Tool Learning with Foundation Models

YUJIA QIN, Computer Science, Tsinghua University, Beijing, China
SHENGDING HU, Computer Science, Tsinghua University, Beijing, China
YANKAI LIN, Renmin University of China, Beijing, China
WEIZE CHEN, Computer Science, Tsinghua University, Beijing, China
NING DING, Computer Science, Tsinghua University, Beijing, China
GANQU CUI, Computer Science, Tsinghua University, Beijing, China

Authors' Contact Information: Yujia Qin, Computer Science, Tsinghua University, Beijing, China; e-mail: yujiaqin16@gmail.com; Shengding Hu, Computer Science, Tsinghua University, Beijing, China; e-mail: hsd20@mails.tsinghua.edu.cn; Yankai Lin (Corresponding author), Renmin University of China, Beijing, China; e-mail: mrlyk423@gmail.com; Weize Chen, Computer Science, Tsinghua University, Beijing, China; e-mail: chenweize1998@gmail.com; Ning Ding, Computer Science, Tsinghua University, Beijing, China; e-mail: dingn18@mails.tsinghua.edu.cn; Ganqu Cui, Computer Science, Tsinghua University, Beijing, China; e-mail: cgq22@mails.tsinghua.edu.cn; Zheni Zeng, Computer Science, Tsinghua University, Beijing, China; e-mail: ellen.zeng@foxmail.com; Xuanhe Zhou, Computer Science, Tsinghua University, Beijing, China; e-mail: zhouxh@cs. sjtu.edu.cn; Yufei Huang, Computer Science, Tsinghua University, Beijing, China; e-mail: huangyf20@mails.tsinghua.edu.cn; Chaojun Xiao, Computer Science, Tsinghua University, Beijing, China; e-mail: xcjthu@gmail.com; Chi Han, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA; e-mail: chihan3@illinois.edu; Yi Ren Fung, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA; e-mail: yifung2@illinois.edu; Yusheng Su, Computer Science, Tsinghua University, Beijing, China; e-mail: suys19@mails.tsinghua.edu.cn; Huadong Wang, Computer Science, Tsinghua University, Beijing, China; e-mail: huadw2012@163.com; Cheng Qian, Computer Science, Tsinghua University, Beijing, China; e-mail: qianc20@mails.tsinghua.edu.cn; Runchu Tian, Computer Science, Tsinghua University, Beijing, China; e-mail: runchutian@gmail.com; Kunlun Zhu, OpenBMB Inc., Beijing, China; e-mail: zhuklun@mail2.sysu.edu.cn; Shihao Liang, OpenBMB Inc., Beijing, China; e-mail: shihaoliang0828@gmail.com; Xingyu Shen, Computer Science, Tsinghua University, Beijing, China; email: xingyu-c21@mails.tsinghua.edu.cn; Bokai Xu, Computer Science, Tsinghua University, Beijing, China; e-mail: bokaixu@link.cuhk.edu.cn; Zhen Zhang, Computer Science, Tsinghua University, Beijing, China; e-mail: zhen-zha19@mails.tsinghua.edu.cn; Yining Ye, Computer Science, Tsinghua University, Beijing, China; e-mail: yeyn19@mails.tsinghua.edu.cn; Bowen Li, Computer Science, Tsinghua University, Beijing, China; e-mail: libowen.ne@gmail.com; Ziwei Tang, Beijing University of Posts and Telecommunications, Beijing, Beijing, China; e-mail: tzw@bupt.edu.cn; Jing Yi, Computer Science, Tsinghua University, Beijing, China; e-mail: 18715465095@163.com; Yuzhang Zhu, Computer Science, Tsinghua University, Beijing, China; e-mail: zhuyz21@mails.tsinghua.edu.cn; Zhenning Dai, Computer Science, Tsinghua University, Beijing, China; e-mail: daizn20@mails.tsinghua.edu.cn; Lan Yan, Computer Science, Tsinghua University, Beijing, China; e-mail: yan-l18@tsinghua.org.cn; Xin Cong, Computer Science, Tsinghua University, Beijing, China; e-mail: congxin1995@mail.tsinghua.edu.cn; Yaxi Lu, Computer Science, Tsinghua University, Beijing, China; e-mail: luyaxi@live.com; Weilin Zhao, Computer Science, Tsinghua University, Beijing, China; e-mail: zwl19@mails.tsinghua.edu.cn; Yuxiang Huang, Computer Science, Tsinghua University, Beijing, China; e-mail: huangyx21@mails.tsinghua.edu.cn; Junxi Yan, Computer Science, Tsinghua University, Beijing, China; e-mail: yanjx21@mails.tsinghua.edu.cn; Xu Han, Computer Science, Tsinghua University, Beijing, China; e-mail: han-xu@tsinghua.edu.cn; Xian Sun, Zhihu Inc., Beijing, China; e-mail: 3477704335@qq.com; Dahai Li, Zhihu Inc., Beijing, China; e-mail: 835571620@qq.com; Jason Phang, New York University, New York, New York, USA; e-mail: email@jasonphang.com; Cheng Yang, Beijing University of Posts and Telecommunications, Beijing, China; e-mail: albertyang33@gmail.com; Tongshuang Wu, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA; e-mail: sherryw@cs.cmu.edu; Heng Ji, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA; e-mail: hengji@illinois.edu; Guoliang Li, Tsinghua University, Beijing, China; e-mail: liguoliang@tsinghua.edu.cn; Zhiyuan Liu (Corresponding author), Tsinghua University, Beijing, China; e-mail: liuzy@tsinghua.edu.cn; Maosong Sun (Corresponding author), Tsinghua University, Beijing, China; e-mail: sms@tsinghua.edu.cn.

