Igniting Language Intelligence: The Hitchhiker's Guide from Chain-of-Thought Reasoning to Language Agents

ZHUOSHENG ZHANG, Shanghai Jiao Tong University, Shanghai, China
YAO YAO, Shanghai Jiao Tong University, Shanghai, China
ASTON ZHANG, Amazon Web Services, Seattle, United States
XIANGRU TANG, Yale University, New Haven, United States
XINBEI MA, Shanghai Jiao Tong University, Shanghai, China
ZHIWEI HE, Shanghai Jiao Tong University, Shanghai, China
YIMING WANG, Shanghai Jiao Tong University, Shanghai, China
MARK GERSTEIN, Yale University, New Haven, United States
RUI WANG, Shanghai Jiao Tong University, Shanghai, China
GONGSHEN LIU, Shanghai Jiao Tong University, Shanghai, China
HAI ZHAO, Shanghai Jiao Tong University, Shanghai, China

Large language models (LLMs) have dramatically enhanced the field of language intelligence, as demonstrably evidenced by their formidable empirical performance across a spectrum of complex reasoning tasks. Additionally, theoretical proofs have illuminated their emergent reasoning capabilities, providing a compelling showcase of their advanced cognitive abilities in linguistic contexts. Critical to their remarkable efficacy in handling complex reasoning tasks, LLMs leverage the intriguing chain-of-thought (CoT) reasoning techniques, obliging them to formulate intermediate steps en route to deriving an answer. The CoT reasoning approach has not only exhibited proficiency in amplifying reasoning performance but also in enhancing interpretability, controllability, and flexibility. In light of these merits, recent research endeavors have extended CoT reasoning methodologies to nurture the development of autonomous language agents, which adeptly adhere to language instructions and execute actions within varied environments. This survey article orchestrates a thorough discourse, penetrating vital research dimensions, encompassing (i) the foundational mechanics of CoT techniques, with a focus on elucidating the circumstances and justification behind its efficacy; (ii) the paradigm shift in CoT; and (iii) the burgeoning of language agents fortified by CoT approaches.

Z. Zhang and Y. Yao contributed equally to this work.
This work was partially supported by National Natural Science Foundation of China (62406188), Joint Funds of the National Natural Science Foundation of China (U21B2020), and Natural Science Foundation of Shanghai (24ZR1440300).
Authors' Contact Information: Zhuosheng Zhang (Corresponding author), Shanghai Jiao Tong University, Shanghai, Shanghai, China; e-mail: zhangzs@sjtu.edu.cn; Yao Yao, Shanghai Jiao Tong University, Shanghai, Shanghai, China; e-mail: yaoyao27@sjtu.edu.cn; Aston Zhang, Amazon Web Services, Seattle, Washington, United States; e-mail: az@astonzhang.com; Xiangru Tang, Yale University, New Haven, Connecticut, United States; e-mail: xiangru.tang@yale.edu; Xinbei Ma, Shanghai Jiao Tong University, Shanghai, Shanghai, China; e-mail: sjtumaxb@sjtu.edu.cn; Zhiwei He, Shanghai Jiao Tong University, Shanghai, Shanghai, China; e-mail: zwhe.cs@sjtu.edu.cn; Yiming Wang, Shanghai Jiao Tong University, Shanghai, Shanghai, China; e-mail: alsaceym@gmail.com; Mark Gerstein, Yale University, New Haven, Connecticut, United States; e-mail: mark.gerstein@yale.edu; Rui Wang, Shanghai Jiao Tong University, Shanghai, Shanghai, China; e-mail: wangrui12@sjtu.edu.cn; Gongshen Liu, Shanghai Jiao Tong University, Shanghai, Shanghai, China; e-mail: lgshen@sjtu.edu.cn; Hai Zhao, Shanghai Jiao Tong University, Shanghai, Shanghai, China; e-mail: zhaohai@cs.sjtu.edu.cn.

ACM 0360-0300/2025/03-ART206
https://doi.org/10.1145/3719341

Prospective research avenues envelop explorations into generalization, efficiency, customization, scaling, and safety. A repository for the related papers is available at https://github.com/Zoeyyao27/CoT-Igniting-Agent.

CCS Concepts: • **Computing methodologies → Natural language processing;**

Additional Key Words and Phrases: Large language models, chain-of-thought reasoning, autonomous agents

ACM Reference Format:
Zhuosheng Zhang, Yao Yao, Aston Zhang, Xiangru Tang, Xinbei Ma, Zhiwei He, Yiming Wang, Mark Gerstein, Rui Wang, Gongshen Liu, and Hai Zhao. 2025. Igniting Language Intelligence: The Hitchhiker's Guide from Chain-of-Thought Reasoning to Language Agents. *ACM Comput. Surv.* 57, 8, Article 206 (March 2025), 39 pages. https://doi.org/10.1145/3719341

1 Introduction

Language intelligence pertains to the aptitude for comprehending and reasoning through concepts articulated in natural languages [77, 110, 118, 132]. Spurred by advancements in scale, **large language models (LLMs)** have achieved remarkable progress in pursuing human-level language intelligence, compellingly evidenced by the strong empirical benchmarking performance in complex reasoning tasks [153], as well as theoretical proofs for the emergent reasoning abilities [10, 101, 148]. Reasoning, a pivotal research topic within the realm of language intelligence, is characterized as a multi-step process wherein inferences are drawn from discrete pieces of evidence, culminating in the formation of more abstract concepts that are instrumental in facilitating high-level predictions [44, 89, 179]. Recent research revealed that remarkable enhancements in performance could be attained by prompting LLMs to engage in a step-by-step reasoning process, as opposed to generating answers in a direct manner [94, 154]. The way to prompt LLMs to generate a series of intermediate reasoning steps for solving a problem is called **chain-of-thought (CoT)** prompting [154].

Optimization of CoT prompting techniques has garnered escalating interest, catalyzing notable paradigm shifts within the CoT framework. These shifts encompass three key aspects: (i) *prompting pattern*: from manual design of **in-context learning (ICL)** demonstrations to automatic prompt construction [170, 189, 199]; (ii) *reasoning format*: from unstructured natural language formats to structured ones [18, 173, 177, 201]; and (iii) *application scenario*: from singular language settings to multilingual environments [125], from a language modality to embracing multimodal approaches [188], and from complex reasoning tasks to general-purpose tasks [38, 59, 149, 151].

CoT reasoning is a representative emergent ability of LLMs [153]. Existing studies [137, 154] indicate that scaling up language models substantially improves their reasoning capabilities. Specifically, successful CoT reasoning consistently emerges in models exceeding 10 billion parameters. CoT provides a proficient strategy for deconstructing intricate issues into smaller, manageable sub-problems, systematically enabling solutions through a step-by-step approach. Leveraging the reasoning capabilities developed during pre-training [142, 163], CoT prompting adeptly identifies atomic knowledge components essential for reasoning processes and seamlessly integrates their relationships, thereby constructing intermediate, coherent reasoning steps [101, 148]. While the effectiveness of CoT reasoning is closely linked to model size, recent research highlights the significance of inference scaling laws in optimizing reasoning performance via dynamically allocating computational resources, whether through iterative refinement [96, 144, 183], search-based verification [129], collaborative verification [64], or inference-time voting [17]. By expanding CoT into a comprehensive framework for perception, memory, and reasoning, LLM-powered language agents have been developed to follow instructions and execute actions in real-world or simulated environments [111, 187] (Figure 1). These language agents come in two flavors: (i) autonomous agents [1, 42, 90, 113] and (ii) communicative agents [42, 98, 143, 200].

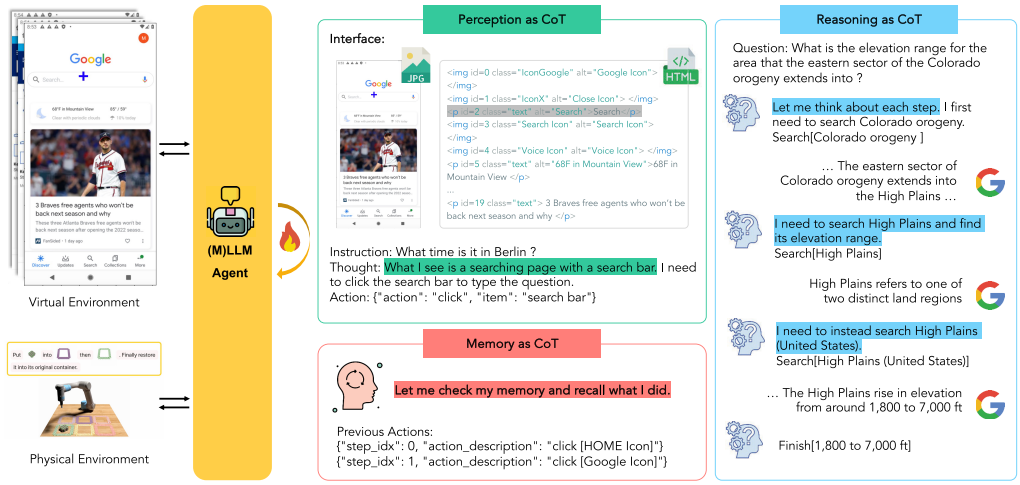

Fig. 1. An overview of language agent framework empowered with the CoT mechanism in perception, memory, and reasoning.

In this article, we navigate through research topics that encompass (i) unraveling the underlying mechanisms of CoT techniques, with a particular focus on discerning when and why CoT is effective; (ii) identifying and analyzing the paradigm shift occurring within CoT; and (iii) investigating the advent of language agents enabled by CoT techniques. The rest of this article is structured for a coherent and sequential exploration. Initially, we immerse ourselves in the fundamental aspects of CoT reasoning, which encompasses its defining features and the merits arising from employing CoT techniques. Subsequently, we delve deeper into the inherent mechanisms of CoT, striving to elucidate the specific conditions and reasons that determine its functionality. In the ensuing section, we classify paradigm shifts, directing our attention toward various prompting techniques, reasoning formats, and application scenarios. Following that, we explore the emerging landscape of language agents, spotlighting those facilitated by CoT techniques. To conclude, we engage in a discussion about the challenges encountered and future opportunities looming on the horizon.

Various related papers have selectively concentrated on distinct facets of LLMs [176], CoT reasoning [20, 75, 104, 180], and autonomous agents [145, 162], each providing overviews and taxonomies tailored to their respective domains. In contrast, our article transcends a mere summarization and aspires to furnish a thorough exploration of the fundamental mechanisms that underscore CoT reasoning, alongside a deep dive into the paradigm shifts enveloping this domain. Moreover, our article chronicles the trajectory from CoT reasoning in LLMs to the most recent advancements in autonomous language agents, with a dedicated aim to illuminate the intricate interconnections weaving through these crucial areas of study. This article caters to a wide audience, including beginners seeking comprehensive knowledge of CoT reasoning and language agents, as well as experienced researchers interested in foundational mechanics and cutting-edge discussions on these topics.

Key Takeaways. To the best of our knowledge, this constitutes the inaugural work to systematically explore the foundational mechanics of CoT techniques, the paradigm shift in CoT, and the complex interplay between CoT and agents. The key takeaways are as follows:

— CoT demonstrates efficacy under two overarching conditions: first, when an LLM with preferably at least 10 billion parameters is employed, and second, when the parametric

knowledge within the LLM encompasses knowledge pieces that are (i) pertinent to the task at hand and (ii) maintain strong mutual interconnections (Section 3.1).
— CoT functions by assisting in the identification of atomic knowledge pieces pivotal for reasoning and seamlessly connecting these components via the formation of intermediate reasoning steps (Section 3.2).
— CoT techniques have experienced substantial paradigm shifts, embracing alterations in prompting patterns, reasoning formats, and application scenarios (Section 4).
— CoT has acted as a catalyst in the evolution of LLM-empowered agents capable of understanding language instructions and executing actions in both real-world and simulated environments, specifically augmenting agent capabilities inperception, memory, and reasoning (Section 5).
— Despite the swift advancement of LLMs, CoT reasoning, and language agents, numerous challenges persist, such as generalization to unseen domains, efficiency amidst redundant interactions, customization of language agents, scaling up of language agents, and ensuring the safety of language agents (Section 6).

2 Preliminaries of CoT

In this section, we first compare CoT reasoning with the traditional approach of direct reasoning. Subsequently, we proffer definitions for the key components within CoT. Finally, we delineate the advantages of adopting CoT.

2.1 Definition

The concept of *chain-of-thought* refers to a series of intermediate reasoning steps that are generated to solve a problem or arrive at an answer [154], in the form of <input→reasoning chain (rationale)→output> mappings. This approach is often more effective than traditional direct reasoning, which attempts to tackle the entire problem all at once. For example, standard classification, multiple choice, and question answering problems often leverage direct reasoning in the form of <input→output> mappings.

To elucidate CoT, we establish standard definitions for its key components as illustrated in Figure 2. Formally, assuming that the reasoning dataset distribution is $\mathcal{D}$, we denote $s = (x, y) \sim \mathcal{D}$ as a sampling on $\mathcal{D}$, where x and y denote the question (input) and the answer (output), respectively, and they are both in the form of text sequences. We use $|x|$ to denote the length of the sequence x, and p_θ to denote the pre-trained language model parameterized with θ. Details of definitions are presented next.

Instruction. Instructions are usually short sentences used to prompt an LLM to generate answers in the desired format. They guide the LLM to think step by step in the reasoning process. We notate the instruction as p, which is set to different text sequences depending on the task requirements.

Rationale. We uniformly refer to the intermediate processes of CoT reasoning as "rationales." Rationales can encompass solutions, intermediate reasoning steps, or any relevant external knowledge pertaining to a question. We define rationale as r. If r is generated by the LLM, instruction p can be used to obtain $r \sim p_\theta(x, \mathrm{p})$. If r is written by a human, instruction p can be exempted, and $r = f(x)$, where $f(\cdot)$ indicates the handwriting operation.

Exemplars. Exemplars are typically presented as desired input-output pairs in few-shot prompting approaches, each of which contains questions, a rationale, and an answer. Exemplars serve as in-context demonstrations of input-output relationships before generating predictions for test-time examples. Exemplars are usually concatenated before input questions. Specifically, assuming the

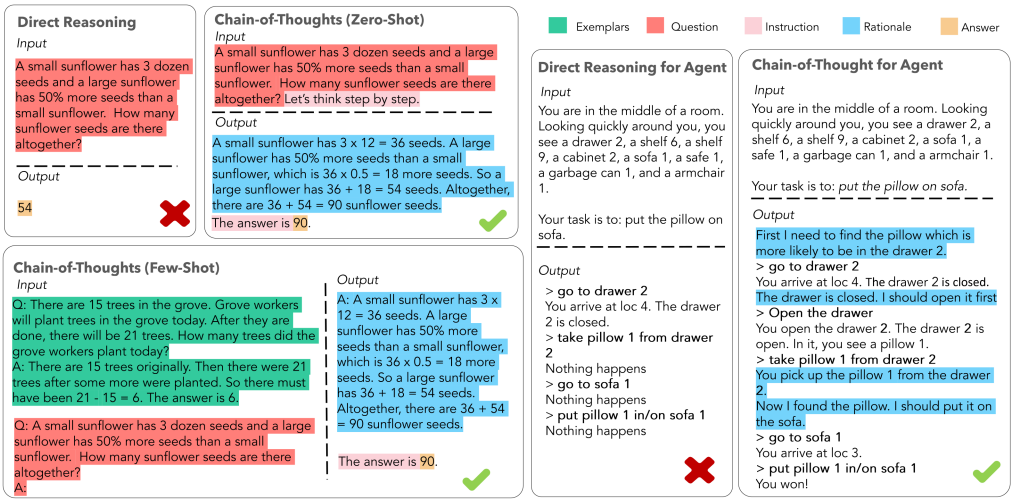

Fig. 2. Comparison between CoT reasoning and direct reasoning. CoT refers to a series of intermediate reasoning steps that are generated to solve a problem or arrive at an answer [154]. This approach is often more effective than direct reasoning, which attempts to tackle the entire problem all at once.

exemplar size of n, exemplars E can be formulated as

$$E = [(x_1, r_1, y_1) \circ \cdots \circ (x_{n-1}, r_{n-1}, y_{n-1}) \circ (x_n, r_n, y_n)], \tag{1}$$

where $\circ$ represents concatenation, $(x_i, y_i) \sim \mathcal{D}$ and $r_i = f(x_i)$.

