and Bing Xiang. 2022. Cocomic: Code completion by jointly modeling in-file and cross-file context. *arXiv preprint arXiv:2212.10007* (2022).
- [26] Hyo Jin Do, Rachel Ostrand, Justin D Weisz, Casey Dugan, Prasanna Sattigeri, Dennis Wei, Keerthiram Murugesan, and Werner Geyer. 2024. Facilitating Human-LLM Collaboration through Factuality Scores and Source Attributions. *arXiv preprint arXiv:2405.20434* (2024).
- [27] Lun Du, Xiaozhou Shi, Yanlin Wang, Ensheng Shi, Shi Han, and Dongmei Zhang. 2021. Is a single model enough? mucos: A multi-model ensemble learning approach for semantic code search. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2994–2998.
- [28] Mingzhe Du, Anh Tuan Luu, Bin Ji, and See-Kiong Ng. 2024. Mercury: An efficiency benchmark for llm code synthesis. *arXiv preprint arXiv:2402.07844* (2024).
- [29] Yiming Du, Wenyu Huang, Danna Zheng, Zhaowei Wang, Sebastien Montella, Mirella Lapata, Kam-Fai Wong, and Jeff Z Pan. 2025. Rethinking Memory in AI: Taxonomy, Operations, Topics, and Future Directions. *arXiv preprint arXiv:2505.00675* (2025).
- [30] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitan-sky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130* (2024).
- [31] Steffen Eger, Yong Cao, Jennifer D’Souza, Andreas Geiger, Christian Greisinger, Stephanie Gross, Yufang Hou, Brigitte Krenn, Anne Lauscher, Yizhi Li, Chenghua Lin, Nafise Sadat Moosavi, Wei Zhao, and Tristan Miller. 2025. Transforming Science with Large Language Models: A Survey on AI-assisted Scientific Discovery, Experimentation, Content Generation, and Evaluation. arXiv:2502.05151 [cs.CL] <https://arxiv.org/abs/2502.05151>
- [32] Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025. Efficient Reasoning Models: A Survey. *arXiv preprint arXiv:2504.10903* (2025).
- [33] Dayuan Fu, Keqing He, Yejie Wang, Wentao Hong, Zhuoma Gongque, Weihao Zeng, Wei Wang, Jingang Wang, Xunliang Cai, and Weiran Xu. 2025. AgentRefine: Enhancing Agent Generalization through Refinement Tuning. *arXiv preprint arXiv:2501.01702* (2025).
- [34] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided Language Models. In *International Conference on Machine Learning, ICML 2023, 23–29 July 2023, Honolulu, Hawaii, USA (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 10764–10799. <https://proceedings.mlr.press/v202/gao23f.html>
- [35] Shuzheng Gao, Cuiyun Gao, Wenchao Gu, and Michael Lyu. 2024. Search-based llms for code optimization. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 254–266.
- [36] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* 2 (2023).
- [37] Vedant Gaur and Nikunj Saunshi. 2023. Reasoning in Large Language Models Through Symbolic Math Word Problems. In *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics,

- Toronto, Canada, 5889–5903. <https://doi.org/10.18653/v1/2023.findings-acl.364>
- [38] Zorik Gekhman, Gal Yona, Roee Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. 2024. Does Fine-Tuning LLMs on New Knowledge Encourage Hallucinations?. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 7765–7784. <https://doi.org/10.18653/v1/2024.emnlp-main.444>
- [39] Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, et al. 2024. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *arXiv preprint arXiv:2411.04872* (2024).
- [40] Antoine Grosnit, Alexandre Maraval, James Doran, Giuseppe Paolo, Albert Thomas, Refinath Shahul Hameed Nabeezath Beevi, Jonas Gonzalez, Khyati Khandelwal, Ignacio Iacobacci, Abdelhakim Benechehab, Hamza Cherkaoui, Youssef Attia El-Hili, Kun Shao, Jianye Hao, Jun Yao, Balazs Kegl, Haitham Bou-Ammar, and Jun Wang. 2024. Large Language Models Orchestrating Structured Reasoning Achieve Kaggle Grandmaster Level. *arXiv:2411.03562 [cs.LG]* <https://arxiv.org/abs/2411.03562>
- [41] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. 2025. A Survey on LLM-as-a-Judge. *arXiv:2411.15594 [cs.CL]* <https://arxiv.org/abs/2411.15594>
- [42] Zhouchong Gu, Xiaoxuan Zhu, Haoming Ye, Lin Zhang, Jianchen Wang, Yixin Zhu, Sihang Jiang, Zhuozhi Xiong, Zihan Li, Weijie Wu, et al. 2024. Xiezhi: An ever-updating benchmark for holistic domain knowledge evaluation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 18099–18107.