ACM 0360-0300/2024/12-ART101
https://doi.org/10.1145/3704435

ZHENI ZENG, Computer Science, Tsinghua University, Beijing, China
XUANHE ZHOU, Computer Science, Tsinghua University, Beijing, China
YUFEI HUANG, Computer Science, Tsinghua University, Beijing, China
CHAOJUN XIAO, Computer Science, Tsinghua University, Beijing, China
CHI HAN, University of Illinois at Urbana-Champaign, Urbana, United States
YI REN FUNG, University of Illinois at Urbana-Champaign, Urbana, United States
YUSHENG SU, Computer Science, Tsinghua University, Beijing, China
HUADONG WANG, Computer Science, Tsinghua University, Beijing, China
CHENG QIAN, Computer Science, Tsinghua University, Beijing, China
RUNCHU TIAN, Computer Science, Tsinghua University, Beijing, China
KUNLUN ZHU, OpenBMB Inc., Beijing, China
SHIHAO LIANG, OpenBMB Inc., Beijing, China
XINGYU SHEN, Computer Science, Tsinghua University, Beijing, China
BOKAI XU, Computer Science, Tsinghua University, Beijing, China
ZHEN ZHANG, Computer Science, Tsinghua University, Beijing, China
YINING YE, Computer Science, Tsinghua University, Beijing, China
BOWEN LI, Computer Science, Tsinghua University, Beijing, China
ZIWEI TANG, Beijing University of Posts and Telecommunications, Beijing, China
JING YI, Computer Science, Tsinghua University, Beijing, China
YUZHANG ZHU, Computer Science, Tsinghua University, Beijing, China
ZHENNING DAI, Computer Science, Tsinghua University, Beijing, China
LAN YAN, Computer Science, Tsinghua University, Beijing, China
XIN CONG, Computer Science, Tsinghua University, Beijing, China
YAXI LU, Computer Science, Tsinghua University, Beijing, China
WEILIN ZHAO, Computer Science, Tsinghua University, Beijing, China
YUXIANG HUANG, Computer Science, Tsinghua University, Beijing, China
JUNXI YAN, Computer Science, Tsinghua University, Beijing, China
XU HAN, Computer Science, Tsinghua University, Beijing, China
XIAN SUN, Zhihu Inc., Beijing, China
DAHAI LI, Zhihu Inc., Beijing, China
JASON PHANG, New York University, New York, United States
CHENG YANG, Beijing University of Posts and Telecommunications, Beijing, China
TONGSHUANG WU, Carnegie Mellon University, Pittsburgh, United States
HENG JI, University of Illinois at Urbana-Champaign, Urbana, United States
GUOLIANG LI, Tsinghua University, Beijing, China
ZHIYUAN LIU, Tsinghua University, Beijing, China
MAOSONG SUN, Tsinghua University, Beijing, China