Zero-Shot-CoT. Zero-Shot-CoT does not require users to provide exemplars. Instead, it typically relies on instructions to facilitate the LLM in conducting step-by-step reasoning, thereby generating answers. For example, Kojima et al. [54] first elicited the LLM to generate the rationale r using the instruction p_1 such as "Let's think step by step," and then use the instruction p_2 such as "The answer is" to obtain the final answer following the question and rationale. Formally, the output y can be computed as follows:

$$r \sim \prod_{i=1}^{|r|} p_\theta(r_i | x, \mathrm{p}_1, r_{<i}), \quad y \sim \prod_{i=1}^{|y|} p_\theta(y_i | x, \mathrm{p}_1, r, \mathrm{p}_2, y_{<i}). \tag{2}$$

Few-Shot-CoT. Few-Shot-CoT involves providing a set of exemplars with associated rationales. These exemplars are concatenated with the question to prompt the LLM to generate the rationale and answer. In this setting, the output y is obtained in an end-to-end mode, which can be formulated as

$$y \sim \prod_{i=1}^{|y|} p_\theta(y_i | E, x, y_{<i}). \tag{3}$$

2.2 Benefits of CoT

CoT techniques have shown various kinds of benefits, including improved reasoning performance, interpretability, controllability, and flexibility. We summarize them in detail next.

Improved Reasoning Performance. CoT facilitates a step-by-step progression in the reasoning process for LLMs. By breaking down complex, multi-step problems into intermediate stages, CoT minimizes the risk of overlooking crucial details. Moreover, it ensures the efficient allocation of additional computational resources to problems demanding a higher degree of reasoning steps.

Numerous studies have conclusively demonstrated the efficacy of CoT across a wide range of domains, encompassing arithmetic reasoning, commonsense reasoning, and symbolic reasoning [54, 150, 154].

Improved Interpretability. CoT offers an interpretable glimpse into the decision-making process of LLMs. Breaking down complex reasoning tasks into a chain of interconnected thoughts makes it easier to understand the underlying logic and reasoning behind a decision or conclusion made by LLM. It sheds light on how the model may have reached a specific answer, offering valuable insights for debugging and pinpointing where the reasoning process may have deviated from the correct path. However, it is important to note that fully characterizing the model's computations supporting an answer still presents an open challenge [154].

Improved Controllability. By prompting LLMs to output a chain of interconnected thoughts, users can exert greater influence over the cognitive processes of LLM. Many studies [69, 173] were dedicated to the identification and rectification of specific thought units where the reasoning path may have gone off track or where additional information is required. This increased controllability allows for more deliberate and accurate answers.

Improved Flexibility. The utilization of CoT reasoning can be easily prompted in adequately large, off-the-shelf LLMs by simply adding instruction at the end of the input question for Zero-Shot-CoT or incorporating CoT exemplars used for Few-Shot-CoT [154]. The flexibility of CoT extends beyond the realm of reasoning tasks, making it applicable to various fields, including classic **natural language processing (NLP)**, scientific applications, and agent-based systems.

3 Underlying Mechanism of CoT

This section explores the foundational mechanisms of CoT, encompassing the general conditions that determine when and why CoT is effective.

3.1 When CoT Works

Although CoT has shown promising benefits, it may not be suitable in any conditions [54, 154, 188]. We will introduce when CoT works in engineering and theoretical perspectives. Then we summarize the general conditions to suggest the effective application scopes of CoT reasoning.

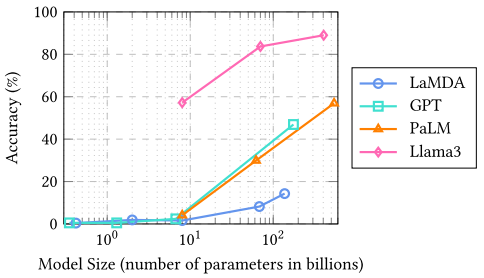

Fig. 3. Impact of model size on accuracy with CoT prompting on the GSM8K dataset. The *x*-axis uses a logarithmic scale. Four model families (LaMDA [139], GPT [12], PaLM [4], and Llama 3 [27]) across varying model sizes are involved.

— From an engineering perspective, Wei et al. [154] posited that CoT reasoning is helpful under three conditions: (i) an LLM with more than 10B parameters is used, (ii) the task is challenging and requires multi-step reasoning, and (iii) the performance of direct prompting does not increase dramatically while scaling the model size.[1] Figure 3 shows the performance of different model sizes within the same family on the GSM8K [22] dataset,[2] comparing accuracy between using CoT prompts and direct prompts. From the illustration, we can observe that once the model size

[1]It should be noted that recent studies have explored fine-tuning smaller language models (with less than 10 billion parameters) to perform CoT reasoning for specific tasks [81, 182]. Here we only discuss general scenarios where LLMs can achieve effective CoT reasoning per se—better performance than direct reasoning—without additional task-specific fine-tuning on CoT-style training data.

[2]Results are taken from Wei et al. [154]

exceeds 10 billion parameters, CoT prompting becomes significantly more effective in enhancing the model's reasoning capabilities. Otherwise, CoT techniques tend to struggle with smaller LLMs [153]. It may lead to hallucination because of lacking supportive knowledge in LLMs [188] and inferior reasoning capabilities [81]. CoT reasoning is also less effective in simple-step tasks such as matching, sequence labeling [106], and single-choice question [15].

— From a theoretical perspective, Prystawski et al. [101] proved that CoT reasoning is helpful when training data (possibly considered as the parametric knowledge in an LLM) consists of local clusters of variables that strongly influence each other. This finding implied that the LLM must have the knowledge related to the task to support CoT reasoning. We call such knowledge *atomic knowledge*.

As CoT reasoning is often elicited by ICL, such as Zero-Shot-CoT and Few-Shot-CoT, another line of study tries to understand when CoT works from the perspective of ICL. Zhang et al. [189] showed that CoT reasoning works effectively when prompted with diverse exemplars. Wang et al. [142] found that rationales being relevant to the query and correctly ordering the reasoning steps are the keys to the effectiveness of CoT prompting.

Besides prompting, introducing reasoning materials and necessary knowledge for LLMs in the training corpus has also exhibited a profound improvement in CoT reasoning ability in LLMs [180]. Recent studies found that pre-training with code data [21] or fine-tuning (e.g., instruction tuning) with CoT-style data [182] is beneficial for effective CoT reasoning. In other words, the CoT reasoning in the same LLMs can be improved or the CoT reasoning ability can be induced in smaller models.

Based on the preceding discussion, CoT demonstrates efficacy under two overarching conditions: first, when an LLM with preferably at least 20 billion parameters is employed, and second, when the parametric knowledge within the LLM encompasses knowledge pieces that (i) are pertinent to the task at hand and (ii) maintain strong mutual interconnections.

3.2 Why CoT Works

Recent studies have employed both empirical and theoretical approaches in an effort to comprehend the underlying reasons for the effectiveness of CoT.

— Empirically, Wei et al. [154] believed that the success of CoT reasoning constitutes a multifaceted phenomenon that likely involves various emergent abilities. Those abilities include semantic understanding, symbol mapping, topic coherence, arithmetic ability, and faithfulness. Interestingly, Zhang et al. [189] found that mistakes in exemplar rationales do not lead to significant performance drops. Wang et al. [142] reported a similar observation that LLMs can generate coherent reasoning steps and achieve over 80% to 90% of the performance, although prompted with invalid reasoning steps in the exemplars. Those findings imply that LLMs already have an innate ability to reason after pre-training [148, 189]. CoT prompting specifies an output format that regularizes the model generation to generate step by step while being in order and relevant to the query [142]. In other words, CoT techniques help *compel* the model to conduct reasoning rather than teaching it *how* to accomplish reasoning [189].

— Theoretically, Bayesian inference is a popular way to investigate why CoT works from a theoretical perspective [101, 148]. Prystawski et al. [101] proved that CoT is effective when the training data exhibits a localized structure with respect to dependencies between variables. In the context of LLMs, the proof can be interpreted that the parametric knowledge within the LLM comprises knowledge pieces that are related to the target problem, and those knowledge pieces exert strong mutual connections with each other. To verify the proof, Bi

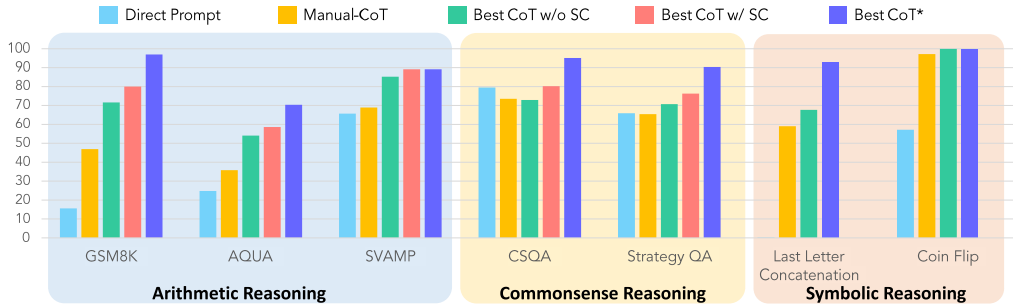

Fig. 4. Performance on seven reasoning tasks. "Direct Prompt" refers to the standard few-shot prompting approach, where exemplars are formatted as questions and answers, with the model providing direct answers. "Best CoT w/o SC," "Best CoT w/ SC," and "Best CoT*" represent the highest accuracy (%) achieved as of October 2023 ("SC" stands for self-consistency [150]). While the first two use the text-davinci-002 as the LLM engine, the latter allows for the model employed to vary for each task (details in Table 1). For a fair comparison, the performances of "Direct Prompt," "Manual-CoT," "Best CoT w/o SC," and "Best CoT w/ SC" are all based on using the text-davinci-002 as the LLM engine.

et al. [10] conducted an empirical study on code data and found that the local structural properties of the data are crucial for improving CoT reasoning abilities. These findings by Prystawski et al. [101] and Bi et al. [10] compellingly indicated that CoT may help identify the atomic pieces of knowledge used for reasoning and bridge the relationship between the atomic pieces of knowledge with intermediate reasoning steps. Similarly, Wang and Wang [148] used knowledge graphs for analysis and found that organizing the known facts as "chains"—that is, CoT, can significantly impact the effectiveness of reasoning. By doing so, LLMs are able to accurately deduce previously unseen facts from known ones to answer a given query without explicitly encoding reasoning rules.

4 Paradigm Shifts of CoT

After elucidating the general conditions determining when and why CoT is effective, we seek to achieve a more profound and intuitive understanding of the improvements in CoT's reasoning capabilities for LLMs. To this end, we compile and summarize the best performances of CoT across seven of the most emblematic reasoning tasks as of October 2023. We compare these performances with those achieved without CoT and present our findings in Figure 4. These seven reasoning tasks span across distinct categories, including (i) *arithmetic reasoning*: GSM8K [22], AQuA [68], and SVAMP [99]; (ii) *commonsense reasoning*: CSQA [136] and StrategyQA [30]; and (iii) *symbolic reasoning*: Last Letter Concatenation [154] and Coin Flip [154].

Figure 4 illustrates that the benchmark performance in complex reasoning tasks has advanced rapidly, with CoT exerting a significant influence on the reasoning abilities of LLMs across all seven tasks. Notably, apart from commonsense reasoning, the relatively straightforward CoT format Manual-CoT proposed by Wei et al. [154] substantially improves overall accuracy compared to the direct prompt in both arithmetic and symbolic reasoning. The deficiency in CoT's performance regarding commonsense reasoning tasks has been observed both in Manual-CoT [154] and Zero-Shot-CoT [54]. However, when CoT is integrated with a significantly larger PaLM (540B) model, it consistently enhances commonsense reasoning. Notably, Zero-Shot-CoT also notes that the rationales generated through CoT often exhibit logical correctness or only contain human-understandable errors. This suggests that CoT encourages improved commonsense reasoning, even when the task metrics do not explicitly measure it. While Manual-CoT fails to yield

Table 1. Best CoT* on Seven Reasoning Tasks (SC: Self-consistency by Wang et al. [150])

Category	Dataset	Model	Best Acc	LLM
Arithmetic Reasoning	GSM8K	CSV [194]	97.00	GPT-4 Code Interpreter
	AQuA	Natural Program [69]	70.34	ChatGPT
	SVAMP	PoT [18] + SC	89.10	Text-davinci-002
Commonsense Reasoning	CSQA	Manual-CoT [154] + SC	95.10	PaLM 2
	StrategyQA	Manual-CoT [154] + SC	90.40	PaLM 2
Symbolic Reasoning	Last Letter Concatenation	Natural Program [69]	92.98	ChatGPT
	Coin Flip	Auto-CoT [189]	99.90	Text-davinci-002

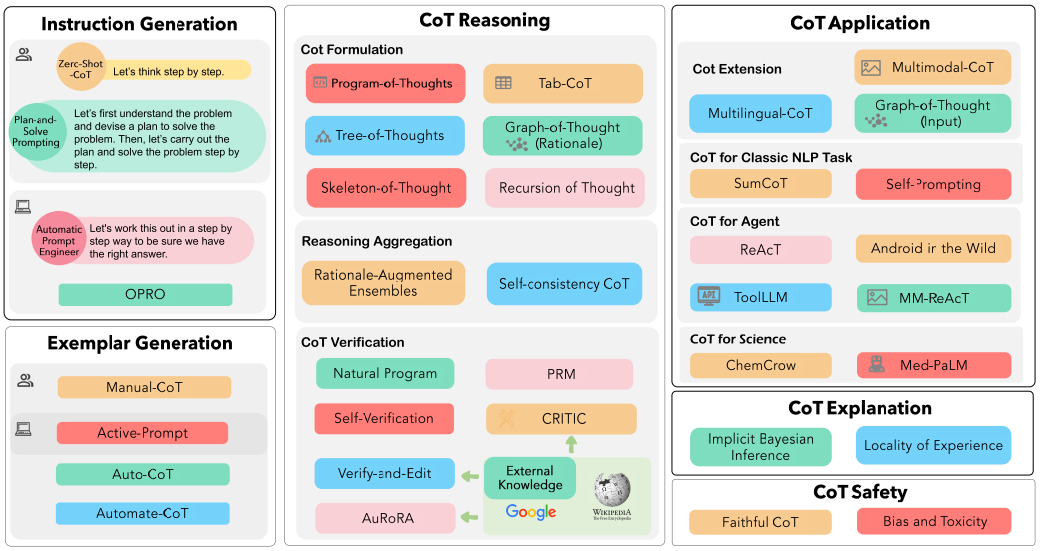

Fig. 5. Overview of representative CoT approaches. We delve into the paradigm shifts of CoT techniques in three key directions: (i) *prompting pattern* (instruction generation and exemplar generation), (ii) *reasoning format* (CoT formulation, reasoning aggregation, and CoT verification), and (iii) *application scenario* (multilingualism, multimodality, and general-purpose tasks).

performance gains in commonsense reasoning, the optimization of reasoning techniques, such as self-consistency answer aggregation [150] and automatic exemplar construction [127], reveals the potential of CoT to achieve remarkable results generally. Moreover, for ease of reference and to provide a clear overview of how CoT can achieve top performance on the seven different datasets, we have included the latest models that have achieved the best performance and the specific LLM engine they utilized in Table 1.

In conclusion, we see that compared with the vanilla prompting approach in the work of Wei et al. [154], the latest CoT reasoning techniques have been strengthened throughout the full stack of the reasoning process, such as multimodal perception [46, 114, 177, 188], automatic prompting [24, 189], reasoning verification [65, 69, 156], and consistency-based sampling [149, 150]. With the growing interest in CoT, researchers are continually striving to harness its full potential for enhancing LLMs reasoning capabilities. In this section, we will embark on a journey through the realm of CoT research, following the map of CoT overview as illustrated in Figure 5, delving into the comprehensive discussions of advancements made in three key directions: (i) *prompting pattern*, (ii) *reasoning format*, and (iii) *application scenario*.