- [43] Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. 2024. DeepSeek-Coder: When the Large Language Model Meets Programming—The Rise of Code Intelligence. *arXiv preprint arXiv:2401.14196* (2024).
- [44] Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. 2024. DS-Agent: Automated Data Science by Empowering Large Language Models with Case-Based Reasoning. *arXiv:2402.17453 [cs.LG]* <https://arxiv.org/abs/2402.17453>
- [45] Yingdi Guo. 2023. ArthModel: Enhance Arithmetic Skills to Large Language Model. *arXiv preprint arXiv:2311.18609* (2023).
- [46] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation. (2024).
- [47] Joe B Hakim, Jeffery L Painter, Darmendra Ramcharan, Vijay Kara, Greg Powell, Paulina Sobczak, Chiho Sato, Andrew Bate, and Andrew Beam. 2024. The Need for Guardrails with Large Language Models in Medical Safety-Critical Settings: An Artificial Intelligence Application in the Pharmacovigilance Ecosystem. *arXiv:2407.18322 [cs.CL]* <https://arxiv.org/abs/2407.18322>
- [48] Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. 2024. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309* (2024).
- [49] Yue He et al. 2023. Solving Math Word Problems by Combining Language Models With Symbolic Solvers. *arXiv preprint arXiv:2304.09102* (2023). <https://arxiv.org/abs/2304.09102>
- [50] Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Wenyi Wang, Xiangru Tang, Xiangtao Lu, Xiawu Zheng, Xinbing Liang, Yaying Fei, Yuheng Cheng, Zongze Xu, and Chenglin Wu. 2024. Data Interpreter: An LLM Agent For Data Science. *arXiv:2402.18679 [cs.AI]*
- [51] Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards Reasoning in Large Language Models: A Survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 1049–1065. <https://doi.org/10.18653/v1/2023.findings-acl.67>
- [52] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large Language Models Cannot Self-Correct Reasoning Yet. In *The Twelfth International Conference on Learning Representations*.
- [53] Kai Huang, Xiangxin Meng, Jian Zhang, Yang Liu, Wenjie Wang, Shuhao Li, and Yuqing Zhang. 2023. An Empirical Study on Fine-Tuning Large Language Models of Code for Automated Program Repair. *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (2023), 1162–1174. <https://api.semanticscholar.org/CorpusID:265054960>
- [54] Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. 2024. MLAGentBench: Evaluating Language Agents on Machine Learning Experimentation. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net. <https://openreview.net/forum?id=1Fs1LvjYQW>

- [55] Zilin Huang, Zihao Sheng, Chengyuan Ma, and Sikai Chen. 2024. Human as AI mentor: Enhanced human-in-the-loop reinforcement learning for safe and efficient autonomous driving. *Communications in Transportation Research* 4 (Dec. 2024), 100127. <https://doi.org/10.1016/j.commtr.2024.100127>
- [56] Shima Imani, Liang Du, and Harsh Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models. *arXiv preprint arXiv:2303.05398* (2023).
- [57] Jacqueline A Jansen, Artür Manukyan, Nour Al Khoury, and Altuna Akalin. 2023. Leveraging large language models for data analysis automation. *bioRxiv* (2023), 2023–12.
- [58] Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024. A Survey on Large Language Models for Code Generation. *arXiv preprint arXiv:2406.00515* (2024).
- [59] Nan Jiang, Kevin Liu, Thibaud Lutellier, and Lin Tan. 2023. Impact of Code Language Models on Automated Program Repair. *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)* (2023), 1430–1442. <https://api.semanticscholar.org/CorpusID:256808267>
- [60] Qiao Jin, Robert Leaman, and Zhiyong Lu. 2023. PubMed and Beyond: Biomedical Literature Search in the Age of Artificial Intelligence. *arXiv preprint arXiv:2307.09683* (2023).
- [61] Liqiang Jing, Zhehui Huang, Xiaoyang Wang, Wenlin Yao, Wenhao Yu, Kaixin Ma, Hongming Zhang, Xinya Du, and Dong Yu. 2024. DSbench: How Far Are Data Science Agents to Becoming Data Science Experts? *CoRR* abs/2409.07703 (2024). <https://doi.org/10.48550/ARXIV.2409.07703> arXiv:2409.07703
- [62] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvanakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andrew J Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikолов, Rishabh Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michał Zieliński, Martin Steinegger, Michałina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature* 596, 7873 (2021), 583–589. <https://doi.org/10.1038/s41586-021-03819-2>
- [63] Lee Kim et al. 2023. Can LLMs Solve Longer Math Word Problems Better? *AI Models* (2023). <https://arxiv.org/abs/2402.17916>
- [64] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2024. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) (*NIPS* '22). Article 1613, 15 pages.