Humans possess an extraordinary ability to create and utilize tools. With the advent of foundation models, artificial intelligence systems have the potential to be equally adept in tool use as humans. This paradigm, which is dubbed as *tool learning with foundation models*, combines the strengths of tools and foundation models to achieve enhanced accuracy, efficiency, and automation in problem-solving. This article presents a systematic investigation and comprehensive review of tool learning. We first introduce the background of tool learning, including its cognitive origins, the paradigm shift of foundation models, and the complementary roles of tools and models. Then we recapitulate existing tool learning research and formulate a general

framework: starting from understanding the user instruction, models should learn to decompose a complex task into several subtasks, dynamically adjust their plan through reasoning, and effectively conquer each subtask by selecting appropriate tools. We also discuss how to train models for improved tool-use capabilities and facilitate generalization in tool learning. Finally, we discuss several open problems that require further investigation, such as ensuring trustworthy tool use, enabling tool creation with foundation models, and addressing personalization challenges. Overall, we hope this article could inspire future research in integrating tools with foundation models.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence;**

Additional Key Words and Phrases: Tool use, foundation models, literature survey

ACM Reference Format:
Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Guoliang Li, Zhiyuan Liu, and Maosong Sun. 2024. Tool Learning with Foundation Models. *ACM Comput. Surv.* 57, 4, Article 101 (December 2024), 40 pages. https://doi.org/10.1145/3704435

1 Introduction

Tools are extensions of human capabilities designed to enhance productivity, efficiency, and problem-solving in human activities. Since the dawn of civilization, tools have been integral to the very essence of our existence [150]. Tool creation and utilization are motivated by a deep-rooted desire to overcome our physical limitations and discover new territories. More specifically, with advancements in tools, we can accomplish increasingly complex tasks with ease and efficiency, liberating time and resources to pursue more ambitious ventures. As such, tools have served as the crucial foundation upon which our cultural and social practices are built, transforming our modes of learning, communication, working, and entertainment, infusing these domains with new dimensions of accessibility and interactivity [38]. Throughout history, it is undeniable that human beings have played a pivotal role in the invention and manipulation of tools, which is a striking manifestation of intelligence [122]. Given the rise of **Artificial Intelligence** (**AI**), one natural question is, does AI possess the potential to be equally adept and capable as its creators?

The prerequisite of the manipulation of tools is a thorough comprehension of the tools' functionalities, as well as the ability to understand user intents and perform planning and reasoning for tool use. Before the advent of powerful foundation models [13], conducting tool-oriented AI research was exceedingly challenging. While certain basic tools could be fitted using shallow statistical models or deep neural models [2, 87, 104], their performance and stability remained inadequate to meet the demands of practical applications, let alone generalizing across various tools. This is due to the limitations of traditional supervised learning in capturing the complex operations essential for tool utilization and the insufficiency of trial-and-error approaches like **reinforcement learning** (**RL**) in mastering the extensive decision space associated with tool use. In a nutshell, the fundamental limitations in tool use by earlier AI lie in the insufficient capabilities of the models. Recently, the emergence of more capable foundation models, characterized by significantly improved capabilities, has rendered tool learning practicable. They have shown enormous semantic understanding capacity in diverse tasks, spanning the fields of **natural language processing** (**NLP**) [15], **computer vision** (**CV**) [111], biology [53], and so on. Additionally, they have demonstrated superior reasoning and decision-making abilities in complex interactive environments [89]. By harnessing

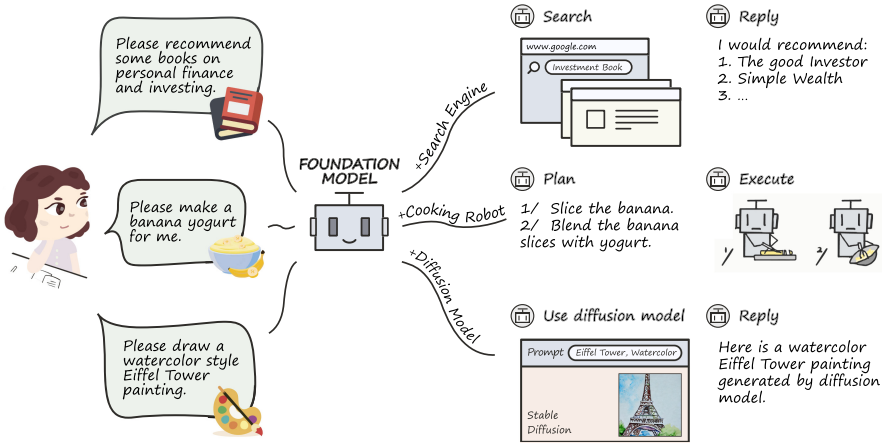


Fig. 1. Tool learning paradigm aims to combine the strengths of specialized tools and foundation models.

the extensive world knowledge garnered during pre-training, they can perform grounded actions and interact with the real world. Notably, the emergence of ChatGPT [92] highlights the potential of foundation models to understand human intentions, automate intricate processes, and generate natural responses; the advent of GPT-4 [93] offers immense potential for multi-modal perception, which is essential to the real-world grounding ability.