4.1 Prompting Pattern

The prompting pattern can be primarily divided into two components: *instruction generation* and *exemplar generation*. Instruction generation primarily focuses on finding the optimal instructions to prompt LLM, enabling them to engage in step-by-step reasoning instead of directly answering the question. This approach mainly aims to maximize LLM's zero-shot capability. Exemplar generation primarily focuses on finding the best set of input-output demonstration exemplar pairs for Few-Shot-CoT. These exemplars are used to prompt LLMs along with a test input, enabling the model to predict the corresponding output.

4.1.1 Instruction Generation. Instruction generation can be categorized into two distinct methods: manual instruction generation and automatic instruction generation, based on their respective generation processes.

Early efforts primarily involved manual construction of instruction prompts. The earliest and most traditional instruction generation method was Zero-Shot-CoT proposed by Kojima et al. [54]. Zero-Shot-CoT demonstrates that LLMs can perform zero-shot reasoning by adding a simple prompt, "Let's think step by step," before each answer. Zero-Shot-CoT outperforms zero-shot LLM performances on various reasoning tasks without the need for hand-crafted few-shot examples, marking the inception of a new era in Zero-Shot-CoT.

Wang et al. [146] further proposed Plan-and-Solve prompting to address the missing-step errors in Zero-Shot-CoT reasoning. It consists of devising a plan to divide the task into smaller sub-tasks and carrying out the sub-tasks according to the plan. Plan-and-Solve prompting consists of two stages. In the first stage, the author prompts the LLM using the proposed prompting template "Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out the plan and solve the problem step by step" to generate the reasoning process and the answer. The second stage extracts the answer using an answer prompt (e.g., "Therefore, the answer (arabic numerals) is").

However, manually designing instructions may not always yield the desired results, and users often need to experiment with various prompts to achieve the desired behavior. In response to this challenge, Zhou et al. [199] proposed **Automatic Prompt Engineer (APE)**, a method designed for the automated generation and selection of instructions for LLMs. APE treats instruction generation as a form of natural language program synthesis and optimizes this process by searching through a pool of instruction candidates proposed by an LLM. The primary goal is to maximize a chosen score function. To elaborate further, APE initiates the process by instructing the LLM to generate a set of candidate instructions using manually crafted templates. Subsequently, it utilizes the LLM to infer the most likely instructions with the highest score, based on input-output exemplars. By harnessing the capabilities of LLMs, APE streamlines the prompt engineering process, alleviating extensive human intervention and generating high-quality instructions.

Yang et al. [170] presented **Optimization by PROmpting (OPRO)**, a straightforward yet highly effective approach that harnesses the power of a language model (LLM) as optimizers. OPRO represents a groundbreaking method in optimization, utilizing LLMs to their full potential. OPRO initiates the optimization process by presenting a natural language description of both the optimization problem and the optimization trajectory. This trajectory includes prior solutions along with their associated optimization scores. Subsequently, updated solutions are devised and evaluated for their performance and quality. The prompt for the subsequent optimization step incorporates these solutions after thorough examination. As the iterative process unfolds, the solutions undergo progressive refinement, ultimately improving their quality.

Initially, OPRO is applied to address two classic optimization challenges: the linear regression problem and the traveling salesman problem. The study then proceeds to demonstrate that prompts

optimized by OPRO surpass human-designed prompts in performance, particularly on tasks such as GSM8K and Big-Bench Hard tasks. OPRO showcases its efficiency in resolving common optimization challenges and enhancing prompts by presenting optimization tasks in natural language for LLMs, consistently generating and refining solutions.

4.1.2 Exemplar Generation. Similar to instruction generation, exemplar generation can also be classified into two categories based on the method of constructing exemplars: manual exemplar generation and automatic exemplar generation.

Few-Shot-CoT reasoning, formally explored by Wei et al. [154], represents a discrete prompt learning approach that uses multiple input-output pairs to prompt the LLM to output rationales and obtain the final answer. To provide a clearer distinction, we will refer to their work as Manual-CoT. Manual-CoT follows the traditional manual exemplar generation method. In contrast to the conventional ICL, where LLMs are prompted with a list of input-output demonstration pairs alongside a test input to enable the model to predict the output, Manual-CoT involves prompting the model's outputs with manually designed additional logical reasoning procedures in addition to the target output.

Diao et al. [24] took Manual-CoT a step further by optimizing the selection of exemplars and introduced Active-Prompt, which uses task-specific example prompts annotated with human-designed rationales. Active-Prompt exists in a state that falls between manual exemplar generation and automatic exemplar generation. The method selects the most uncertain questions from a pool of task-specific queries using uncertainty-based active learning metrics. Active-Prompt first asks LLM to answer questions multiple times following the Manual-CoT [154]. The model then selects the most uncertain questions based on the uncertainty metric (e.g., disagreement, entropy, variance, self-confidence), manually annotates the rationales, and uses the questions and rationales as examples for inference.

To eliminate the need for manual efforts in hand-crafting task-specific demonstrations to generate reasoning chains one by one, Zhang et al. [189] proposed Auto-CoT, which maintains the diversity of sampled questions and generates reasoning chains to automatically construct demonstrations. Specifically, Auto-CoT consists of two main stages: (i) *problem clustering*: divide the given dataset of problems into several clusters, and (ii) *demonstration sampling*: select a representative problem from each cluster and use Zero-Shot-CoT to generate its reasoning chain.

Shum et al. [127] proposed a strategy called *Automate-CoT* (Automatic Prompt Augmentation and Selection with Chain-of-Thought) that automates the process of augmenting and selecting rational chains for CoT prompting. The process consists of three steps: augmenting the language model to generate multiple pseudo-chains, pruning the pseudo-chains based on consistency with ground-truth answers, and selecting the most helpful CoT using a variance-reduced policy gradient strategy.

4.2 Reasoning Format

The enhancements in reasoning format primarily encompass three aspects: *CoT formulation*, *reasoning aggregation*, and *CoT verification*. CoT formulation focuses on transforming the sequential CoT into various cognitive structures, such as tree-like, graph-like, or table-like formats, thereby incorporating structural thinking cues. Reasoning aggregation primarily concerns the enhancement of LLM CoT reasoning accuracy through the aggregation of results sampled from the LLM. CoT verification primarily emphasizes the introduction of verification methods to verify and amend the CoT reasoning process. We will elaborate on these three aspects in the following sections.

4.2.1 CoT Formulation. We present five representative CoT formulations in Figure 6. We will progressively delve into the CoT formulation shifts based on this illustration.

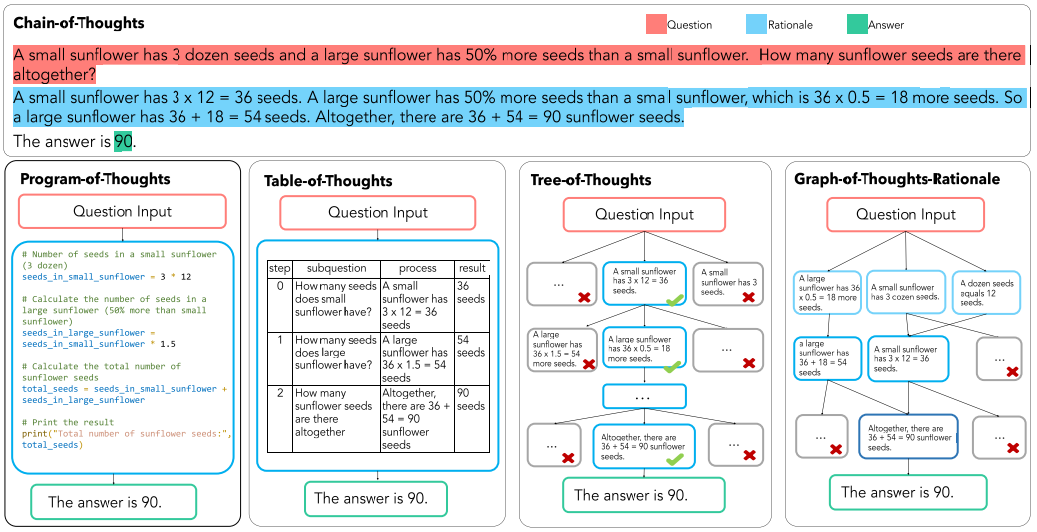

Fig. 6. Formulation shifts of CoT. We illustrate five representative CoT formulations in chronological order: (i) Chain-of-Thoughts (CoT), (ii) Program-of-Thoughts (PoT) [18], (iii) Tabular Chain-of-Thought (Tab-CoT) [201], (iv) Tree-of-Thoughts (ToT) [173], and (v) Graph-of-Thoughts-Rationale (GoT-Rationale) [9].

Chen et al. [18] introduced PoT (Program-of-Thoughts) for solving complex numerical reasoning tasks. PoT uses language models to generate both text and programming language statements, which can be executed on a program interpreter to decouple complex computation from reasoning and language understanding.

Ziqi and Lu [201] explored the structural reasoning ablilty for LLMs. They introduce Tab-CoT (Tabular Chain-of-Thought), which adopts a table-filling approach to model CoT. In Tab-CoT, an instruction of "| step | subquestion | process | result |" is manually designed to prompt LLMs to generate a table while conducting the reasoning process. The answer is then extracted from the generated table at the end of the process. Tab-CoT showcases robust zero-shot and few-shot capabilities in performing reasoning across multiple dimensions, encompassing both rows and columns.

Yao et al. [173] proposed ToT (Tree-of-Thoughts) that breaks CoT into thought units and formulates them into tree structure. ToT allows LLMs to explore coherent thought units that serve as intermediate steps toward problem solving, consider different options, and evaluate their decisions. By incorporating different methods, ToT is able to look ahead to determine what to do next or trace-back to correct history decisions. Experiments have demonstrated that ToT significantly elevates the problem-solving capabilities of language models. This improvement is particularly noteworthy in the context of tasks that demand intricate non-trivial planning or search processes.

Based on ToT, Besta et al. [9] further extended the tree structure into graph structure and propose GoT-Rationale (Graph-of-Thoughts-Rationale) (note that to distinguish from GoT proposed by Yao et al. [177], the Besta et al. [9] model is dubbed as GoT-Rationale). GoT-Rationale models the thought generation process of language models as a graph. The system architecture of GoT comprises multiple interacting modules: (i) Prompter that prepares a prompt for the LLM, which are then used to generate responses; (ii) Parser that extracts information from the LLM's responses, which is then used by other modules in the architecture; (iii) Scorer that verifies and scores the LLM's replies to determine their quality and relevance to the task at hand; and (iv) Controller that process with two elements: GoO (Graph of Operations) and GRS (Graph Reasoning State). GoO

is a static structure that specifies the graph decomposition of a given task, which means that it prescribes the transformations to be applied to LLM thoughts, along with their order and dependencies. GRS is a dynamic structure that maintains the state of the ongoing LLM reasoning process, which includes the history of its thoughts and their states. Experimental results indicate that GoT outperforms state-of-the-art techniques in tasks such as sorting, set operations, keyword counting, and document merging.

Lee and Kim [55] proposed RoT (Recursion of Thought), which empowers language models to recursively generate multiple contexts for problem solving. In RoT, LLMs are prompted to output special tokens such as GO, THINK, and STOP, which serve to initiate context-related operations. The THINK token indicates the model needs to solve a sub-problem, which triggers a recursive process to generate a new context for that sub-problem. This innovative approach enables the models to effectively handle problems whose solutions exceed the maximum context size by creating and managing multiple nested contexts.

Different from preceding works that focus on introducing structural information into CoT reasoning, Ning et al. [93] proposed SoT (Skeleton-of-Thought) to accelerate the CoT reasoning process. SoT consists of two stages: (i) *skeleton stage*: SoT guides the LLM to output a concise skeleton of the answer through a manually designed a skeleton prompt template and extracts points from the skeleton response, and (ii) *point-expanding stage*: SoT prompts the LLM to expand on each point in parallel through a point expanding template and finally concatenates all the points to get the final answer.

4.2.2 Reasoning Aggregation. Wang et al. [150] introduced a novel decoding strategy called *self-consistency* to replace the greedy decoding strategy in CoT. Self-consistency CoT first prompts the language model following the Manual-CoT [154] and then samples a diverse set of reasoning paths from the language model's decoder. Finally, Self-consistency CoT finds the most consistent answer by taking a majority vote which was found to significantly improve the performance of the CoT.

Wang et al. [149] further developed a unified framework for rationale-augmented ensembles which aims at aggregating over multiple rationales generated from the language model to mitigate the brittleness of the results. The authors explore three distinct approaches of rationale-augmented ensembles, each differing in how randomness is introduced into the input or output space: (i) *self-consistency* [150]: the ensembling is based on sampling multiple language model outputs; (ii) *prompt-order ensembling*: the ensembling process is based on the order of the input exemplars; and (iii) *input-rationale ensembling*: the ensembling is based on sampling multiple input exemplars rationales from LLMs. The authors found that regardless of the variation in input or prompt, the best way to improve task performance is sampling rationale in the output space.

4.2.3 CoT Verification. CoT verification initially focused on self-verification through multiple rounds of questioning, enabling models to validate their own responses. Later works involve leveraging external tools for information validation, such as information retrieval, calculators, or program execution. This section explores various methods and strategies within CoT verification, contributing to the enhancement of model reliability and response accuracy.

Weng et al. [156] first proposed and proved that LLMs have self-verification abilities by using the conclusion obtained through CoT as a condition for verifying the original problem. Self-verification consists of two steps: (i) forward reasoning that samples multiple candidate reasoning paths, and (ii) backward verification that calculates the verification scores for each candidate's answer by masking the original conditions and predicting their results in turn. The answer with the highest score is selected as the final answer.

Lightman et al. [65] focused on training reward models and conducted a comparison between the outcome supervision reward model (ORM) and process supervision reward model (PRM) for LLM

to solve problems from the MATH dataset [41], finding that process supervision significantly outperforms outcome supervision. Outcome supervision is provided without humans, as the MATH dataset has automatically checkable answers. Process supervision, however, requires human data labelers to label the correctness of each step in model-generated solutions. The authors also released PRM800K, a complete dataset of 800,000 step-level human feedback labels used to train their best reward model.

Ling et al. [69] proposed a verification process using a "Natural Program" format. Natural Program breaks down the reasoning process into individual steps, which is accompanied by its corresponding minimal set of premises. Then, the authors employed a two-phase sequence generation strategy, Unanimity-Plurality Voting, to verify the deductive reasoning process. Unanimity-Plurality Voting first performs deductive validations on sampled reasoning chains and then conducts a majority-based voting among the verified candidate chains to obtain the final answer.

Based on Self-consistency CoT [150], Zhao et al. [191] designed the Verify-and-Edit framework to improve the factuality and accuracy of reasoning chains generated by CoT. The framework first passes predictions with lower-than-average consistency to the next stages for further processing. The second step involves producing verifying questions using manually designed prompts to test the factual correctness of the predictions. The framework then retrieves external knowledge from reliable systems (e.g., Wikipedia, Google) and edits the generated rationales with the informed answers obtained from external knowledge. Finally, the framework produces new predictions based on the edited rationales.

Similarly, Gou et al. [31] introduced a framework called *CRITIC* that allows LLMs to validate and amend their own outputs through tool-interactive critiquing. CRITIC formulates various external tools into text-to-text functions (e.g., search engines, code interpreters) to integrate external tools into LLMs. Through a manually designed prompt template, the framework starts with an initial output and interacts with appropriate external tools to evaluate certain aspects of the text, revising the output based on the feedback obtained during the validation process.

Zou et al. [202] proposed AuRoRA, an augmented reasoning and refining system with task-adaptive CoT prompting. AuRoRA has the characteristics of task self-adaptation and process automation. It extracts relevant knowledge from multiple sources, reducing the issue of incorrect information. Knowledge from different sources (e.g., Wikipedia) is then combined, double-checked, and refined to enhance reliability. The system revises the initial CoT using high-quality extracted knowledge to enhance accuracy and logic.