- [65] Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Sharqi Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. 2024. Training Language Models to Self-Correct via Reinforcement Learning. *arXiv:2409.12917 [cs.LG]*
- [66] Krishna Kumar and Yongjin Choi. 2023. Accelerating Particle and Fluid Simulations with Differentiable Graph Networks for Solving Forward and Inverse Problems. *arXiv:2309.13348 [physics.geo-ph]* <https://arxiv.org/abs/2309.13348>
- [67] JE Laird, JF Lehman, and P Rosenbloom. 1996. A gentle introduction to Soar, an architecture for human cognition. *QYLWDWLRQ WR & RJQLWLYH 6FLHQFH* (1996).
- [68] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36, 4 (2020), 1234–1240.
- [69] Jia Li, Ge Li, Yongmin Li, and Zhi Jin. 2025. Structured chain-of-thought prompting for code generation. *ACM Transactions on Software Engineering and Methodology* 34, 2 (2025), 1–23.
- [70] Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. 2025. ScreenSpot-Pro: GUI Grounding for Professional High-Resolution Computer Use. https://likaixin2000.github.io/papers/ScreenSpot_Pro.pdf Preprint.
- [71] Kaixin Li, Yuchen Tian, Qisheng Hu, Ziyang Luo, and Jing Ma. 2024. Mmcode: Evaluating multi-modal code large language models with visually rich programming problems. *arXiv preprint arXiv:2404.09486* (2024).
- [72] Xiaonan Li and Xipeng Qiu. 2023. MoT: Memory-of-Thought Enables ChatGPT to Self-Improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 6354–6374. <https://doi.org/10.18653/V1/2023.EMNLP-MAIN.392>
- [73] Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2024. Chain-of-Knowledge: Grounding Large Language Models via Dynamic Knowledge Adapting over Heterogeneous Sources. In *The Twelfth International Conference on Learning Representations*.
- [74] Xuefeng Li, Haoyang Zou, and Pengfei Liu. 2025. Limr: Less is more for rl scaling. *arXiv preprint arXiv:2502.11886* (2025).

- [75] Zhaoyu Li, Jialiang Sun, Logan Murphy, Qidong Su, Zenan Li, Xian Zhang, Kaiyu Yang, and Xujie Si. 2024. A Survey on Deep Learning for Theorem Proving. *arXiv:2404.09939 [cs.AI]* <https://arxiv.org/abs/2404.09939>
- [76] Ziming Li, Qianbo Zang, David Ma, Jiawei Guo, Tuney Zheng, Minghao Liu, Xinyao Niu, Yue Wang, Jian Yang, Jiaheng Liu, Wanjun Zhong, Wangchunshu Zhou, Wenhao Huang, and Ge Zhang. 2024. AutoKaggle: A Multi-Agent Framework for Autonomous Data Science Competitions. *CoRR abs/2410.20424* (2024). <https://doi.org/10.48550/ARXIV.2410.20424> arXiv:2410.20424
- [77] Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, et al. 2025. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419* (2025).
- [78] Lei Liang, Mengshu Sun, Zhengke Gui, Zhongshu Zhu, Zhouyu Jiang, Ling Zhong, Yuan Qu, Peilong Zhao, Zhongpu Bo, Jin Yang, et al. 2024. Kag: Boosting llms in professional domains via knowledge augmented generation. *arXiv preprint arXiv:2409.13731* (2024).
- [79] Ming Liang, Xiaoheng Xie, Gehao Zhang, Xunjin Zheng, Peng Di, Hongwei Chen, Chengpeng Wang, Gang Fan, et al. 2024. REPOFUSE: Repository-Level Code Completion with Fused Dual Context. *arXiv preprint arXiv:2402.14323* (2024).
- [80] Antonio Lieto, Federico Perrone, Gian Luca Pozzato, and Eleonora Chiodino. 2019. Beyond subgoaling: A dynamic knowledge generation framework for creative problem solving in cognitive architectures. *Cognitive Systems Research* 58 (2019), 305–316.
- [81] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's Verify Step by Step. In *The Twelfth International Conference on Learning Representations*.
- [82] Jiate Liu, Yiqin Zhu, Kaiwen Xiao, Qiang Fu, Xiao Han, Wei Yang, and Deheng Ye. 2023. Rltf: Reinforcement learning from unit test feedback. *arXiv preprint arXiv:2307.04349* (2023).
- [83] Tiedong Liu and Bryan Kian Hsiang Low. 2023. Goat: Fine-tuned llama outperforms gpt-4 on arithmetic tasks. *arXiv preprint arXiv:2305.14201* (2023).
- [84] Wei Liu, Ailun Yu, Daoguang Zan, Bo Shen, Wei Zhang, Haiyan Zhao, Zhi Jin, and Qianxiang Wang. 2024. GraphCoder: Enhancing Repository-Level Code Completion via Code Context Graph-based Retrieval and Language Model. *arXiv preprint arXiv:2406.07003* (2024).