Therefore, foundation models enable AI to harness tools, which can lead to more potent and streamlined solutions for real-world tasks. Foundation models are able to decipher complex data, simulate human-like planning capabilities, and generate a broad spectrum of outputs. Concurrently, specialized tools can be employed to refine and target specific goals. The amalgamation of tools and models unveils vast potential where sophisticated procedures can be automated with limited human involvement. This paradigm, dubbed as **tool learning with foundation models** in this article (Figure 1), aims to combine the strengths of specialized tools and foundation models, thereby culminating in greater accuracy, efficiency, and autonomy in problem-solving. Recent research has shed light on foundation models' potential to exhibit a level of dexterity and finesse in tool use [1, 16, 27, 46, 62, 89, 117, 136, 155, 163, 164]. Despite these breakthroughs, the efforts mainly focus on applying foundation models to specific tasks and domains with delicate algorithm designs. The current understanding of tool learning is still not comprehensive enough to estimate its characteristics and future developments. We believe that it is crucial to examine the current progress of tool learning to explore their potential and challenges and to better pave the way for future technological advancements.

In this article, we conduct a systematic investigation and comprehensive review of tool learning, attempting to build a full grasp of the key challenges, opportunities, and directions in this field.

Before delving into the tool learning framework, we introduce essential **backgrounds** (Section 2), covering both tools and foundation models and their interaction. Specifically, we first recapitulate the cognitive origins of tool use in human history and its potential implications for tool use in AI systems (Section 2.1), followed by a categorization of tools from the perspective of the user interface (Section 2.2). Then we review the AI paradigm shift brought by foundation models and highlight the emergence and significance of tool learning (Section 2.3). After that, we discuss the complementary roles of tools and foundation models, and argue that integrating both can bring various advantages, such as improving interpretability, enhancing robustness, and delivering better user experiences (Section 2.4).

Next, we present a comprehensive literature review for existing exploration in tool learning. While previous works often focus on specific aspects in isolation, we strive to formulate a general tool learning **framework** (Section 3.1), which comprises the controller (typically modeled using a foundation model), tool set, environment, perceiver, and human. Based on the unified framework, we review existing works of tool learning, highlight core research problems, and introduce their existing solutions as well as future explorations. The whole procedure (Section 3.2) of tool learning starts with a user instruction, and models are required to make an executable plan for tool execution. To bridge user instructions with appropriate tools, models should first learn to understand the user intents underlying the instruction (i.e., *intent understanding*) and understand the functionalities and usage of tools (i.e., *tool understanding*). Models should also learn to decompose a complex task into several subtasks, dynamically adjust their plan through reasoning, and effectively conquer each sub-task with the appropriate tools. Regarding the training strategy (Section 3.3) to facilitate models for improved tool utilization, we conclude with two mainstream methods: learning from demonstrations and learning from feedback. We discuss how to construct effective training supervision under different settings. To facilitate transferring the learned tool-use skills to new tools and situations, i.e., generalizable tool learning, it is important to design a unified interface that enables the model to interact with different tools in a standardized manner.

Finally, we discuss the remaining important **research topics** (Section 4) for applying our general framework to real-world scenarios, including (1) safety and trustworthiness, where we emphasize the potential risks from adversaries, governance, and trustworthiness. We contend that careful considerations are necessary before deploying tool learning models in high-stakes scenarios (Section 4.1); (2) tool creation, where we discuss the possibility that AI can also create new tools, challenging the long-held beliefs about what makes humans unique (Section 4.2); (3) personalized tool learning, where models provide tailored assistance to users in tool use. We highlight the challenges of aligning user preference with tool manipulation and introduce the shift from reactive to proactive systems, and the privacy-preserving concerns (Section 4.3); (4) knowledge conflicts in tool augmentation, where we review how tools can be leveraged to enhance models' generation and the practical problems of knowledge conflicts, which can lead to inaccurate and unreliable model predictions (Section 4.4); (5) other open problems, such as viewing tool use capability as a measure for machine intelligence and tool learning for scientific discovery (Section 4.5). Overall, we hope this article could inspire further research in integrating tools with foundation models and developing more intelligent and capable AI systems.