Instead of using a single LLM to refine their outputs based on feedback on their previous outputs, multi-agent debate has been proposed to improve reasoning performance [26]. Liang et al. [63] identified a DoT (Degeneration-of-Thought) problem—the LLM fails to generate novel thoughts through reflection even if its initial stance is incorrect once the LLM has established confidence in its solutions. The DoT problem can be addressed by allowing divergent thinking using a MAD (Multi-Agent Debate) framework where multiple agents express their arguments and a judge manages the debate process to obtain a final solution. Du et al. [26] also leveraged multiple instances of an LLM to debate their individual reasoning processes over multiple rounds to arrive at a consistent final answer. The approach has been shown to improve the factual validity of generated content and reduce fallacious answers and hallucinations.

— ***Can LLMs Perform Reliable CoT Verification?*** Even though the preceding CoT verification approaches have been proposed as a remedy to improve reasoning performance and reliability, the role and efficacy of the verification are questioned. Recent work has tried to examine the self-verification capabilities of LLMs in reasoning tasks [45, 131, 141]. Huang et al. [45] identified that the enhancements observed in CoT verification studies were often facilitated by the utilization of oracles, which guided the self-correction process using

ground-truth labels, external tools, or feedback from the environment to evaluate the correctness of the responses. However, it is crucial to note that obtaining high-quality external feedback is challenging in real-world applications. In the absence of oracles, LLMs encounter difficulties in rectifying their initial responses solely relying on their inherent capabilities—which we regard as *imperfect verification*. In the imperfect verification scenario, LLMs tend to non-existent violations, and over-correct the reasoning process with false positives—walk right over the correct solution especially when there are mistakes in the verification process [141]. This phenomenon raises concerns about the inherent capability of the LLM to accurately assess the correctness of its reasoning process. It becomes evident that the key to achieving effective CoT verification lies in harnessing external, high-quality feedback for verification. For instance, integrating external tools such as search engines and calculators into the verification process has shown to be beneficial [18, 19, 95, 97].

4.3 Application Scenarios

Inspired by the latest techniques proposed previously to enhance the reasoning capabilities of LLMs, CoT techniques have shown greater impact with the shifts of its application scenarios. The application scenario shifts include the extension from single-language tasks to *multilingual tasks*, from single-language modality to *multimodalities*, and from complex reasoning tasks to *general-purpose tasks*.

4.3.1 From Single Language to Multilingual Scenarios. Shi et al. [125] extended the CoT to encompass the realm of multilingualism and introduced the MGSM (Multilingual Grade School Math) benchmark, which evaluates the reasoning abilities of LLMs in multilingual settings. This benchmark comprises 250 grade-school math problems that have been translated into 10 linguistically diverse languages. Furthermore, the authors proposed a concept called *Multi-lingual CoT*, which involves prompting LLMs with multilingual exemplars and incorporating English intermediate reasoning steps. This approach has been shown to yield competitive or even superior results. Multilingual CoT suggests that employing English CoT prompting as a baseline could be a valuable strategy for multilingual reasoning research.

Qin et al. [107] introduced a cross-lingual prompting technique aimed at extending zero-shot CoT reasoning to multiple languages. The technique comprises two key procedures. First, a cross-lingual alignment prompting is employed to synchronize representations across different languages, utilizing the prompt "Let's understand the task in English step-by-step!" Second, a task-specific solver prompting is implemented to guide the model in completing the final task, employing the prompt "Let's resolve the task you understand above step-by-step!"

4.3.2 From Text Modality to Multimodalities. Multimodalities in CoT can be classified into two categories: input multimodalities and output multimodalities, depending on where the multimodal elements are introduced. Figure 7 illustrates these types of multimodalities in CoT.

Zhang et al. [188] first explored input multimodalities CoT, which enables the CoT to transcend beyond textual information, and propose a multimodal CoT called *MM-CoT*. Instead of prompting LLMs, MM-CoT focuses on fine-tuning. MM-CoT incorporates language (text) and vision (images) modalities into a two-stage framework: rationale generation and answer inference. MM-CoT fine-tunes smaller LLMs and integrates language and visual modalities using a gated fusion mechanism. The results of this approach have demonstrated that incorporating visual information can enhance the LLM's ability to generate reasoning paths and mitigate the hallucination challenges, resulting in improved performance.

Based on Zhang et al. [188], Yao et al. [177] first introduced graph structures into input multimodalities CoT and proposes a two-stage pipeline, GoT-Input (Graph-of-Thought-Input).

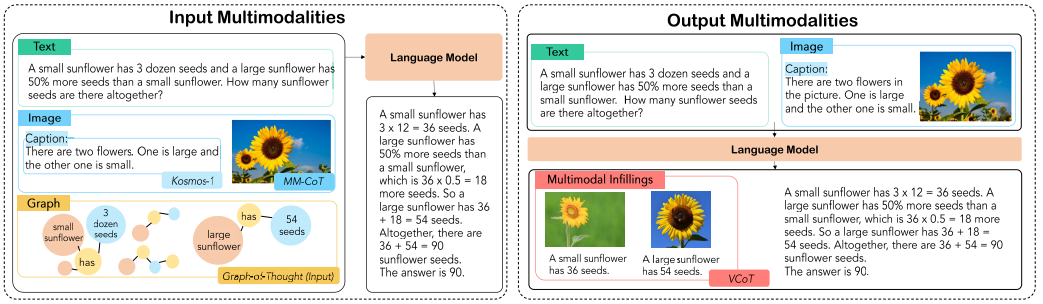

Fig. 7. Formulation of multimodalities CoT. We categorized multimodalities in CoT into two types: (i) input multimodalities, in which various modalities such as text, image [188], caption [46], and graph (Yao et al. [177]) are incorporated into the model's input, and (ii) output multimodalities, in which multimodalities, including text and image [114], are introduced into the model's output.

Different from GoT-Rationale [9], which models the thought generation process as a graph structure, GoT-Input centers its attention on modeling thought graphs derived from CoT rationales to enhance the model's reasoning capabilities. In the first stage, the model generates the rationale given the input question and a thought graph built by leveraging open IE systems to extract the sub-verb-obj triplets from the input. In the second stage, the model generates the answer given the question and the generated rationales as inputs and a new thought graph based on the input text. GoT employs different encoders for text, graph, and image (optional) respectively and enhances the deductive reasoning capability through the usage of GNN. GoT then fuses the features using a gated fusion method to generate the final answer. By modeling the non-sequential nature of human thinking within LLMs, GoT proves to enhance the LLMs with deductive reasoning abilities.

In the work of Huang et al. [46], KOSMOS-1, a multimodal language model capable of processing various modalities, was introduced. The authors explored a multimodal CoT prompting approach using KOSMOS-1. In the initial stage, when presented with an image, the authors employed the prompt "Introduce this picture in detail:" to generate a detailed description of the image as the rationale. Subsequently, the model was provided with both the rationale and a task-specific prompt to generate the final results.

In contrast to the input multimodalities CoT mentioned previously, VCoT (Visual Chain of Thoughts) [114] introduces multimodalities into the output space. VCoT initiates the process by generating captions for visual elements and identifying multipoint foveation to maintain input sequence consistency when producing multimodal infillings. Subsequently, it employs a recursive approach to generate multimodal infillings, encompassing both images and image captions. This is achieved through a combination of novelty-driven recursive infilling and consistency-driven visual augmentation. These strategies are employed to enhance interpretability for multi-step reasoning and bridge logical gaps, ultimately contributing to improved downstream task performance.

4.3.3 From Complex Reasoning Tasks to General-Purpose Tasks. The applicability of CoT has expanded from its initial utilization in mathematical, commonsense, and logical reasoning tasks to encompass a wide range of NLP tasks.

Wang et al. [151] introduced CoT into the realm of summarization and proposed the SumCoT (Summary Chain-of-Thought) technique with the aim of guiding LLMs to generate summaries in a step-by-step fashion. This approach enables the integration of more fine-grained details from source documents into the final summaries. SumCoT begins by instructing LLMs to extract core

news elements from the source document using manually designed guiding question prompts. Subsequently, it involves integrating the extracted elements along with additional details from the source documents to produce comprehensive and informative summaries.

Li et al. [61] proposed Self-Prompting LLMs for ODQA (Open-Domain QA). Self-Prompting consists of two stages. In the first stage, the model tasks the LLM with generating a pseudo ODQA dataset by prompting it to automatically construct QA pairs with context paragraphs and explanations. In the second stage, the model dynamically selects a few examples from a pool using a clustering-based retrieval method to serve as context demonstrations. These selected examples aid in understanding and answering specific questions.

He et al. [38] explored the CoT technique in machine translation and introduced MAPS (Multi-Aspect Prompting and Selection). Drawing inspiration from strategies employed by human translators, MAPS breaks down the machine translation process into several steps. It requires the LLM to initially discern the topics and keywords of the sentence awaiting translation, and then to retrieve analogous example sentences. By integrating this extracted knowledge, the LLM produces more accurate translations.

In addition to the classical NLP tasks mentioned earlier, numerous studies have actively explored the integration of CoT reasoning in the development of automated intelligent agents within scientific domains.

Singhal et al. [128] presented MultiMedQA, a benchmark that combines six existing open question answering datasets, and HealthSearchQA, a new free-response dataset of medical questions searched online. Based on the benchmark, the authors then proposed instruction prompt tuning to further align Flan-PaLM to the medical domain, producing Med-PaLM. Specifically, the authors used the soft prompt as an initial prefix shared across multiple medical datasets, followed by the relevant task-specific manual exemplars or instructions along with the target question. Following the CoT reasoning format, Med-PaLM's answers to consumer medical questions compared favorably with clinician-generated answers, demonstrating the effectiveness of instruction prompt tuning. The research provides a glimpse into the opportunities and challenges of applying LLMs to the medical domain.

Bran et al. [78] incorporated CoT into the field of chemistry and proposed ChemCrow, a chemistry agent powered by LLM. Designed to tackle a wide spectrum of challenges spanning organic synthesis, drug discovery, and materials design, ChemCrow operates within the structured CoT reasoning format. Specifically, ChemCrow initially assembles a toolkit using various chemistry-related packages and software tools. The LLM in ChemCrow, guided by CoT reasoning principles, embarks on an automated and iterative CoT process. It begins by assessing the current state of the task, considering its alignment with the ultimate objective, planning the next steps and the choice of tools accordingly, and finally solving the problem. Through the integration of 18 expert-designed tools, the LLM's performance in chemistry-related tasks is significantly improved. By integrating the CoT reasoning format, ChemCrow showcases its capacity to independently plan and execute a range of chemical syntheses, including an insect repellent, three organocatalysts, and even the discovery of a novel chromophore. This exemplifies its effectiveness in automating a diverse array of chemical tasks.

5 Toward Language Agents

With improved capabilities by the advanced techniques presented previously, CoT reasoning has yielded a broader impact on the **artificial intelligence (AI)** community, notably fueling the development of autonomous agents in real life. Building intelligent autonomous agents that are capable of learning and acting in a distinct environment is a long-standing goal of AI [39, 80, 120, 145, 158, 162, 197]. In light of the swift advancements detailed previously, CoT

Table 2. Technical Comparison of Representative Agents

Agent	Type	Memory			Methodology	Domain	Environment Interaction		External Tools		
		Operation	Short Term	Long Term			Modality	Model	Web	Code	Other
CAMEL [56]	Communicative	Prompt	✓	-	Prompting	AI Society & Coding	Text	GPT-3.5-Turbo	-	-	-
Generative Agents [98]	Communicative	Tree Search	✓	✓	Prompting	AI Society	Text	GPT-3.5-Turbo	-	-	-
Voyager [143]	Communicative	Tree Search	✓	✓	Prompting	MineCraft	Text	GPT-4	-	-	-
GITM [200]	Communicative	Tree Search	✓	✓	Prompting	MineCraft	Text	GPT-3.5-Turbo	-	-	-
MetaGPT [42]	Communicative	Text Retrieval	✓	✓	Prompting	AI Society & Coding	Text	GPT-4	✓	✓	✓
ChatDev [103]	Communicative	Text Summary	✓	✓	Prompting	Software Engineering	Text	GPT-3.5-Turbo	-	✓	-
MAD [63]	Communicative	Prompt	✓	-	Prompting	Reasoning	Text	GPT-3.5-Turbo	-	-	-
Multiagent Debate [26]	Communicative	Prompt	✓	-	Prompting	Reasoning	Text	GPT-3.5-Turbo	-	-	-
FORD [164]	Communicative	Prompt	✓	-	Prompting	Reasoning	Text	GPT-4	-	-	-
AutoGPT [113]	Autonomous	Vector Search	✓	✓	Prompting	Task Management	Text Speech Image	GPT-4 DALL-e ElevenLabs	✓	✓	✓
BabyAGI [90]	Autonomous	Vector Search	✓	✓	Prompting	Task Management	Text	GPT-4	-	-	-
AgentGPT [112]	Autonomous	Vector Search	✓	✓	Prompting	Task Management	Text	GPT-4	✓	✓	✓
Auto-UI [187]	Autonomous	Prompt	✓	-	Fine-Tuning	UI Control	Text Image	FLAN-Alpaca BLIP-2	-	-	-
AITW [111]	Autonomous	Prompt	✓	-	Prompting	UI Control	Text	PaLM 2	-	-	-
DCACQ [84]	Autonomous	Prompt	✓	-	Fine-Tuning	Engineer	Text	BART	-	-	-
ChemCrow [78]	Autonomous	Prompt	✓	-	Prompting	Chemistry	Text	GPT-4	✓	✓	✓
Chatmof [49]	Autonomous	Text Search	-	✓	Prompting	Material Sciences	Text	GPT-4	✓	✓	✓
IASSE [11]	Autonomous	Vector Search	-	✓	Prompting	Scientific Experiments	Text	GPT-4	✓	✓	✓
TE [2]	Communicative	Prompt	✓	-	Prompting	Social Science	Text	GPT-4	-	-	-
CodePlan [7]	Autonomous	Tree Search	✓	✓	Prompting	Coding	Text	GPT-4	-	✓	-
VIMA [48]	Communicative	Prompt	✓	-	Fine-Tuning	Robot Manipulation	Text Image	T5 ViT	-	-	-
React [174]	Communicative	Text Retrieval	✓	✓	Fine-Tuning	Decision Making	Text	PaLM-8B	✓	-	-
Reflexion [126]	Communicative	Text Retrieval	✓	✓	Prompting	Decision Making	Text	GPT-4	✓	-	-
ToRA [32]	Autonomous	Prompt	✓	-	Prompting	Mathematical Reasoning	Text	GPT-4	-	✓	-
Toolformer [119]	Autonomous	Text Retrieval	✓	✓	Fine-Tuning	Mathematical Reasoning	Text	GPT-J	✓	✓	✓
Fireact [15]	Autonomous	Prompt	✓	-	Fine-Tuning	Question Answering	Text	GPT-3.5	✓	-	-

We classify the memory modules into two main types: short-term memory and long-term memory. As defined in Section 5.2.2, short-term memory is dynamic in nature and can be easily read and written via prompts. Long-term memory, however, is static and is typically stored in a database, accessible through various retrieval methods, including tree search, text search, and vector retrieval. For the external tools module, we divide the tools into three types: Web search (Web), Code interpreter (Code), and other tools (Other). More details on tool use can be found in Section 5.1.3.

reasoning approaches have been leveraged for perception, memory, and reasoning, and language agents, thereby enabling interaction within increasingly complex environments. These abilities serve as the foundation for developing autonomous agents that help solve complex tasks through human-agent and agent-agent collaboration.

As a result, LLM-based language agents, empowered by CoT techniques, have emerged in a wide range of research areas, such as engineering [56, 84, 103], natural sciences [11, 49, 78], and social sciences [2, 79]. Those language agents are capable of following language instructions and executing actions or even self-evolving in real-world or simulated environments. The representative application scenarios of agents include autonomous control [48, 111], research [11, 78], programming [7], and interaction [98]. A detailed technical comparison of existing agents is presented in Table 2. We will elaborate on the technical philosophy in the following discussion.