- [85] Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Dan S Weld. 2019. S2ORC: The semantic scholar open research corpus. *arXiv preprint arXiv:1911.02782* (2019).
- [86] Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtiar, Jiawei Liu, Yuxiang Wei, et al. 2024. Starcoder 2 and the stack v2: The next generation. *arXiv preprint arXiv:2402.19173* (2024).
- [87] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery. *arXiv:2408.06292 [cs.AI]* <https://arxiv.org/abs/2408.06292>
- [88] Xiaoliang Luo, Akilles Rechardt, Guangzhi Sun, Kevin K Nejad, Felipe Yáñez, Bati Yilmaz, Kangjoo Lee, Alexandra O Cohen, Valentina Borghesani, Anton Pashkov, et al. 2024. Large language models surpass human experts in predicting neuroscience results. *Nature Human Behaviour* (2024), 1–11.
- [89] Yujie Luo, Xiangyuan Ru, Kangwei Liu, Lin Yuan, Mengshu Sun, Ningyu Zhang, Lei Liang, Zhiqiang Zhang, Jun Zhou, Lanning Wei, et al. 2024. OneKE: A Dockerized Schema-Guided LLM Agent-based Knowledge Extraction System. *arXiv preprint arXiv:2412.20005* (2024).
- [90] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Dixin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568* (2023).
- [91] Yingwei Ma, Rongyu Cao, Yongchang Cao, Yue Zhang, Jue Chen, Yibo Liu, Yuchen Liu, Binhua Li, Fei Huang, and Yongbin Li. 2024. Lingma SWE-GPT: An Open Development-Process-Centric Language Model for Automated Software Improvement. *arXiv preprint arXiv:2411.00622* (2024).
- [92] Hamed Mahdavi, Alireza Hashemi, Majid Daliri, Pegah Mohammadipour, Alireza Farhadi, Samira Malek, Yekta Yazdanifard, Amir Khasahmadi, and Vasant Honavar. 2025. Brains vs. Bytes: Evaluating LLM Proficiency in Olympiad Mathematics. *arXiv preprint arXiv:2504.01995* (2025).
- [93] R. Thomas McCoy, Shunyu Yao, Dan Friedman, Matthew Hardy, and Thomas L. Griffiths. 2023. Embers of Autoregression: Understanding Large Language Models Through the Problem They are Trained to Solve. *arXiv:2309.13638 [cs.CL]* <https://arxiv.org/abs/2309.13638>
- [94] Qinghai Miao and Fei-Yue Wang. 2024. AI for Astronomy. In *Artificial Intelligence for Science (AI4S) Frontiers and Perspectives Based on Parallel Intelligence*. Springer, 93–103.
- [95] So Yeon Min, Yue Wu, Jimin Sun, Max Kaufmann, Fahim Tajwar, Yonatan Bisk, and Ruslan Salakhutdinov. 2025. Self-Regulation and Requesting Interventions. *arXiv preprint arXiv:2502.04576* (2025).

- [96] Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, et al. 2024. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. *arXiv preprint arXiv:2412.09413* (2024).
- [97] Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models. *arXiv:2410.05229 [cs.LG]* <https://arxiv.org/abs/2410.05229>
- [98] Logan Murphy, Kaiyu Yang, Jialiang Sun, Zhaoyu Li, Anima Anandkumar, and Xujie Si. 2024. Autoformalizing Euclidean Geometry. *arXiv preprint arXiv:2405.17216* (2024).
- [99] OpenAI. 2024. OpenAI o1 Guide: How It Works, Use Cases, API & More. (2024). <https://www.datacamp.com> Accessed: 2024-10-27.
- [100] Rangeet Pan, Ali Reza Ibrahimzada, Rahul Krishna, Divya Sankar, Lambert Pouguem Wassi, Michele Merler, Boris Sobolev, Raju Pavuluri, Saurabh Sinha, and Reyhaneh Jabbarvand. 2023. Understanding the effectiveness of large language models in code translation. *arXiv preprint arXiv:2308.03109* (2023).
- [101] Xinyu Pang, Ruixin Hong, Zhanke Zhou, Fangrui Lv, Xinwei Yang, Zhilong Liang, Bo Han, and Changshui Zhang. 2024. Physics Reasoner: Knowledge-Augmented Reasoning for Solving Physics Problems with Large Language Models. *arXiv preprint arXiv:2412.13791* (2024).
- [102] Yun Peng, Shuzheng Gao, Cuiyun Gao, Yintong Huo, and Michael Lyu. 2024. Domain knowledge matters: Improving prompts with fix templates for repairing python type errors. In *Proceedings of the 46th ieee/acm international conference on software engineering*. 1–13.
- [103] Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. ToolRL: Reward is All Tool Learning Needs. *arXiv preprint arXiv:2504.13958* (2025).