2 Background

In this section, we first discuss the cognitive origins of human tool use (Section 2.1), followed by a tool categorization through the lens of the user interface (Section 2.2). Then we review the recent AI paradigm shift brought by foundation models (Section 2.3) and its significance in tool learning. After that, we examine the respective roles of specialized tools and foundation models in problem-solving, and discuss the benefits and challenges of their integration (Section 2.4).

2.1 Cognitive Origins of Tool Use

Tools are commonly viewed as extensions of human beings. Tool use is defined as a unique characteristic of human beings that is distinguished from other species [142]. Throughout evolution, the ability to use tools has been essential for animals, particularly those with advanced intellectual development [122]. For example, chimpanzees have been observed using stones or other materials to crack nuts [11], while New Caledonian crows can craft and utilize two distinct types of hook tools to aid in capturing prey [47]. However, human tool behavior diverges from these observations in several ways. Humans can create much more complicated tools than other animals, such as

converting our actions into fundamentally different mechanical actions in tools like hammers and pencils [35]. Additionally, we can harness natural forces such as wind turbines to create tools. This ability may be attributed to our deep comprehension of cause-and-effect relations, which allows us to engage in technical reasoning [98].

Neural Basis of Tool Use. To better understand human tool use behaviors, researchers analyze the neural basis of tool observation and execution. It is verified that humans have parietal systems involved in grasping objects and using tools, and the anterior supramarginal gyrus activation of observing tool use is typical of human subjects, of which macaques do not exhibit [94]. This neurocognitive bases of tool observation may be related to the origins of cumulative technological evolution (e.g., the improvement in the efficiency and complexity of human tools and techniques over generations [98]). While for the tool execution, researchers hold different views on manipulation-based versus reasoning-based approaches [95]. The former claims that tool use has to be supported by the simulation of sensorimotor experiences, and the latter demonstrates the importance of reasoning based on mechanical knowledge in tool use. Nevertheless, the overall trend in cognitive science is understanding cognition as an enactive process that emphasizes interaction with the external world [33], and the feedback from observation, communication, and hands-on practice is important for mastering tool use.

Three Intelligence Levels of Tool Use. Besides, there are specific frameworks designed to discuss the level of intelligence represented by human tool use. For instance, "intoolligence" [96] divides the tool use behavior into three modes: *assistive tool use* is usually passive and unaware (e.g., walking in the rain shelter corridor); *arbitrary tool use* requires active interaction (e.g., driving, using smart phones); *free tool use* further needs to comprehend and choose appropriate tools for the scenarios (e.g., cooking new dishes). The three modes of tool use present a progressive relationship, and the key cognitive process for achieving free tool use is technical reasoning, which allows someone to learn new actions by observing others using, selecting, or making a tool instead of numerous practices.

Transition from Physical Tools to Conceptual Tools. Apart from tools in the physical world, we can also turn to more abstract tools. Take cognitive tools [44] as an example: it refers to an auxiliary aid that facilitates higher-order thinking (e.g., multi-step critical analysis, the generation of creative problem-solving solutions). Cognitive tools can be classified based on the functionalities they provide [60]. These include (1) supporting cognitive processes (e.g., documenting intermediate reasoning outcomes), (2) alleviating lower-level cognitive load to free up resources for advanced-level thinking, (3) enabling learners to engage in activities out of their reach and (4) allowing learners to generate and test hypotheses (e.g., simulated diagnoses for medical students).

2.2 Tool Categorization: A User-Interface Perspective

The growing number and complexity of tools in our world make it increasingly important to understand and group them in a meaningful way. Here we introduce a taxonomy that sorts these tools based on their modes of expression and interaction. As depicted in Figure 2, this taxonomy incorporates three levels of interaction, arranged from the most tangible to the least. *The physical level* involves direct physical interactions with tools. *The* ***graphical user interface*** **(GUI)** level facilitates user interaction with visual representations of tools. *The program level* involves users engaging directly with the underlying source code of tools.