— ***What Is New in Language Agents Compared with RL Agents?*** The pursuit of developing generally intelligent agents has been a long-standing goal of AI research. In the early stages, research on agents primarily focused on RL techniques [87, 157]. RL agents are trained to make decisions through iterative interactions with an environment, receiving feedback in the form of rewards or penalties—correct moves are rewarded, whereas erroneous ones are penalized. This iterative process aims to minimize mistakes and maximize accurate

Table 3. Comparison between RL Agents and Language Agents

Aspect	RL Agents	Language Agents
Knowledge Acquisition	Primarily use RL techniques.	Leverage commonsense priors embedded in LLMs.
Training Process	Trained through iterative interactions with the environment, receiving rewards or penalties.	Adaptation to new tasks or environments with reduced dependence on human annotation, primarily through prompts.
Self-Evolution	Possess the ability to self-evolve through continuous interactions with the environment.	Face challenges in evolving parameters in response to environmental changes. Adaptation is mainly through prompts or costly fine-tuning of LLMs.
Limitations	Heavily rely on expert data and task-specific reward functions. Effectiveness is often confined to individual tasks.	Challenges in evolving parameters dynamically. Focus on language reasoning tasks, and may lack adaptability to broader tasks.
Transparency	Working mechanism often lacks transparency and interpretability.	Generally allow better interpretability with commonsense priors but have challenges in parameter evolution transparency.
Generalization	Limited generalization capabilities to novel tasks or domains.	Facilitate easy adaptation to new tasks or environments, reducing dependence on task-specific training. Primary focus remains on language tasks.
Future Goals	Aim to bridge the gap between RL agents and language agents, facilitating more versatile and adaptable architectures.	

decisions. RL agents possess a key trait: the ability to self-evolve through continuous interactions with their environments [5]. However, RL agents face limitations. They heavily rely on expert data and meticulously designed reward functions tailored for specific tasks. Consequently, their effectiveness is often confined to individual tasks, hampering their generalization capabilities to novel tasks or domains [51]. Furthermore, the inner workings of RL agents often lack transparency and interpretability [76, 172]. In contrast, language agents distinguish themselves from RL agents by leveraging commonsense priors embedded in LLMs. These priors reduce dependence on human annotation and trial-and-error learning, enabling easy adaptation to new tasks or environments and allowing better interpretability with CoT [121, 174]. However, language agents face challenges in evolving their parameters in response to environmental changes, primarily because they are predominantly adapted to environments through prompts or the heavy costs of fine-tuning the LLMs. While recent studies on language agents, such as Retroformer [175], have incorporated RL-like policies to enhance the capabilities of language agents, the focus remains largely limited to language reasoning tasks. It holds promise to see how to bridge the gap between RL agents and language agents to facilitate future architectures that can work generally with strong performance and high interpretability in complex environments. In consideration of the pros and cons of RL agents and language agents, please refer to Table 3 for more details.

The following discussion will introduce the basic concepts of language agents and show how CoT is utilized in those agents.

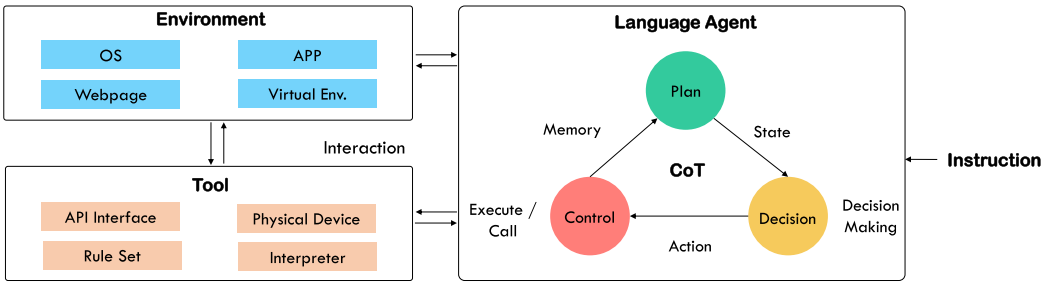

Fig. 8. General framework of language agents. Language agents are capable of following language instructions and executing actions in real-world or simulated environments.

5.1 General Framework

The landscape of language agent frameworks within the existing literature is notably diverse. We outline representative architectures in recent studies and summarize a cohesive and overarching conceptual framework for language agents.

Modulized Framework. Wang et al. [145] designed a modulized agent framework with four modules: (i) a profiling module to identify the role of the agent, (ii) a memory module to recall past behaviors, (iii) a planning module to plan future action, and (iv) an action module to translate the agent's decisions into specific outputs. Yin et al. [178] introduced the LUMOS framework for training language agents, which employs a unified data format and a modular architecture with planning, grounding, and execution modules. Similarly, Zhou et al. [197] presented a modulized agent framework that supports diverse features, including planning, memory, tool use, multi-agent communication, and fine-grained symbolic control.

Conceptual Framework. Lin et al. [66] proposed a conceptual agent framework called *SwiftSage*, which is inspired by the dual-process theory of human cognition with fast and slow thinking for complex interactive Tasks. Xi et al. [162] designed a conceptual agent framework with three components: (i) brain that undertakes basic tasks like memorizing, thinking, and decision making, (ii) perception that perceives and processes multimodal information from the external environment, and (iii) action that carries out the execution using tools and influences the surroundings. Sumers et al. [133] proposed another conceptual architecture for language agents called *CoALA*. CoALA organizes agents along three key dimensions: (i) information storage that is divided into working and long-term memories, (ii) action space that is divided into internal and external actions, and (iii) decision-making procedure that is structured as an interactive loop with planning and execution.

Although the preceding distinct architectures have been designed, recent technical research [98, 174, 175, 200] tends to follow the line of the conceptual framework by prompting LLMs to imitate the agent processes such as perception, memory, and reasoning. The basic assumption is that LLMs have already captured world knowledge to some extent [35], which can be induced by CoT prompting step by step.

Therefore, we summarize a general conceptual framework of language agents in view of technical practice as shown in Figure 8. Given a user instruction (also known as a *goal*), an agent needs to complete the task with multiple steps of interaction across the environment, possibly operating with tools. Without the loss of generality, we focus on a single agent when introducing the framework. It is worth noting that multiple agents can cooperate or compete with each other in a multi-agent environment. Before diving into the technical discussion, we first present the basic concepts of language agents (i.e., agent, environment, and tool use) as follows.

5.1.1 Agent Backbone Model. A language agent can be built upon either a single-modality LLM or a multimodal LLM. Completing a task often comes with multiple steps of interaction. The entire process is called an *episode*, which is composed of a series of *turns*. To accomplish the task, the agent needs to plan ahead, make decisions, and execute actions at each turn of the episode. The process of planning, decision making, and action execution may reflect the reasoning ability of LLMs as LLMs are exposed to real-world or virtual environments that do not exist during the pre-training of LLMs. In such environments, the LLM must perceive the world's knowledge and take action, in which cases we will show that CoT helps bridge the gap between the environment perception and the innate ability of LLMs.

Such agents expand the landscape of language models to compete in specific fields, including application operation, web searching, and web shopping. There are two popular types of language agents: autonomous agents and communicative agents. Typical examples of autonomous agents are AutoGPT [113], BabyAGI [90], and AgentGPT [112]. In contrast, communicative agents are personalized and socialized agents with human behaviors that can communicate [98, 143, 200], collaborate [42, 103], and debate [26, 63, 164] with each other. They are often deployed in immersive environments.

5.1.2 Environment Interaction. An intrinsic characteristic of language agents is communicating, interacting, and evolving with environments. Such environments include operation systems, third-party applications, webpages, and virtual environments. LLMs handle environments with two kinds of approaches, namely *environment parsing* and *multimodal perception*, depending on whether the LLM has the ability to model the multimodal inputs. Environment parsing refers to those approaches that leverage external tools such as OCR (optical character recognition) and icon detectors [134, 185] to parse the environment into textual elements (e.g., HTML layouts) as inputs to an LLM. In contrast, multimodal perception, also dubbed as first principles thinking [187], refers to using a multimodal LLM to simultaneously process the inputs in different modalities. To build a multimodal LLM, a popular way is to use a simple projection matrix to integrate a pre-trained large vision model (e.g., CLIP [109] and BLIP-2 [58]) into an LLM [71, 184]. More recent studies have also explored modeling the inputs of different modalities into the same vector space, thus resulting in any-to-any representation learning [46, 88, 160] and interleaved multimodal representation learning [57, 190].

5.1.3 Tool Use. Tool use can be seen as an expansion of a language model's ability boundary, compensating for parametric knowledge for reasoning and grounding the language model's capabilities to interact with environments [108]. Tools coming into play include knowledge bases, search engines, code interpreters, online models, applications, databases, and even bespoke tools specially created for specific tasks, overcoming the constraints of generic APIs [13, 60, 119, 138, 197].

The purpose of tool use comes with the three aspects presented next.

Action Execution. The language model is not confined to merely predicting the next action; it has the capability to execute it in the real environment. This includes everything from executing codes or queries through a JavaScript element selection on a webpage [196], executing programs via code interpreters or compilers [32, 34, 92, 115, 135], to interacting with online expert models which serve as callable APIs [29, 100, 124]. These steps can be dynamically adjusted with effective scaling of the tool set depending on task requirements and computational capacity [181].

External Knowledge Acquisition. Retrieval augmentation has been shown so effective that has been regarded as a standard solution to alleviate the factuality drawback [140, 174]. To empower the CoT process, up-to-date knowledge is accessible through search engines [50, 91], whereas domain-specific knowledge is accessible through expert candidates [29, 78]. The purpose of tool

use extends beyond augmenting the language model's scope; they enable language models to adapt to a complex environment or a vast application ecosystem and ensure that the information language models have access to is up-to-date, thereby reducing the propensity to generate non-factual information [145].

Reasoning and Verification. In the reasoning process, language models are sometimes prone to errors. Tools that provide accurate, real-time knowledge can help correct reasoning errors and formulate more accurate responses. Pieces of evidence from these tools are used to rewrite the initial output for self-correction [31]. Code LLMs can be further verified with execution results from program executors [92]. Multi-tool and multi-step planning and retrieval strategies, involving depth-first or breadth-first approaches, can be deployed for a deep or diverse range of possible pathways [74, 108].

5.1.4 Self-Evolution. Through interactions with the environment and the use of tools, LLM agents are gradually developing the capability for self-evolution. Self-evolution means that agents can autonomously improve themselves through a cycle of learning, refining, and updating their abilities, much like the human learning process. We categorize self-evolution into two main categories.

Environment Evolution. The first category focuses on the evolution of the environment in which the agent operates, specifically the evolution of the LLM-powered world simulator. For instance, Wu et al. [159] proposed a game engine named *Delta-Engine*, designed to simulate an evolving virtual world where both the environment and characters grow, change, and acquire new capabilities over time. The Delta-Engine begins with a base engine that outlines the static structure of the virtual world, then receives signals–such as user commands, environmental observations, or special triggers–that prompt a neural proxy to incrementally update the world. Similarly, Wu et al. [161] introduced an innovative approach using LLMs to create immersive, interactive drama experiences. The LLM evolves by guiding the progression of the narrative through a series of interconnected scenes, referred to as a Narrative Chain. This chain allows the model to autonomously direct the user's experience while maintaining coherence in the storyline. As users interact with characters and items in the virtual world, the LLM adapts the plot in response to these interactions, ensuring that the narrative remains dynamic and immersive.

Agent Evolution. In contrast to the first category, which primarily focuses on the evolution of environments generated by world engine agents, the second category explores the self-evolution of agents within these environments. Qiao et al. [105] introduced AutoAct, a framework that autonomously synthesizes planning trajectories in a CoT format, incorporating thought-action-observation loops. This system implements a division-of-labor strategy, wherein multiple sub-agents collaboratively generate and refine trajectories. This process involves filtering out incorrect trajectories and enhancing planning capabilities through a reflection mechanism, facilitating incremental self-evolution. Furthermore, Qian et al. [102] presented the ICE (Investigate-Consolidate-Exploit) framework, which supports self-evolution by enabling knowledge transfer across tasks, akin to human experiential learning. Within the ICE framework, the agent prunes and organizes successful experiences into workflows and pipelines. Workflows represent the planning phases of tasks, whereas pipelines encapsulate the sequences of execution. These standardized formats are retained in the agent's memory, promoting automatic reuse in subsequent tasks.

5.2 CoT Facilitates Agent Abilities

Language agents are placed in interactive loops with the external environment [133]. The interface loops can be elicited in three ways (Figure 1), namely perception, memory, and reasoning. CoT methods empower the agents from all three perspectives.

5.2.1 Perception as CoT. Prompting the agent to interpret the perception step by step, as a chain of perception, has been shown to improve the action success rate. It enhances the understanding of the environment or the context. Notably, Rawles et al. [111] found that using the CoT template "Answer: Let's think step by step. I see <Screen Caption>, I need to ..." substantially improves the action prediction accuracy. As an example shown in Figure 1, the prompt of perception as CoT can be "Let's think step by step. I see unrelated search results in the Google app." Furthermore, Zhang et al. [188] and Huang et al. [46] leveraged external tools to obtain the image captions as supplemental inputs to help improve the perception of the multimodal environments. The captions are placed in <Screen Caption> to organize the input prompt.

In addition to static environments (images, web pages, etc.), more complex dynamic environments pose greater challenges and highlight the evolving capabilities of LLM-powered world models. These world models function as internal environments, enhancing the agent's ability to learn and make decisions effectively [36]. By capturing both spatial and temporal features, the world model adeptly represents the environment. It often serves as an environment simulator, enabling agents to train and evaluate their actions in a simulated setting without direct real-world interaction. Hao et al. [37] introduced a novel framework called *RAP* (Reasoning via Planning), which innovatively incorporates the world model into the CoT reasoning framework. RAP transforms the LLM into a dual-function agent-both a reasoning agent and a world model. It employs Monte Carlo Tree Search to navigate through the extensive landscape of potential reasoning paths. In this framework, the LLM simulates future states of the reasoning process, taking on the role of the world model to predict the impact of actions on the world state. This simulation capability enables the generation of more coherent and substantiated reasoning paths. Guan et al. [33] further enhanced the LLM's ability to generate PDDL (Planning Domain Definition Language), thereby providing a formal and symbolic representation of actions, preconditions, and effects within specific domains. This proposed framework integrates LLMs with symbolic planning through PDDL, facilitating the construction of world models that receive iterative feedback to refine their predictions and assist agents in generating and validating strategic plans.

In addition to the one-way interpretation of perception, language agents can benefit significantly from integrating environmental feedback, especially in the context of multi-turn interactions where the environment is subject to alterations [19, 95]. Effectively integrating this feedback necessitates the implementation of a crucial method: self-correction with environment feedback [169, 175, 192, 195]. Self-correction entails exposing the model to intricate sequences of operations, encompassing tasks such as executing codes, conducting operations, and controlling robots. These operations can lead to execution failures and generate error messages. In this context, the agent is not only required to comprehend these environmental cues but must also actively engage in iterative error correction processes until the desired outcome is achieved. Consequently, the agent's performance within these dynamic environments serves as a direct indicator of its self-correction proficiency. This proficiency, in turn, showcases the agent's ability to assimilate feedback from the environment effectively. The seamless incorporation of such feedback not only refines the interpretive capacities but also enhances its overall functionality, making it pivotal in the realm of advanced language agents.