- [104] Shuofei Qiao, Runnan Fang, Zhisong Qiu, Xiaobin Wang, Ningyu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024. Benchmarking Agentic Workflow Generation. *arXiv preprint arXiv:2410.07869* (2024).
- [105] Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanchi Tan, Fei Huang, and Huajun Chen. 2023. Reasoning with Language Model Prompting: A Survey. *arXiv:2212.09597 [cs.CL]* <https://arxiv.org/abs/2212.09597>
- [106] Mengjie Ren, Boxi Cao, Hongyu Lin, Cao Liu, Xianpei Han, Ke Zeng, Guanglu Wan, Xunliang Cai, and Le Sun. 2024. Learning or Self-aligning? Rethinking Instruction Fine-tuning. *arXiv:2402.18243 [cs.CL]* <https://arxiv.org/abs/2402.18243>
- [107] Francisco Ribeiro, José Nuno Macedo, Kanae Tsushima, Rui Abreu, and João Saraiva. 2023. GPT-3-Powered Type Error Debugging: Investigating the Use of Large Language Models for Code Repair. *Proceedings of the 16th ACM SIGPLAN International Conference on Software Language Engineering* (2023). <https://api.semanticscholar.org/CorpusID:264306954>
- [108] 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. 2024. Mathematical discoveries from program search with large language models. *Nature* 625, 7995 (2024), 468–475.
- [109] Dmitry Scherbakov, Nina Hubig, Vinita Jansari, Alexander Bakumenko, and Leslie A Lenert. 2024. The emergence of Large Language Models (LLM) as a tool in literature reviews: an LLM automated systematic review. *arXiv preprint arXiv:2409.04600* (2024).
- [110] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärlí, and Denny Zhou. 2023. Large Language Models Can Be Easily Distracted by Irrelevant Context. *arXiv:2302.00093 [cs.CL]* <https://arxiv.org/abs/2302.00093>
- [111] Alexander Shypula, Aman Madaan, Yimeng Zeng, Uri Alon, Jacob Gardner, Milad Hashemi, Graham Neubig, Parthasarathy Ranganathan, Osbert Bastani, and Amir Yazdanbakhsh. 2023. Learning performance-improving code edits. *arXiv preprint arXiv:2302.07867* (2023).
- [112] Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. 2024. Can LLMs Generate Novel Research Ideas? A Large-Scale Human Study with 100+ NLP Researchers. *arXiv:2409.04109 [cs.CL]*
- [113] Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron T Parisi, Abhishek Kumar, Alexander A Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Fathy Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshitij Mahajan, Laura A Culp, Lechao Xiao, Maxwell Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. 2024. Beyond Human Data: Scaling Self-Training for Problem-Solving with Language Models. *Transactions on Machine Learning Research* (2024).
- [114] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters. *arXiv:2408.03314 [cs.LG]*

- [115] Zirui Song, Bin Yan, Yuhuan Liu, Miao Fang, Mingzhe Li, Rui Yan, and Xiuying Chen. 2025. Injecting Domain-Specific Knowledge into Large Language Models: A Comprehensive Survey. *arXiv preprint arXiv:2502.10708* (2025).
- [116] Kv Aditya Srivatsa and Ekaterina Kochmar. 2024. What Makes Math Word Problems Challenging for LLMs?. In *Findings of the Association for Computational Linguistics: NAACL 2024*. Association for Computational Linguistics, Mexico City, Mexico, 1138–1148. <https://doi.org/10.18653/v1/2024.findings-naacl.72>
- [117] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems* 33 (2020), 3008–3021.
- [118] Haoyang Su, Renqi Chen, Shixiang Tang, Xinzheng Zheng, Jingzhe Li, Zhenfei Yin, Wanli Ouyang, and Nanqing Dong. 2024. Two Heads Are Better Than One: A Multi-Agent System Has the Potential to Improve Scientific Idea Generation. *arXiv:2410.09403 [cs.AI]* <https://arxiv.org/abs/2410.09403>
- [119] Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Hanjie Chen, Xia Hu, et al. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419* (2025).
- [120] Haoxiang Sun, Yingqian Min, Zhipeng Chen, Wayne Xin Zhao, Zheng Liu, Zhongyuan Wang, Lei Fang, and Ji-Rong Wen. 2025. Challenging the Boundaries of Reasoning: An Olympiad-Level Math Benchmark for Large Language Models. *arXiv preprint arXiv:2503.21380* (2025).