Physical Interaction-based Tools. We start with the most tangible genre of tools, physical interaction-based tools. This class of tools involves direct interactions with the physical world, including devices like robots, sensors, and wearables that could physically impact the environment. Physical interaction tools have the capability to sense and respond to the physical environment of users, making them useful in a wide range of applications, from manufacturing to healthcare

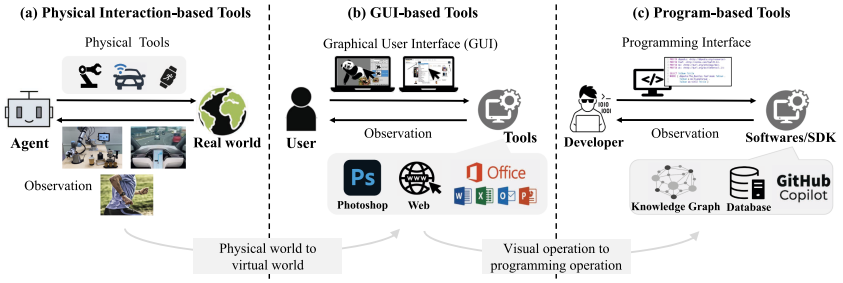


Fig. 2. Tool categorization from the perspective of the user interface.

and education. Physical interaction tools are close to the real world, and they have the potential to substantially improve efficiency and productivity. For example, robots can perform from simple to intricate, even adventurous tasks to reduce human errors and labor costs. Sensors can collect valuable data, such as temperature and pressure, allowing for real-time monitoring and optimization of industrial processes.

GUI-based Tools. Some tools allow users to manipulate them through an interactive interface, i.e., visual representations of tools, with pre-defined operations. These tools, defined as GUI-based tools, do not have a direct impact on the physical world. The GUI interface typically includes buttons, menus, text boxes, and other graphical elements that allow users to interact with the underlying system. These tools are extensively employed in various industries and applications such as software development, data analysis, and design. Particularly, GUI-based tools can improve productivity by streamlining workflows and automating repetitive tasks. GUI-based tools could considerably simplify complex tasks and reduce the learning curve for non-technical users. From this viewpoint, tool learning with foundation models share the same primary goal, which simplifies intricate tasks to a natural language format. Representative GUI-based tools are usually well-developed software such as browsers, Microsoft Office, Adobe PhotoShop, and so on. These applications showcase the versatility that graphical interfaces can provide and enable users to access and manipulate complex features within the software. On the other hand, the main limitation of GUI-based tools is that they may not provide the flexibility and customizability of command-line interfaces or APIs.

Program-based Tools. The innermost layer of tools that users can access is the source code, offering a high degree of flexibility for the input and output of these program-based tools. Program-based tools are software tools primarily designed for use through programming interfaces rather than visual interfaces. They can take various forms, including declarative languages, programming libraries, **software development kits (SDKs)**, and even neural network-based tools. These tools are typically used by developers or technical users who possess a deeper understanding of the underlying data, system or technology, with which the users could complete complex software applications. The main advantage of program-based tools is that they provide greater flexibility and customizability than GUI-based tools, and users can build more sophisticated solutions for current problems. As a result, such tools require a greater degree of technical expertise and programming knowledge, which may not be accessible to non-technical users. For example, program-based tools can be more time-consuming to set up and configure and may require more maintenance and support in the learning process. It is noteworthy that, although these tools pose difficulties for human beings in terms of the learning curve, they may not have the same level of challenges for foundation models.

The above three interaction modes have varying levels of connectivity with the tool kernel. They are not strictly mutually exclusive but indicate a tendency to intermingle with each other.

2.3 Paradigm Shift of Foundation Models

In recent years, the field of NLP has undergone a paradigm shift, marked by the advent of **pre-trained language models** (**PLMs**) [13, 31, 40]. Previously, NLP was a challenging field that necessitated designing separate learning objectives for distinct research domains, such as dependency parsing [58], named entity recognition [88], and summarization [90]. Despite the successful design of effective models and methods for these specific tasks, the separated nature of this paradigm impeded progress toward a holistic comprehension of language, thereby limiting its potential.