— ***Is Language-Centered Perception the Future?*** Multimodal perception stands as one of the key steps toward achieving artificial general intelligence. Current trends, likely inspired by the impressive reasoning capacities of language models, predominantly adopt a language-centered perception approach (Figure 9(a)). Typically, distinct encoders are utilized to process inputs from various modalities, such as images. The resulting encodings are then linked to an existing language model through cross-attention or supplementary adapters,

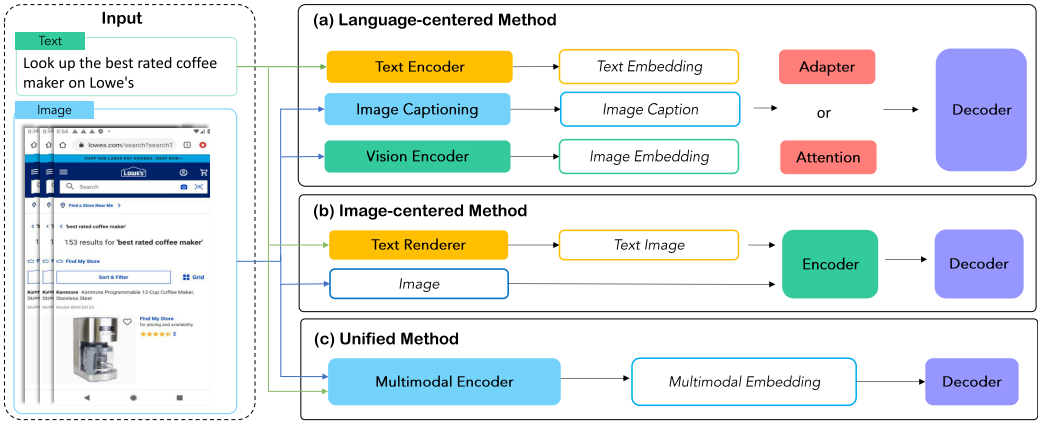

Fig. 9. Multimodal perception methods including the language-Centered method (a), the image-Centered method (b), and the unified method (c).

facilitating the integration of multimodal inputs into the language model's embedding space [3, 6, 25, 70, 160, 184]. In contrast to this prevailing language-centric modeling, Rust et al. [117] has proposed an image-centered approach (see Figure 9(b)) by rendering text as images, enabling the transfer of representations across languages based on orthographic similarity or the co-activation of pixels. To better align the inputs from different modalities and allow for convenient scaling up model parameters, recent research endeavors have explored a unified approach (see Figure 9(c)). For instance, in the context of vision-language modalities, instead of employing a separate image encoder, image patches are treated as tokens and linearly projected into the embedding layer of the transformer. These patches are then fused with the representations of language tokens, allowing for seamless integration [8, 46].

Even though various kinds of perception approaches, including language-centered, image-centered, and unified methods, have been proposed in the realm of agent perception, determining the most suitable choice remains a formidable challenge. This difficulty arises due to the involvement of more diverse and complex modalities such as auditory, tactile, and brain signals during interactions between agents and environments. Besides, these modalities often come with imbalanced data scales, complicating the perception process. Additionally, the diversity in types and formats of multimodal data poses challenges related to computation efficiency and the scalability of models. Exploring innovative methods to address these challenges will pave the way for the development of effective and efficient perception frameworks in the future.

5.2.2 Memory as CoT. A language agent is commonly equipped with both long-term memory and short-term memory [133, 147].

Short-Term Memory. Short-term memory is formed as temporal information that may be flexible to change in different steps of episodes (also known as *working memory* in the work of Sumers et al. [133]. Short-term memory is more temporal specific, offering explicit, recent context that facilitates the agent. On the one hand, short-term memory shows direct support and closer relations with the exact current state. On the other hand, short-term memory yields a relatively moderate impact on the whole environment. For example, short-term memory can be modeled within an episode of a multi-step task, the chain of action history [187], or the rationales or sub-question in the last several hops of multi-hop question answering [50, 174]. Due to the significant temporal character, short-term memory raises little storage concern.

Long-Term Memory. Long-term memory provides the agent with the capability to retain and recall static information over episodes [155]. In contrast to short-term memory, long-term memory is more general to the task, as a macroscopic and abstract understanding of the whole world. This can include *procedural memory* that stores the production system itself, *semantic memory* that stores facts about the world, and *episodic memory* that stores sequences of the agent's past behavior [133]. For example, given a goal, *upvote the latest post*, in the varied environment states, two chains of actions have been observed to accomplish the goal: (i) [*opening Instagram, going to home feed, looking at the latest post, upvoting the latest post*] and (ii) = [*go to the HOME screen, opening Instagram, going to home feed, looking at a post, upvoting the latest post*]. It can be found that atom actions [opening Instagram, going to home feed, looking at the latest post, upvoting the latest post] can serve as long-term memory for this goal (i.e., a chain of static memory).

Long-term memories can rely on both parametric and non-parametric knowledge storage. They can be from the trainable parameters of the language agents or maintained as external knowledge that can be leveraged through retrieval systems. For example, the earlier hops of former episodes are long-term memories from agent parameters, and the output action formulations are parametric long-term memories.

— ***Toward Efficient Memory Operation.*** Modeling memory as linear natural language sequences becomes inefficient as sequences lengthen during the agent's interaction with environments. Besides, the context window of LLMs is predetermined to be limited in length. To pursue more efficient memory operations, recent studies have explored two types of approaches: leveraging (i) tree search and (ii) vector retrieval.

Tree search. Memory can be stored with a tree structure and fetched by searching on the tree. Notably, MemWalker [16] empowered agents to access textual memory information through iterative prompting. In this approach, the agent initially processes the lengthy context into a tree of summary nodes. Upon receiving a query, the agent navigates this tree to search for relevant information and responds after gathering sufficient information. Similarly, GITM [200] proposed an LLM Decomposer that recursively decomposes goals into a sub-goal tree. The hierarchical tree structure helps the model explicitly capture the relationships between goals and corresponding plans in the memory. Park et al. [98] proposed the reflection tree to organize the memory of a communicative agent. When facing the trivial observations during the interaction with the environment, the agent periodically reflects on existing memories in a abstract manner, thus forming a reflection tree: "the leaf nodes of the tree represent the base observations, and the non-leaf nodes represent thoughts that become more abstract and higher-level the higher up the tree they are."

Vector retrieval. The other way to store memory is via vector storage [43, 198]. The vector database has become a key carrier for storing, managing, and retrieving high-dimensional data, such as the long-term memory of language agents. It can represent complex data types such as text, images, videos, and even structured data. AgentSims [67] employed a vector database to enable efficient storage and retrieval within long-term memory. Specifically, it stores daily memories as embeddings within this vector database. When the agent encounters new situations and necessitates the recall of past memories, the long-term memory system adeptly retrieves pertinent information, thereby ensuring the consistency of the agent's behavior.

5.2.3 Reasoning as CoT. Inspired by the success of eliciting LLMs' step-by-step reasoning abilities, CoT has also been applied in inducing the agents to reason via planning or decision making. More importantly, CoT methods for language agents require careful design to handle the action execution and state observation.

The gap between reasoning and action is bridged by combining interleaving thought, action, and observation [50, 126, 174]. By exploring the use of LLMs to generate both CoT traces and

task-specific actions in an interleaved manner, it has been found that reasoning and acting achieve mutual promotion. Reasoning traces help the model make action plans and handle exceptions, whereas actions allow the LLM to interface with external sources, such as knowledge bases or environments, to gather additional information for knowledge support. Xu et al. [166] detached the reasoning process from external observations to reduce token consumption during multiple steps of CoT.

Similarly, AgentBench [72] compelled language agents to complete tasks via "think" and "Act" steps. Further, Zhang and Zhang [187] proposed a chain-of-action techniques—leveraging a series of intermediate previous action histories and future action plans—to help the agent decide what action to execute, which transforms the decision making as a CoT reasoning problem.

— ***How to Expand the Capability of Agents?*** Currently, the mainstream interest is to apply CoT prompting approaches to elicit LLMs' reasoning abilities during the interaction with the environments as discussed previously. The basic hypothesis is that LLMs already have the prior knowledge to perform as the language agents for our concerned tasks, and CoT prompting approaches are effective in invoking the knowledge. Those prompting techniques have the advantage of flexibility and convenience because it is easy to design and adjust the prompts according to the task requirements and characteristics. However, LLM performance has shown to be sensitive to prompts, and there is a lack of evidence that LLM can actually learn domain knowledge from the prompts. Therefore, purely prompting methods may not be adequate to make LLMs generalizable to new domains. To expand the capability boundary of language agents, there is a recent interest in fine-tuning LLMs on curated datasets to build effective agents. Chen et al. [15] called for a rethinking of fine-tuning language models when the target tasks and data formats are known and enough data can be collected (e.g., possibly automatically with GPT-4). The results have revealed that fine-tuning can not only achieve strong generalization and robustness but also improve performance. Gou et al. [32] curated interleaved tool-use data composed of natural language CoT with tool-integrated programs. Then, a tool-integrated reasoning agent was trained on those high-quality annotations and achieved substantial performance gains on mathematical reasoning tasks.

6 Challenges

Despite the swift advancements in the realms of LLMs, CoT reasoning, and language agents, numerous promising challenges still beckon for deeper exploration, particularly pertaining to generalization to unseen domains, enhancing efficiency amidst redundant interactions, developing customizable agents, scaling up language agents, ensuring the safety of language agents, and capacity evaluation.

6.1 Generalization to Unseen Domains

Language agents have found extensive applications in practical fields such as engineering [56, 84, 103], natural sciences [11, 49, 78], and social sciences [2, 79]. Despite their widespread use, a significant challenge persists: adapting LLMs to specific, especially unseen domains. This challenge is twofold. The first is determining an efficient method for acquiring domain-specific knowledge, such as employing CoT prompting techniques. The limitations arise from the finite scope of knowledge acquisition during pre-training on textual corpora, lacking substantial interaction with the physical world. A second challenge is effectively adapting LLMs to diverse, unseen domains. Given the substantial variation in action spaces across tasks (e.g., drone control vs. web browsing), aligning the model's knowledge with the specific task requirements remains a formidable obstacle. These challenges underscore a critical gap in current research. The need to enhance LLMs'

adaptability to novel domains and help LLMs learn from environments is paramount, requiring innovative solutions that address both knowledge acquisition and effective task alignment.

Prompting and fine-tuning are widely used techniques to adapt pre-trained LLMs to new domains. However, it remains an underexplored area of when and how to leverage prompting (e.g., prompting pattern and reasoning format) and fine-tuning (e.g., instruction tuning) techniques to help LLMs generalize to unseen domains. In doing so, researchers can pave the way for more versatile and impactful applications of language agents across myriad fields.

6.2 Efficiency against Redundant Interactions

Completing a task necessitates intricate, multi-step interactions with the environment. This process results in extensive and repetitive logs, which have been identified as pivotal for task completion [187]. However, due to computational constraints, most studies utilize only a limited number of log steps [98]. Although recent advancements have expanded the capacity of LLMs to handle extended contexts [165], conducting inference based on these logs is hampered by the inherently slow speed of autoregressive LLMs. This issue is exacerbated in multi-agent interaction environments, where numerous agents generate a substantial volume of interaction logs.

To tackle this challenge, one potential solution is to incorporate a memory mechanism for storing and retrieving knowledge from these logs. However, the key challenge lies in exploring effective methods to discern salient knowledge and distill relevant information from the logs. Addressing this challenge is crucial for enhancing the efficiency of inference processes in complex, multi-agent scenarios.

6.3 Customizable Language Agents

LLMs are usually supposed to acquire general language ability and common knowledge through pre-training on large-scale corpora, and then cater to human preferences following instructions through further alignment tuning, including instruction tuning and RL from human feedback. However, users have specialized requirements and individual characteristics. Thus, building a customizable assistant from LLMs is of great importance.

Existing related studies mostly fall into three general methods. The first is customizable prompting, often with role or tool specifications. CAMEL [56] prompted LLM with formatted profiles of human-agent pairs to simulate the workflow of diverse groups of internet users or occupations. MetaAgents [62] prompted the language agent to play a specific role in some certain social context. ExpertPrompting [167] proposed to prompt LLMs to solve a problem conditioned on an expert identity profile that is best suited for the problem. RoCo [82] assigned robots with an LLM role to talk on their behalves, generating plans for practical tasks. Customizable ChatGPT has also been announced to comply with specified instructions, extra knowledge, and a combination of skills.[3] The second is customizable training. The gradient updates in the language model can further ensure the customizable alignment. Auto-UI [187] was trained on the Android UI control domain, achieving stable performance as an autonomous agent. For the communicative agent, Character-LLM [123] trained the LLMs with profiles and detailed scenes, enabling LLMs to mimic well-known people, like Beethoven. The third is customizable model editing. Besides training, editing is an alternative to changing stored knowledge in language agents, which improves the factuality and reliability of a customized assistant. ROME [85] and MEMIT [86] used the *locate-and-edit* method to correct wrong knowledge. Transformer-Patcher [47] further alleviated the error recurrence by real-time sequential editing. Beyond factual knowledge correctness, PersonalityEdit [83] changed the model response to match the Big Five personality traits.

[3] https://openai.com/blog/introducing-gpts

Despite the recent progress, the challenge of developing customizable agents still lies within three folds. First, existing studies focus on methods for practical applications in certain, separate domains. However, few considerations are oriented to the specific requirements of users. Second, language agent customization requires lightweight, efficient, and low-resource consumption, especially for user-level customization. Different from large-scale, general training, customization peruses effective methods that involve fewer data, partial parameters, or only elaborate designed prompts. Third, the balance between customization and information security needs to be maintained. The user's properties and records (e.g., age, gender, and medical record) may be exposed to an agent, resulting in a risk of privacy leakage.

6.4 Scaling Up Language Agents

Multi-agent systems have exhibited social phenomena [98, 143, 200]. Inspired by the observations, recent interest has considered scaling the number of language agents [56] to form a large-scale language model society. However, computation overhead is still an obstacle when modeling multi-agent communications [162]. In the realm of future prospects, the exploration of scaling unveils intriguing possibilities across two pivotal domains. First, there arises a profound curiosity concerning the potential emergence of novel capabilities within a singular agent amidst communication. Second, comprehending the implications of scaling, such as personality change and social phenomenon, becomes imperative in empowering language agents to address increasingly complex challenges. Furthermore, this comprehension serves as a linchpin in observing, detecting, and mitigating the risks entailed by potentially harmful behaviors, thereby ensuring the secure and beneficial evolution of these agents for the betterment of society.

6.5 Safety of Language Agents

Imagine a near future where intelligent agents are anticipated to seamlessly collaborate with humans and other agents, simplifying daily tasks and interacting with diverse environments. This convenience is accompanied by a significant challenge: ensuring the safety of these agents, especially during prolonged, multi-round interactions. For example, the popular user interface agents designed for web operation [196] and mobile device control [187] may result in privacy leakage and permission abuse. Shaikh et al. [122] called for attention to the bias and toxicity in Zero-Shot-CoT reasoning, as it tends to significantly induce the model to produce harmful or undesirable output, which may also bring negative effects in language agents. Effectively addressing the safety challenge demands a multifaceted approach. First, the exploration of more robust and controllable model architectures, coupled with an in-depth understanding of their underlying mechanisms, shows great promise. Delving into the intricacies of agent behavior and enhancing the reliability of their responses are pivotal in this endeavor. Second, the rapid evolution of attacks tailored for LLMs necessitates a reevaluation of traditional defense techniques.

Existing studies concerning LLM safety mainly focus on content safety, such as offensiveness, fairness, and bias of LLM-generated contents [186]. As language agents are exposed to multi-turn interactions in distinct environments possibly with operating tools, new safety risks may emerge at a systematic level [53, 168], ranging from instruction input, environment perception, and reasoning process, as well as tool use. We summarize three key properties of agent safety risks, including (i) new attacking types, such as operation attacking by environment injection [73], tool misuse [28], jailbreaking [23, 152], and privacy leakage [52]; (ii) new attacking surface during the interaction between agent-human, agent-agent, and agent-environment; and (iii) complex types of environments, such as operation systems, third-party applications, webpages, and virtual environment.

However, the safety of language languages has been underexplored. The definition of language agent safety has not yet reached an agreement. Novel attack methods, specifically designed for

language agents, present unique challenges. Consequently, innovative defense strategies must be developed to mitigate safety risks induced by these sophisticated attacks, particularly in complex environments. This dual focus on building benchmarking resources, enhancing internal safety measures, and fortifying defenses against external threats is paramount for ensuring the secure integration of intelligent agents into our daily lives.