- [121] Kai Sun, Yifan Ethan Xu, Hanwen Zha, Yue Liu, and Xin Luna Dong. 2024. Head-to-Tail: How Knowledgeable are Large Language Models (LLMs)? A.K.A. Will LLMs Replace Knowledge Graphs? *arXiv:2308.10168 [cs.CL]* <https://arxiv.org/abs/2308.10168>
- [122] Weisong Sun, Chunrong Fang, Yifei Ge, Yuling Hu, Yuchen Chen, Quanjun Zhang, Xiuting Ge, Yang Liu, and Zhenyu Chen. 2024. A Survey of Source Code Search: A 3-Dimensional Perspective. *ACM Trans. Softw. Eng. Methodol.* 33, 6, Article 166 (June 2024), 51 pages. <https://doi.org/10.1145/3656341>
- [123] Teo Susnjak, Peter Hwang, Napoleon H Reyes, Andre LC Barczak, Timothy R McIntosh, and Surangika Ranathunga. 2024. Automating research synthesis with domain-specific large language model fine-tuning. *arXiv preprint arXiv:2404.08680* (2024).
- [124] Yuchen Tian, Weixiang Yan, Qian Yang, Qian Chen, Wen Wang, Ziyang Luo, and Lei Ma. 2024. CodeHalu: Code Hallucinations in LLMs Driven by Execution-based Verification. *arXiv preprint arXiv:2405.00253* (2024).
- [125] Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. 2025. OTC: Optimal Tool Calls via Reinforcement Learning. *arXiv preprint arXiv:2504.14870* (2025).
- [126] Keheng Wang, Feiyu Duan, Sirui Wang, Peiguang Li, Yunsen Xian, Chuantao Yin, Wenge Rong, and Zhang Xiong. 2023. Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering. *arXiv preprint arXiv:2308.13259* (2023).
- [127] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. 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)*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.).
- [128] Qingyun Wang, Doug Downey, Heng Ji, and Tom Hope. 2024. SciMON: Scientific Inspiration Machines Optimized for Novelty. *arXiv:2305.14259 [cs.CL]* <https://arxiv.org/abs/2305.14259>
- [129] Rui Wang, Hongru Wang, Boyang Xue, Jianhui Pang, Shudong Liu, Yi Chen, Jiahao Qiu, Derek Fai Wong, Heng Ji, and Kam-Fai Wong. 2025. Harnessing the Reasoning Economy: A Survey of Efficient Reasoning for Large Language Models. *arXiv preprint arXiv:2503.24377* (2025).
- [130] Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2024. Knowledge editing for large language models: A survey. *Comput. Surveys* 57, 3 (2024), 1–37.
- [131] Wenxiao Wang, Lihui Gu, Liye Zhang, Yunxiang Luo, Yi Dai, Chen Shen, Liang Xie, Binbin Lin, Xiaofei He, and Jieping Ye. 2024. SciPIP: An LLM-based Scientific Paper Idea Proposer. *arXiv:2410.23166 [cs.CL]* <https://arxiv.org/abs/2410.23166>
- [132] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*.
- [133] Yanlin Wang, Lun Du, Ensheng Shi, Yuxuan Hu, Shi Han, and Dongmei Zhang. 2020. Cocogum: Contextual code summarization with multi-relational gnn on umls. *Microsoft, Tech. Rep. MSR-TR-2020-16* (2020).
- [134] Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, et al. 2025. RAGEN: Understanding Self-Evolution in LLM Agents via Multi-Turn Reinforcement Learning. *arXiv preprint arXiv:2504.20073* (2025).

- [135] Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2024. Agent Workflow Memory. *CoRR* abs/2409.07429 (2024). <https://doi.org/10.48550/ARXIV.2409.07429> arXiv:2409.07429
- [136] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2024. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) (NIPS '22). Article 1800, 14 pages.
- [137] Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2023. Magicoder: Source code is all you need. *arXiv preprint arXiv:2312.02120* (2023).
- [138] Sean Welleck, Jiacheng Liu, Ximing Lu, Hannaneh Hajishirzi, and Yejin Choi. 2022. NaturalProver: Grounded Mathematical Proof Generation with Language Models. *arXiv:2205.12910 [cs.CL]* <https://arxiv.org/abs/2205.12910>
- [139] Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2023. Generating Sequences by Learning to Self-Correct. In *The Eleventh International Conference on Learning Representations*.
- [140] Jingda Wu, Zhiyu Huang, Chao Huang, Zhongxu Hu, Peng Hang, Yang Xing, and Chen Lv. 2021. Human-in-the-Loop Deep Reinforcement Learning with Application to Autonomous Driving. *arXiv:2104.07246 [cs.RO]* <https://arxiv.org/abs/2104.07246>
- [141] Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. 2024. Large language models for generative information extraction: A survey. *Frontiers of Computer Science* 18, 6 (2024), 186357.
- [142] Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Chenyang Shao, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Yu Li, Jie Feng, Chen Gao, and Yong Li. 2025. Towards Large Reasoning Models: A Survey of Reinforced Reasoning with Large Language Models. *arXiv:2501.09686 [cs.AI]* <https://arxiv.org/abs/2501.09686>
- [143] Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. 2025. An illusion of progress? assessing the current state of web agents. *arXiv preprint arXiv:2504.01382* (2025).
- [144] Weixiang Yan, Haitian Liu, Yunkun Wang, Yunzhe Li, Qian Chen, Wen Wang, Tingyu Lin, Weishan Zhao, Li Zhu, Hari Sundaram, et al. 2023. Codescope: An execution-based multilingual multitask multidimensional benchmark for evaluating llms on code understanding and generation. *arXiv preprint arXiv:2311.08588* (2023).
- [145] Weixiang Yan, Yuchen Tian, Yunzhe Li, Qian Chen, and Wen Wang. 2023. Codetransocean: A comprehensive multilingual benchmark for code translation. *arXiv preprint arXiv:2310.04951* (2023).
- [146] John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793* (2024).
- [147] Kaiyu Yang, Gabriel Poesia, Jingxuan He, Wenda Li, Kristin Lauter, Swarat Chaudhuri, and Dawn Song. 2024. Formal Mathematical Reasoning: A New Frontier in AI. *arXiv:2412.16075 [cs.AI]* <https://arxiv.org/abs/2412.16075>
- [148] Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J Prenger, and Animashree Anandkumar. 2024. Leandojo: Theorem proving with retrieval-augmented language models. *Advances in Neural Information Processing Systems* 36 (2024).
- [149] Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. 2024. Buffer of thoughts: Thought-augmented reasoning with large language models. *Advances in Neural Information Processing Systems* 37 (2024), 113519–113544.
- [150] Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E. Gonzalez, and Bin Cui. 2024. Buffer of Thoughts: Thought-Augmented Reasoning with Large Language Models. *CoRR* abs/2406.04271 (2024). <https://doi.org/10.48550/ARXIV.2406.04271> arXiv:2406.04271
- [151] Xiaocheng Yang, Bingsen Chen, and Yik-Cheung Tam. 2024. Arithmetic Reasoning with LLM: Prolog Generation & Permutation. *arXiv preprint arXiv:2405.17893* (2024).
- [152] Xiao-Wen Yang, Zhi Zhou, Haiming Wang, Aoxue Li, Wen-Da Wei, Hui Jin, Zhenguo Li, and Yu-Feng Li. 2025. CARTS: Advancing Neural Theorem Proving with Diversified Tactic Calibration and Bias-Resistant Tree Search. In *ICLR*. OpenReview.net.
- [153] Zhen Yang, Ming Ding, Qingsong Lv, Zhihuan Jiang, Zehai He, Yuyi Guo, Jinfeng Bai, and Jie Tang. 2023. Gpt can solve mathematical problems without a calculator. *arXiv preprint arXiv:2309.03241* (2023).
- [154] Zezhou Yang, Cuiyun Gao, Zhaoqiang Guo, Zhenhao Li, Kui Liu, Xin Xia, and Yuming Zhou. 2024. A Survey on Modern Code Review: Progresses, Challenges and Opportunities. *arXiv preprint arXiv:2405.18216* (2024).
- [155] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Thirty-seventh Conference on Neural Information Processing Systems*.
- [156] Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. *arXiv preprint arXiv:2305.13172* (2023).

- [157] Tong Ye, Tengfei Ma, Lingfei Wu, Xuhong Zhang, Shouling Ji, and Wenhui Wang. 2024. Iterative or Innovative? A Problem-Oriented Perspective for Code Optimization. *arXiv preprint arXiv:2406.11935* (2024).
- [158] Xi Ye, Fangcong Yin, Yinghui He, Joie Zhang, Howard Yen, Tianyu Gao, Greg Durrett, and Danqi Chen. 2025. LongProc: Benchmarking Long-Context Language Models on Long Procedural Generation. *arXiv preprint arXiv:2501.05414* (2025).
- [159] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. STaR: Bootstrapping Reasoning With Reasoning. *arXiv:2203.14465 [cs.LG]*
- [160] Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Bo Wang, Shimin Li, Yunhua Zhou, Qipeng Guo, Xuanjing Huang, and Xipeng Qiu. 2024. Scaling of Search and Learning: A Roadmap to Reproduce o1 from Reinforcement Learning Perspective. *arXiv:2412.14135 [cs.AI]* <https://arxiv.org/abs/2412.14135>
- [161] Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Yunhua Zhou, and Xipeng Qiu. 2025. Revisiting the Test-Time Scaling of o1-like Models: Do they Truly Possess Test-Time Scaling Capabilities? *arXiv preprint arXiv:2502.12215* (2025).