6.6 Evaluation of Language Agents

Early studies in NLP mainly focus on assessing a specific ability of models—for example, machine translation, question answering, and summarization [14]. The evaluation tends to be dataset centered, which makes it hard to reflect the model's general ability. In the era of LLMs, more comprehensive benchmark datasets have been released, such as MMLU [40], BIG-Bench [130], and AGI Eval [193]. However, the major focus of those benchmark datasets is on the understanding and reasoning abilities of LLMs. Besides, they are mostly single-turn evaluations, which makes it hard to evaluate the planning and decision-making abilities of LLM in distinct environments.

There is an increasing interest in developing environment-centered evaluation approaches. As language agents are exposed to interactive environments, it remains challenging to evaluate those agents. Specifically, the measurement of task success might be task specific and ambiguous. For example, in a system control problem [111], a user instruction can be completed by different trajectories; however, it is hard to annotate all possible ways as gold labels for evaluation. To address the challenge, simulation-based evaluation [116, 147, 171] has attracted increasing interest. Execution feedback or external judgment can be used to measure if the task is successful or not. However, execution feedback is not always accessible in every kind of environment, and using external judgment may also include model bias [147].

Besides assessing the task success rate, it is critical to consider safety risks as discussed in Section 6.5. Furthermore, as language agents may evolve in the environments, especially in multi-agent communities, how to track and evaluate the agent properties is also a challenge.

7 Conclusion

Recently, CoT techniques have substantially enhanced the reasoning capabilities of LLMs. Going beyond the confines of reasoning tasks in NLP, CoT techniques have been expanded to facilitate the development of language agents. These agents have demonstrated the ability to comprehend language instructions and execute actions in diverse environments. This study meticulously examined the evolution from CoT reasoning to the automation of language agents, offering a comprehensive review and delving into key research topics. These topics included the foundational mechanics underpinning CoT techniques, the paradigm shift associated with CoT, and the emergence of language agents facilitated by CoT techniques. Furthermore, this research delineated several promising avenues for future exploration, including aspects related to generalization, efficiency, customization, scaling, and safety.

Acknowledgments

We thank Diyi Yang for providing valuable feedback on the draft.

References

[1] Adept. 2022. ACT-1: Transformer for Actions. Retrieved February 26, 2025 from https://www.adept.ai/act

[2] Gati V. Aher, Rosa I. Arriaga, and Adam Tauman Kalai. 2023. Using large language models to simulate multiple humans and replicate human subject studies. In *Proceedings of the 40th International Conference on Machine Learning (ICML'23)*. 337–371.

[3] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. 2022. Flamingo: A visual language model for few-shot learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS'22)*. 23716–23736.

[4] Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403* (2023).

[5] Hui Bai, Ran Cheng, and Yaochu Jin. 2023. Evolutionary reinforcement learning: A survey. *Intelligent Computing* 2 (2023), 0025.

[6] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-VL: A frontier large vision-language model with versatile abilities. *arXiv preprint abs/2308.12966* (2023).

[7] Ramakrishna Bairi, Atharv Sonwane, Aditya Kanade, Arun Iyer, Suresh Parthasarathy, Sriram Rajamani, B. Ashok, and Shashank Shet. 2024. CodePlan: Repository-level coding using LLMs and planning. *Proceedings of the ACM on Software Engineering* 1, FSE (2024), 675–698.

[8] Rohan Bavishi, Erich Elsen, Curtis Hawthorne, Maxwell Nye, Augustus Odena, Arushi Somani, and Sağnak Taşırlar. 2023. Fuyu-8B: A Multimodal Architecture for AI Agents. Retrieved February 26, 2025 from https://www.adept.ai/blog/fuyu-8b

[9] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence, the 36th Conference on Innovative Applications of Artificial Intelligence, and the 14th Symposium on Educational Advances in Artificial Intelligence (AAAI'24/IAAI'24/EAAI'24)*. 17682–17690.

[10] Zhen Bi, Ningyu Zhang, Yinuo Jiang, Shumin Deng, Guozhou Zheng, and Huajun Chen. 2024. When do program-of-thoughts works for reasoning? In *Proceedings of the 38th AAAI Conference on Artificial Intelligence, the 36th Conference on Innovative Applications of Artificial Intelligence, and the 14th Symposium on Educational Advances in Artificial Intelligence (AAAI'24/IAAI'24/EAAI'24)*. 17691–17699.

[11] Daniil A. Boiko, Robert MacKnight, and Gabe Gomes. 2023. Emergent autonomous scientific research capabilities of large language models. *arXiv preprint abs/2304.05332* (2023).

[12] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS'20)*. 1877–1901.

[13] Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. 2024. Large language models as tool makers. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.

[14] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology* 15, 3 (2024), 1–45.

[15] Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. FireAct: Toward language agent fine-tuning. *arXiv preprint abs/2310.05915* (2023).

[16] Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. 2023. Walking down the memory maze: Beyond context limit through interactive reading. *arXiv preprint abs/2310.05029* (2023).

[17] Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei A Zaharia, and James Y. Zou. 2025. Are more LLM calls all you need? Towards the scaling properties of compound AI systems. *Advances in Neural Information Processing Systems* 37 (2025), 45767–45790.

[18] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*. Published Online, November 1, 2023.

[19] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2024. Teaching large language models to self-debug. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.

[20] Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2024. Navigate through enigmatic labyrinth—A survey of chain of thought reasoning: Advances, frontiers and future. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1173–1203.

[21] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research* 25, 70 (2024), 1–53.

[22] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint abs/2110.14168* (2021).

[23] Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. 2024. MASTERKEY: Automated jailbreaking of large language model chatbots. In *Proceedings of the 31st Annual Network and Distributed System Security Symposium (NDSS'24)*.

[24] Shizhe Diao, Pengcheng Wang, Yong Lin, Rui Pan, Xiang Liu, and Tong Zhang. 2024. Active prompting with chain-of-thought for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1330–1350.

[25] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. 2023. PaLM-E: An embodied multimodal language model. In *Proceedings of the 11th International Conference on Learning Representations (ICLR'23)*. 8469–8488.

[26] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*.

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

[28] Xiaohan Fu, Zihan Wang, Shuheng Li, Rajesh K. Gupta, Niloofar Mireshghallah, Taylor Berg-Kirkpatrick, and Earlence Fernandes. 2023. Misusing tools in large language models with visual adversarial examples. *arXiv preprint arXiv:2310.03185* (2023).

[29] Yingqiang Ge, Wenyue Hua, Kai Mei, Jianchao Ji, Juntao Tan, Shuyuan Xu, Zelong Li, and Yongfeng Zhang. 2023. OpenAGI: When LLM meets domain experts. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. 5539–5568.

[30] Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did Aristotle use a laptop? A question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics* 9 (2021), 346–361.

[31] Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.

[32] Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2024. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.

[33] Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. 79081–79094.

[34] Izzeddin Gur, Hiroki Furuta, Austin V. Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2024. A real-world webagent with planning, long context understanding, and program synthesis. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.

[35] Wes Gurnee and Max Tegmark. 2024. Language models represent space and time. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.

[36] David Ha and Jürgen Schmidhuber. 2018. Recurrent world models facilitate policy evolution. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*. 2455–2467.

[37] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP'23)*. 8154–8173.

[38] Zhiwei He, Tian Liang, Wenxiang Jiao, Zhuosheng Zhang, Yujiu Yang, Rui Wang, Zhaopeng Tu, Shuming Shi, and Xing Wang. 2024. Exploring human-like translation strategy with large language models. *Transactions of the Association for Computational Linguistics* 12 (2024), 229–246.

[39] James Hendler. 1999. Is there an intelligent agent in your future? *Nature Web Matters* 11 (1999), 1–8.

[40] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *Proceedings of the 9th International Conference on Learning Representations (ICLR'21)*.

[41] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.

[42] Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.

[43] Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. 2023. ChatDB: Augmenting LLMs with databases as their symbolic memory. *arXiv preprint abs/2306.03901* (2023).
[44] 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*. 1049–1065.
[45] 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 *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.
[46] Shaohan Huang, Li Dong, Wenhui Wang, Yaru Hao, Saksham Singhal, Shuming Ma, Tengchao Lv, Lei Cui, Owais Khan Mohammed, Barun Patra, et al. 2023. Language is not all you need: Aligning perception with language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. 72096–72109.
[47] Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-Patcher: One mistake worth one neuron. In *Proceedings of the 11th International Conference on Learning Representations (ICLR'23)*.
[48] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. 2022. VIMA: General robot manipulation with multimodal prompts. In *Proceedings of the NeurIPS 2022 Foundation Models for Decision Making Workshop*.
[49] Yeonghun Kang and Jihan Kim. 2023. ChatMOF: An autonomous AI system for predicting and generating metal-organic frameworks. *arXiv preprint abs/2308.01423* (2023).
[50] Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2022. Demonstrate-Search-Predict: Composing retrieval and language models for knowledge-intensive NLP. *arXiv preprint abs/2212.14024* (2022).
[51] Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. 39648–39677.
[52] Siwon Kim, Sangdoo Yun, Hwaran Lee, Martin Gubri, Sungroh Yoon, and Seong Joon Oh. 2023. ProPILE: Probing privacy leakage in large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. 20750–20762.
[53] Megan Kinniment, Lucas Jun Koba Sato, Haoxing Du, Brian Goodrich, Max Hasin, Lawrence Chan, Luke Harold Miles, Tao R. Lin, Hjalmar Wijk, Joel Burget, et al. 2023. Evaluating language-model agents on realistic autonomous tasks. *arXiv preprint arXiv:2312.11671* (2023).
[54] Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS'22)*. 22199–22213.
[55] Soochan Lee and Gunhee Kim. 2023. Recursion of thought: A divide-and-conquer approach to multi-context reasoning with language models. In *Findings of the Association for Computational Linguistics: ACL 2023*. 623–658.
[56] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. 51991–52008.
[57] Huayang Li, Siheng Li, Deng Cai, Longyue Wang, Lemao Liu, Taro Watanabe, Yujiu Yang, and Shuming Shi. 2024. TextBind: Multi-turn interleaved multimodal instruction-following in the wild. In *Findings of the Association for Computational Linguistics: ACL 2024*. 9053–9076.
[58] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML'23)*.
[59] Junlong Li, Jinyuan Wang, Zhuosheng Zhang, and Hai Zhao. 2024. Self-prompting large language models for zero-shot open-domain QA. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 296–310.
[60] Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP'23)*. 3102–3116.
[61] Yuang Li, Yu Wu, Jinyu Li, and Shujie Liu. 2023. Prompting large language models for zero-shot domain adaptation in speech recognition. In *Proceedings of the 2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU'23)*. 1–8.
[62] Yuan Li, Yixuan Zhang, and Lichao Sun. 2023. MetaAgents: Simulating interactions of human behaviors for LLM-based task-oriented coordination via collaborative generative agents. *arXiv preprint abs/2310.06500* (2023).
[63] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP'24)*.

[64] Zhenwen Liang, Ye Liu, Tong Niu, Xiangliang Zhang, Yingbo Zhou, and Semih Yavuz. 2024. Improving LLM reasoning through scaling inference computation with collaborative verification. *CoRR abs/2410.05318* (2024). https://doi.org/10.48550/ARXIV.2410.05318

[65] 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 *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.

[66] Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. 2023. SwiftSage: A generative agent with fast and slow thinking for complex interactive tasks. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. 23813–23825.

[67] Jiaju Lin, Haoran Zhao, Aochi Zhang, Yiting Wu, Huqiuyue Ping, and Qin Chen. 2023. AgentSims: An open-source sandbox for large language model evaluation. *arXiv preprint abs/2308.04026* (2023).

[68] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 158–167.

[69] Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. 36407–36433.

[70] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 26296–26306.

[71] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. 34892–34916.

[72] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2024. AgentBench: Evaluating LLMs as agents. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.

[73] Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and Yang Liu. 2023. Prompt injection attack against LLM-integrated applications. *arXiv preprint abs/2306.05499* (2023).

[74] Zhaoyang Liu, Zeqiang Lai, Zhangwei Gao, Erfei Cui, Xizhou Zhu, Lewei Lu, Qifeng Chen, Yu Qiao, Jifeng Dai, and Wenhai Wang. 2023. ControlLLM: Augment language models with tools by searching on graphs. *arXiv preprint abs/2310.17796* (2023).

[75] Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. 2023. A survey of deep learning for mathematical reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 14605–14631.

[76] Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. 4765–4774.

[77] Koen Luwel, Ageliki Foustana, Patrick Onghena, and Lieven Verschaffel. 2013. The role of verbal and performance intelligence in children's strategy selection and execution. *Learning and Individual Differences* 24 (2013), 134–138.

[78] Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. 2024. Augmenting large language models with chemistry tools. *Nature Machine Intelligence* 6 (2024), 525–535.

[79] Zilin Ma, Yiyang Mei, and Zhaoyuan Su. 2023. Understanding the benefits and challenges of using large language model-based conversational agents for mental well-being support. *AMIA Annual Symposium Proceedings* 2023 (2023), 1105–1114.

[80] Pattie Maes. 1995. Agents that reduce work and information overload. In *Readings in Human–Computer Interaction: Toward the Year 2000* (2nd ed.), Ronald M. Baecker, Jonathan Grudin, William A. S. Buxton, and Saul Greenberg (Eds.). Elsevier, 811–821.

[81] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2023. Teaching small language models to reason. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 1773–1781.

[82] Zhao Mandi, Shreeya Jain, and Shuran Song. 2024. RoCo: Dialectic multi-robot collaboration with large language models. In *Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA'24)*. 286–299.

[83] Shengyu Mao, Ningyu Zhang, Xiaohan Wang, Mengru Wang, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024. Editing personality for LLMs. In *Proceedings of the 13th National CCF Conference on Natural Language Processing and Chinese Computing (NLPCC'24)*. 241–254.

[84] Nikhil Mehta, Milagro Teruel, Xin Deng, Sergio Figueroa Sanz, Ahmed Awadallah, and Julia Kiseleva. 2024. Improving grounded language understanding in a collaborative environment by interacting with agents through help feedback. In *Findings of the Association for Computational Linguistics: EACL 2024*. 1306–1321.

[85] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS'22)*. 17359–17372.

[86] Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *Proceedings of the 11th International Conference on Learning Representations (ICLR'23)*.

[87] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[88] Seungwhan Moon, Andrea Madotto, Zhaojiang Lin, Tushar Nagarajan, Matt Smith, Shashank Jain, Chun-Fu Yeh, Prakash Murugesan, Peyman Heidari, Yue Liu, et al. 2023. AnyMAL: An efficient and scalable any-modality augmented language model. *arXiv:2309.16058 [cs.LG]* (2023).

[89] Louis-Philippe Morency, Amir Zadeh, and Paul Liang. 2022. Advanced Multimodal Machine Learning. Retrieved February 26, 2025 from https://cmu-multicomp-lab.github.io/adv-mmml-course/spring2022/schedule/11877_week5.pdf

[90] Yohei Nakajima. 2023. BabyAGI. Retrieved February 26, 2025 from https://github.com/yoheinakajima/babyagi

[91] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint abs/2112.09332* (2021).

[92] Ansong Ni, Srini Iyer, Dragomir Radev, Ves Stoyanov, Wen-Tau Yih, Sida I. Wang, and Xi Victoria Lin. 2023. LEVER: Learning to verify language-to-code generation with execution. In *Proceedings of the 40th International Conference on Machine Learning (ICML'23)*.

[93] Xuefei Ning, Zinan Lin, Zixuan Zhou, Zifu Wang, Huazhong Yang, and Yu Wang. 2023. Skeleton-of-thought: Large language models can do parallel decoding. In *Proceedings of the Efficient Natural Language and Speech Processing Workshop (ENLSP-III)*.

[94] Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. 2022. Show your work: Scratchpads for intermediate computation with language models. In *Proceedings of the Deep Learning for Code Workshop*.

[95] Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. 2024. Is self-repair a silver bullet for code generation? In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.