- [162] Beichen Zhang, Kun Zhou, Xilin Wei, Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. 2023. Evaluating and Improving Tool-Augmented Computation-Intensive Math Reasoning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.). http://papers.nips.cc/paper_files/paper/2023/hash/4a47dd69242d5af908cdd5d51c971cbf-Abstract-Datasets_and_Benchmarks.html
- [163] Chunyan Zhang, Junchao Wang, Qinglei Zhou, Ting Xu, Ke Tang, Hairen Gui, and Fudong Liu. 2022. A Survey of Automatic Source Code Summarization. *Symmetry* 14, 3 (2022). <https://doi.org/10.3390/sym14030471>
- [164] Fengji Zhang, Bei Chen, Yue Zhang, Jacky Keung, Jin Liu, Daoguang Zan, Yi Mao, Jian-Guang Lou, and Weizhu Chen. 2023. Repocoder: Repository-level code completion through iterative retrieval and generation. *arXiv preprint arXiv:2303.12570* (2023).
- [165] Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. 2024. Aflow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762* (2024).
- [166] Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024. Generative Verifiers: Reward Modeling as Next-Token Prediction. *arXiv:2408.15240 [cs.LG]*
- [167] Ningyu Zhang, Xiang Chen, Xin Xie, Shumin Deng, Chuanqi Tan, Mosha Chen, Fei Huang, Luo Si, and Huajun Chen. 2021. Document-level Relation Extraction as Semantic Segmentation. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization.
- [168] Ningyu Zhang, Shumin Deng, Zhanlin Sun, Guanying Wang, Xi Chen, Wei Zhang, and Huajun Chen. 2019. Long-tail Relation Extraction via Knowledge Graph Embeddings and Graph Convolution Networks. 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)*. 3016–3025.
- [169] Ningyu Zhang, Qianghuai Jia, Shumin Deng, Xiang Chen, Hongbin Ye, Hui Chen, Huaixiao Tou, Gang Huang, Zhao Wang, Nengwei Hua, et al. 2021. Alicg: Fine-grained and evolvable conceptual graph construction for semantic search at alibaba. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 3895–3905.
- [170] Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. 2024. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286* (2024).
- [171] Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Zhihan Guo, Yufei Wang, Irwin King, Xue Liu, and Chen Ma. 2025. What, how, where, and how well? a survey on test-time scaling in large language models. *arXiv preprint arXiv:2503.24235* (2025).
- [172] Yuge Zhang, Qiyang Jiang, Xingyu Han, Nan Chen, Yuqing Yang, and Kan Ren. 2024. Benchmarking data science agents. *arXiv preprint arXiv:2402.17168* (2024).
- [173] Yuxiang Zhang, Shangxi Wu, Yuqi Yang, Jiangming Shu, Jinlin Xiao, Chao Kong, and Jitao Sang. 2024. o1-coder: an o1 replication for coding. *arXiv preprint arXiv:2412.00154* (2024).
- [174] Zihan Zhang, Black Sun, and Pengcheng An. 2025. Breaking Barriers or Building Dependency? Exploring Team-LLM Collaboration in AI-infused Classroom Debate. *arXiv preprint arXiv:2501.09165* (2025).
- [175] Zhuosheng Zhang, Yao Yao, Aston Zhang, Xiangru Tang, Xinbei Ma, Zhiwei He, Yiming Wang, Mark Gerstein, Rui Wang, Gongshen Liu, et al. 2025. Igniting language intelligence: The hitchhiker’s guide from chain-of-thought reasoning to language agents. *Comput. Surveys* 57, 8 (2025), 1–39.
- [176] Ziyin Zhang, Hang Yu, Shijie Li, Peng Di, Jianguo Li, and Rui Wang. 2024. GALLa: Graph Aligned Large Language Models for Improved Source Code Understanding. *arXiv preprint arXiv:2409.04183* (2024).
- [177] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. ExpeL: LLM Agents Are Experiential Learners. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference*

- on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (Eds.). AAAI Press, 19632–19642. <https://doi.org/10.1609/AAAI.V38I17.29936>
- [178] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* 1, 2 (2023).
 - [179] Yilun Zhao, Yitao Long, Hongjun Liu, Ryo Kamoi, Linyong Nan, Lyuhao Chen, Yixin Liu, Xiangru Tang, Rui Zhang, and Arman Cohan. 2024. DocMath-eval: Evaluating math reasoning capabilities of LLMs in understanding long and specialized documents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 16103–16120.
 - [180] Li Zhong, Zilong Wang, and Jingbo Shang. 2024. Ldb: A large language model debugger via verifying runtime execution step-by-step. *arXiv preprint arXiv:2402.16906* (2024).
 - [181] Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. 2023. A comprehensive survey on automatic knowledge graph construction. *Comput. Surveys* 56, 4 (2023), 1–62.
 - [182] Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Ningyu Zhang, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, and Huajun Chen. 2024. Knowagent: Knowledge-augmented planning for llm-based agents. *arXiv preprint arXiv:2403.03101* (2024).