[96] Siru Ouyang, Zhuosheng Zhang, Bing Yan, Xuan Liu, Yejin Choi, Jiawei Han, and Lianhui Qin. 2024. Structured chemistry reasoning with large language models. In *Proceedings of the 41st International Conference on Machine Learning*. https://openreview.net/forum?id=7R3pzxTSlg

[97] Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2024. Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies. *Transactions of the Association for Computational Linguistics* 12 (2024), 484–506.

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

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

[100] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive APIs. *arXiv preprint abs/2305.15334* (2023).

[101] Ben Prystawski, Michael Li, and Noah Goodman. 2023. Why think step by step? Reasoning emerges from the locality of experience. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. 70926–70947.

[102] Cheng Qian, Shihao Liang, Yujia Qin, Yining Ye, Xin Cong, Yankai Lin, Yesai Wu, Zhiyuan Liu, and Maosong Sun. 2024. Investigate-Consolidate-Exploit: A general strategy for inter-task agent self-evolution. *CoRR abs/2401.13996* (2024).

[103] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. 2024. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 15174–15186.

[104] Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2023. Reasoning with language model prompting: A survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 5368–5393.

[105] Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Eleanor Jiang, Huajun Chen, et al. 2024. AutoAct: Automatic agent learning from scratch for QA via self-planning. In *Proceedings of the ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

[106] Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is ChatGPT a general-purpose natural language processing task solver? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP'23)*. 1339–1384.

[107] Libo Qin, Qiguang Chen, Fuxuan Wei, Shijue Huang, and Wanxiang Che. 2023. Cross-lingual prompting: Improving zero-shot chain-of-thought reasoning across languages. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP'23)*. 2695–2709.

[108] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.

[109] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML'21)*. 8748–8763.

[110] Sue Ramsden, Fiona M. Richardson, Goulven Josse, Michael S. C. Thomas, Caroline Ellis, Clare Shakeshaft, Mohamed L. Seghier, and Cathy J. Price. 2011. Verbal and non-verbal intelligence changes in the teenage brain. *Nature* 479, 7371 (2011), 113–116.

[111] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2023. Android in the Wild: A large-scale dataset for Android device control. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. 59708–59728.

[112] Reworkd. 2023. AgentGPT. Retrieved February 26, 2025 from https://github.com/reworkd/AgentGPT

[113] Toran Bruce Richards. 2023. Auto-GPT: An autonomous GPT-4 experiment. Retrieved February 26, 2025 from https://github.com/Significant-Gravitas/Auto-GPT

[114] Daniel Rose, Vaishnavi Himakunthala, Andy Ouyang, Ryan He, Alex Mei, Yujie Lu, Michael Saxon, Chinmay Sonar, Diba Mirza, and William Yang Wang. 2023. Visual chain of thought: Bridging logical gaps with multimodal infillings. *arXiv preprint abs/2305.02317* (2023).

[115] Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Guoqing Du, Shiwei Shi, Hangyu Mao, Ziyue Li, Xingyu Zeng, et al. 2023. TPTU: Task planning and tool usage of large language model-based AI agents. In *Proceedings of the NeurIPS 2023 Foundation Models for Decision Making Workshop*.

[116] Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. 2024. Identifying the risks of LM agents with an LM-emulated sandbox. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.

[117] Phillip Rust, Jonas F. Lotz, Emanuele Bugliarello, Elizabeth Salesky, Miryam de Lhoneux, and Desmond Elliott. 2023. Language modelling with pixels. In *Proceedings of the 11th International Conference on Learning Representations (ICLR'23)*.

[118] Joseph J. Ryan and Shane J. Lopez. 2001. Wechsler Adult Intelligence Scale-III. In *Understanding Psychological Assessment*, William I. Dorfman and Michel Hersen (Eds.). Perspectives in Individual Differences. Springer, 19–42.

[119] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. 68539–68551.

[120] John R. Searle. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Vol. 626. Cambridge University Press.

[121] Dhruv Shah, Michael Robert Equi, Błażej Osiński, Fei Xia, Brian Ichter, and Sergey Levine. 2023. Navigation with large language models: Semantic guesswork as a heuristic for planning. In *Proceedings of the Conference on Robot Learning*. 2683–2699.

[122] Omar Shaikh, Hongxin Zhang, William Held, Michael Bernstein, and Diyi Yang. 2023. On second thought, let's not think step by step! Bias and toxicity in zero-shot reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 4454–4470.

[123] Yunfan Shao, Linyang Li, Junqi Dai, and Xipeng Qiu. 2023. Character-LLM: A trainable agent for role-playing. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP'23)*. 13153–13187.

[124] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. 38154–38180.

[125] Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. 2023. Language models are multilingual chain-of-thought reasoners. In *Proceedings of the 11th International Conference on Learning Representations (ICLR'23)*.

[126] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. 8634–8652.

[127] Kashun Shum, Shizhe Diao, and Tong Zhang. 2023. Automatic prompt augmentation and selection with chain-of-thought from labeled data. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 12113–12139.

[128] Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature* 620, 7972 (2023), 172–180.

[129] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR abs/2408.03314* (2024). https://doi.org/10.48550/ARXIV.2408.03314

[130] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md. Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*. Published Online, May 11, 2023.

[131] Kaya Stechly, Matthew Marquez, and Subbarao Kambhampati. 2023. GPT-4 doesn't know it's wrong: An analysis of iterative prompting for reasoning problems. In *Proceedings of the NeurIPS 2023 Foundation Models for Decision Making Workshop*.

[132] Robert J. Sternberg, Janet S. Powell, and Daniel B. Kaye. 1982. The nature of verbal comprehension. *Poetics* 11, 2 (1982), 155–187.

[133] Theodore Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas Griffiths. 2024. Cognitive architectures for language agents. *Transactions on Machine Learning Research*. Published Online, February 22, 2024.

[134] Srinivas Sunkara, Maria Wang, Lijuan Liu, Gilles Baechler, Yu-Chung Hsiao, Jindong Chen, Abhanshu Sharma, and James W. W. Stout. 2022. Towards better semantic understanding of mobile interfaces. In *Proceedings of the 29th International Conference on Computational Linguistics (COLING'22)*. 5636–5650.

[135] Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. ViperGPT: Visual inference via Python execution for reasoning. In *Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV'23)*. 11888–11898.

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

[137] Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Steven Zheng, et al. 2022. UL2: Unifying language learning paradigms. In *Proceedings of the 10th International Conference on Learning Representations (ICLR'22)*.

[138] XAgent Team. 2023. XAgent: An autonomous agent for complex task solving. *XAgent Blog*.

[139] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. LaMDA: Language models for dialog applications. *arXiv preprint abs/2201.08239* (2022).

[140] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 10014–10037.

[141] Karthik Valmeekam, Matthew Marquez, and Subbarao Kambhampati. 2023. Investigating the effectiveness of self-critiquing in LLMs solving planning tasks. In *Proceedings of the NeurIPS 2023 Foundation Models for Decision Making Workshop*.

[142] Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2717–2739.

[143] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*. Published Online, March 19, 2024.

[144] Jun Wang, Meng Fang, Ziyu Wan, Muning Wen, Jiachen Zhu, Anjie Liu, Ziqin Gong, Yan Song, Lei Chen, Lionel M. Ni, et al. 2024. OpenR: An open source framework for advanced reasoning with large language models. *CoRR abs/2410.09671* (2024). https://doi.org/10.48550/ARXIV.2410.09671

[145] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science* 18, 6 (2024), 186345.

[146] Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-Solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2609–2634.

[147] Lei Wang, Jingsen Zhang, Hao Yang, Zhiyuan Chen, Jiakai Tang, Zeyu Zhang, Xu Chen, Yankai Lin, Ruihua Song, Wayne Xin Zhao, et al. 2023. User behavior simulation with large language model based agents. *arXiv preprint arXiv:2306.02552* (2023).

[148] Xinyi Wang and William Yang Wang. 2023. Reasoning ability emerges in large language models as aggregation of reasoning paths: A case study with knowledge graphs. In *Proceedings of the Workshop on Efficient Systems for Foundation Models (ICML'23)*.

[149] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Rationale-augmented ensembles in language models. *arXiv preprint abs/2207.00747* (2022).

[150] 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 *Proceedings of the 11th International Conference on Learning Representations (ICLR'23)*.

[151] Yiming Wang, Zhuosheng Zhang, and Rui Wang. 2023. Element-aware summarization with large language models: Expert-aligned evaluation and chain-of-thought method. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 8640–8665.

[152] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. Jailbroken: How does LLM safety training fail? In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. 80079–80110.

[153] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Published Online, August 31, 2022.

[154] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS'22)*. 24824–24837.

[155] Lilian Weng. 2023. LLM Powered Autonomous Agents. Retrieved February 26, 2025 from https://lilianweng.github.io

[156] Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 2550–2575.

[157] David E. Wilkins. 2014. *Practical Planning: Extending the Classical AI Planning Paradigm*. Elsevier.

[158] Michael Wooldridge and Nicholas R. Jennings. 1995. Intelligent agents: Theory and practice. *Knowledge Engineering Review* 10, 2 (1995), 115–152.

[159] Hongqiu Wu, Zekai Xu, Tianyang Xu, Jiale Hong, Weiqi Wu, Hai Zhao, Min Zhang, and Zhezhi He. 2024. Scaling virtual world with delta-engine. *arXiv preprint arXiv:2408.05842* (2024).

[160] Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. 2024. NExT-GPT: Any-to-any multimodal LLM. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*.

[161] Weiqi Wu, Hongqiu Wu, Lai Jiang, Xingyuan Liu, Hai Zhao, and Min Zhang. 2024. From role-play to drama-interaction: An LLM solution. In *Findings of the Association for Computational Linguistics: ACL 2024*. 3271–3290.

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

[163] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit Bayesian inference. In *Proceedings of the 10th International Conference on Learning Representations (ICLR'22)*.

[164] Kai Xiong, Xiao Ding, Yixin Cao, Ting Liu, and Bing Qin. 2023. Examining inter-consistency of large language models collaboration: An in-depth analysis via debate. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 7572–7590.

[165] Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. 2024. Effective long-context scaling of foundation models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 4643–4663.

[166] Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. 2023. ReWOO: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint abs/2305.18323* (2023).

[167] Benfeng Xu, An Yang, Junyang Lin, Quan Wang, Chang Zhou, Yongdong Zhang, and Zhendong Mao. 2023. Expert-Prompting: Instructing large language models to be distinguished experts. *arXiv preprint abs/2305.14688* (2023).

[168] Liang Xu, Kangkang Zhao, Lei Zhu, and Hang Xue. 2023. SC-Safety: A multi-round open-ended question adversarial safety benchmark for large language models in Chinese. *arXiv preprint abs/2310.05818* (2023).

[169] Yiheng Xu, Hongjin Su, Chen Xing, Boyu Mi, Qian Liu, Weijia Shi, Binyuan Hui, Fan Zhou, Yitao Liu, Tianbao Xie, et al. 2024. Lemur: Harmonizing natural language and code for language agents. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.

[170] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.
[171] Sherry Yang, Yilun Du, Seyed Kamyar Seyed Ghasemipour, Jonathan Tompson, Leslie Pack Kaelbling, Dale Schuurmans, and Pieter Abbeel. 2024. Learning interactive real-world simulators. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.
[172] Zhao Yang, Song Bai, Li Zhang, and Philip H. S. Torr. 2018. Learn to interpret Atari agents. *arXiv preprint abs/1812.11276* (2018).
[173] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. 11809–11822.
[174] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2022. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the 10th International Conference on Learning Representations (ICLR'22)*.
[175] Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, et al. 2024. Retroformer: Retrospective large language agents with policy gradient optimization. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.
[176] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. 2024. A survey on large language model (LLM) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing* 4, 2 (2024), 100211.
[177] Yao Yao, Zuchao Li, and Hai Zhao. 2024. GoT: Effective graph-of-thought reasoning in language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*. 2901–2921.
[178] Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. 2024. Lumos: Learning agents with unified data, modular design, and open-source LLMs. In *Proceedings of the ICLR 2024 Workshop on Large Language Model (LLM) Agents*.
[179] Fei Yu, Hongbo Zhang, Prayag Tiwari, and Benyou Wang. 2024. Natural language reasoning, a survey. *ACM Computing Surveys* 56, 12 (2024), Article 304, 39 pages.
[180] Zihan Yu, Liang He, Zhen Wu, Xinyu Dai, and Jiajun Chen. 2023. Towards better chain-of-thought prompting strategies: A survey. *arXiv preprint abs/2310.04959* (2023).
[181] Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi Fung, Hao Peng, and Heng Ji. 2024. CRAFT: Customizing LLMs by creating and retrieving from specialized toolsets. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.
[182] Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2024. MAmmoTH: Building math generalist models through hybrid instruction tuning. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.
[183] Zhenrui Yue, Honglei Zhuang, Aijun Bai, Kai Hui, Rolf Jagerman, Hansi Zeng, Zhen Qin, Dong Wang, Xuanhui Wang, and Michael Bendersky. 2024. Inference scaling for long-context retrieval augmented generation. *CoRR abs/2410.04343* (2024). https://doi.org/10.48550/ARXIV.2410.04343
[184] Renrui Zhang, Jiaming Han, Chris Liu, Aojun Zhou, Pan Lu, Yu Qiao, Hongsheng Li, and Peng Gao. 2024. LLaMA-Adapter: Efficient fine-tuning of large language models with zero-initialized attention. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.
[185] Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, et al. 2021. Screen recognition: Creating accessibility metadata for mobile applications from pixels. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.
[186] Zhexin Zhang, Leqi Lei, Lindong Wu, Rui Sun, Yongkang Huang, Chong Long, Xiao Liu, Xuanyu Lei, Jie Tang, and Minlie Huang. 2024. SafetyBench: Evaluating the safety of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 15537–15553.
[187] Zhuosheng Zhang and Aston Zhang. 2024. You only look at screens: Multimodal chain-of-action agents. In *Findings of the Association for Computational Linguistics: ACL 2024*.
[188] Zhuosheng Zhang, Aston Zhang, Mu Li, hai zhao, George Karypis, and Alex Smola. 2024. Multimodal chain-of-thought reasoning in language models. *Transactions on Machine Learning Research*. Published Online, June 3, 2024.
[189] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR'23)*.
[190] Haozhe Zhao, Zefan Cai, Shuzheng Si, Xiaojian Ma, Kaikai An, Liang Chen, Zixuan Liu, Sheng Wang, Wenjuan Han, and Baobao Chang. 2024. MMICL: Empowering vision-language model with multi-modal in-context learning. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.
[191] Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. 2023. Verify-and-Edit: A knowledge-enhanced chain-of-thought framework. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 5823–5840.

[192] Zirui Zhao, Wee Sun Lee, and David Hsu. 2023. Large language models as commonsense knowledge for large-scale task planning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23)*. 31967–31987.

[193] Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2024. AGIEval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024*. 2299–2314.

[194] Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, et al. 2024. Solving challenging math word problems using GPT-4 code interpreter with code-based self-verification. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.

[195] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*.

[196] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2024. WebArena: A realistic web environment for building autonomous agents. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.

[197] Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shuai Wang, Jiamin Chen, Jintian Zhang, Jing Chen, Xiangru Tang, et al. 2024. Agents: An open-source framework for autonomous language agents. In *Proceedings of the ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

[198] Xuanhe Zhou, Guoliang Li, and Zhiyuan Liu. 2023. LLM as DBA. *arXiv preprint abs/2308.05481* (2023).

[199] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *Proceedings of the 11th International Conference on Learning Representations (ICLR'23)*.

[200] Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, et al. 2023. Ghost in the Minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *arXiv preprint abs/2305.17144* (2023).

[201] Jin Ziqi and Wei Lu. 2023. Tab-CoT: Zero-shot tabular chain of thought. In *Findings of the Association for Computational Linguistics: ACL 2023*. 10259–10277.

[202] Anni Zou, Zhuosheng Zhang, and Hai Zhao. 2024. AuRoRA: A one-for-all platform for augmented reasoning and refining with task-adaptive chain-of-thought prompting. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources, and Evaluation (LREC-COLING'24)*. 1801–1807.

Received 1 December 2023; revised 4 November 2024; accepted 14 February 2025