The invention of PLMs changes this paradigm. Building on Transformers [140], PLMs are trained on massive corpora, from which general linguistic ability and world knowledge are learned. This technique has expedited the unification of NLP tasks, giving rise to the *pre-train-then-fine-tune* paradigm, which has achieved new state-of-the-art performance on several NLP benchmarks, such as GLUE [146] and SuperGLUE [145]. At this stage, each task shares the same starting point and only diverges as the task-specific adaptation proceeds. The fusion of task paradigms is still ongoing. T5 [110] transforms all NLP tasks into a text-to-text format with textual descriptions, while GPT-3 [15] has discovered that introducing appropriate textual prompts can yield the desired output for specific tasks. Prompts can stimulate the knowledge learned by PLMs during pre-training. Research even suggests that with appropriate prompt guidance, models can perform complex reasoning tasks [148, 153]. Also, prompts formulated in a natural language format possess remarkable generalization capabilities. Specifically, models that have undergone fine-tuning using diverse instructions are able to effectively generalize to new, unseen data [115, 151]. Overall, prompts demonstrate a proof-of-concept that uses PLMs as the underlying infrastructure and natural language as the medium to uniformly perform various tasks.

Nevertheless, there exist numerous tasks that transcend the scope of purely natural language. For instance, generating presentation slides,[1] constructing 3D models via CAD applications, and scheduling meetings through the analysis of team member calendars are examples of complex tasks that have not been defined in traditional AI. Fortunately, the strong generalization ability of PLM enables us to use natural language as a medium to accomplish these tasks by manipulating tools [169]. Essentially, the key to tool learning is to decompose complex tasks into sub-actions, tokenize actions in the form of natural language and convert them into executable instructions that can be understood by specific tools. Language models serve as "translators", making complex tasks more accessible to individuals without specialized technical knowledge. By enabling machines to understand and interact with human language in a more natural way, we can unlock new possibilities for collaboration and problem-solving that were previously impossible.

Superiority of Foundation Models over Traditional Approaches. The emergence of foundation models has opened up exciting opportunities for tool learning that were difficult to achieve with traditional **machine learning** (**ML**) approaches.

As discussed in Section 3.2.1, foundation models' strong language understanding capabilities enable them to effectively interpret user intents expressed in natural language and map them to relevant tool executions. In contrast, traditional ML models often struggle with such open-ended language understanding tasks. Recent works have also demonstrated the emerging reasoning and planning capabilities of foundation models, allowing them to decompose complex tasks, generate multi-step plans, and adapt to intermediate results (Section 3.2.2). Such reasoning and planning skills are challenging to achieve with traditional ML models, which excel more at pattern recognition within narrow domains. Furthermore, foundation models exhibit remarkable

[1] https://www.microsoft.com/en-us/microsoft-365

few-shot and zero-shot learning abilities, enabling them to learn to use new tools with minimal or no examples. This is particularly valuable in tool learning, where collecting large annotated datasets for every tool is infeasible (Section 3.3.3). Traditional ML models, in contrast, heavily rely on abundant task-specific labeled data. Lastly, foundation models can be easily composed with different tools through a unified interface (e.g., natural language or programming languages). This modularity allows for flexible integration of new tools without requiring significant architectural changes (Section 3.3.3). Traditional ML models often have fixed input/output formats, making it difficult to seamlessly incorporate new tools.

In addition, traditional non-ML approaches cannot handle tool learning well. Statistical methods, for example, often rely on assumptions about data distributions and linear relationships that do not align well with the dynamic and interactive nature of tool learning. They struggle to capture the hierarchical structure and contextual reasoning involved in tool use. Similarly, rule-based systems face challenges in modeling the open-ended and contextual nature of tool use. Defining comprehensive rules for all possible scenarios is infeasible, and such systems often fail to adapt to novel situations or handle the ambiguity inherent in real-world tool interactions.

2.4 Complementary Roles of Tools and Foundation Models

The integration of specialized tools and foundation models represents a promising approach for harnessing the unique strengths of both. Specifically:

Benefits of Tools. Tools that are designed to streamline concrete and specific objectives bring several benefits for tool learning: (1) **Mitigation for Memorization.** Although foundation models have demonstrated an exceptional ability to memorize [18–20], they are not capable of memorizing every piece of training data. Furthermore, foundation models are often prompted with a relatively short context during model generation, thus not all the memorized knowledge can be properly steered [82]. Additionally, memorization alone does not support the real-time coverage of up-to-date knowledge. Besides, foundation models are also criticized to hallucinate knowledge [113, 123] by generating seemingly plausible but non-factual content. This is unacceptable in applications like financial transactions that require the results are 100% correct. Given the above factors, it is necessary to augment foundation models with real-time tool execution to mitigate limitations in memorization. (2) **Enhanced Expertise.** Specialized tools are designed to cater to specific domains with functionalities that are not available in foundation models. As a result, they are better suited to address the needs of domain-specific tasks, such as Wolfram[2] for scientific calculation, through the utilization of tailored algorithms. Instead of solely relying on the foundation model to accomplish the task, models could invoke appropriate tools to generalize to a wider range of tasks that are beyond their capabilities. (3) **Better Interpretability.** Foundation models are criticized for lacking transparency in their decision-making process [69], which can be a significant concern in applications such as healthcare or finance, where interpretability is critical for making informed decisions. In contrast, the process of tool execution reflects the whole process of how models solve complex requests, which allows for better interpretability and transparency. Users can easily understand why certain tools are called and how they contribute to the final output, which can improve trust and facilitate human-machine collaboration. (4) **Improved Robustness.** Foundation models are susceptible to adversarial attacks [52, 143], where slight modifications to the input can flip the model prediction. This is because these models heavily rely on statistical patterns in the training data. Conversely, tools are designed specifically for their intended use cases, which may be agnostic to the input perturbation. This makes tools more resistant to adversarial attacks.

[2]https://www.wolframalpha.com/

Benefits of Foundation Models. Foundation models can provide a solid basis for understanding, planning, reasoning, and generation, which bring several benefits for tool learning as follows: (1) **Improved Decision-Making and Reasoning Abilities.** Foundation models are trained on vast amounts of data, enabling them to acquire world knowledge across a wide range of domains. If properly steered, such knowledge can be wielded to perform decision-making and planning over prolonged time horizons [45]. Besides, foundation models have demonstrated remarkable reasoning abilities [148, 153], thereby enabling them to extrapolate the consequences of actions and make judicious decisions. These reasoning abilities are particularly useful for tasks requiring a deep understanding of cause-and-effect relations (Section 3.2.2). (2) **Better User Experience.** Benefitting from the powerful intent understanding capability of foundation models, tool learning could revolutionize the way we interact with machines and liberate users from the cognition load, allowing them to engage in higher-order thinking and decision-making processes. This, in turn, fosters a seamless and more natural language-based interaction paradigm that revolutionizes traditional GUIs. The user only needs to provide high-level guidance and direction, and the model will seamlessly comprehend the user's intent, thereby delivering more personalized and precise responses. In addition, tool learning has the potential to democratize access to complex tools. With the aid of foundation models, even novice users can easily and quickly get started with a new tool, regardless of their prior experience or technical expertise. This not only reduces the barriers to entry for new users but also unlocks a wealth of possibilities for innovation and creativity.

3 Tool Learning

To provide a comprehensive understanding of tool learning, we first present a general framework, which encompasses four fundamental components, namely tool set, environment, controller, and perceiver, as detailed in Section 3.1. Subsequently, we discuss the general procedure of tool learning in Section 3.2. Lastly, we delve into the training methods for tool learning and discuss how to achieve generalizable tool learning in Section 3.3. Considering the lack of a systematic tool learning evaluation in prior works, based on our framework, we conduct experiments on 18 representative tools with case studies and demonstrate that state-of-the-art foundation models (e.g., ChatGPT) can effectively use tools to solve tasks with simple prompting, highlighting the potential of tool learning. See the supplementary material for details.

3.1 Components of Tool Learning

While research in tool learning has achieved remarkable advancements, these efforts mainly focus on specific tasks or domains with delicate approach designs. In addition, existing surveys relevant to tool learning either focus on augmenting models with tool execution [82] or leveraging models for decision making [162], without investigating both streams under the same unified framework. This may hinder a comprehensive understanding of the core challenges and future directions in tool learning. To this end, we provide a general framework to better organize and understand existing works in tool learning. Specifically, we frame tool learning with four components, as shown in Figure 3. Each component has its own characteristics and functions (Section 3.1.1), but they also interact with each other closely (Section 3.1.2).

3.1.1 Understanding the Components. We first introduce each component of the tool learning process.

Tool Set. Serving as the fundamental ingredient of tool learning, the tool set $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots\}$ contains a collection of different tools that have different functionalities. As we have elaborated in Section 2.2, a tool in $\mathcal{T}$ can have different interfaces. In the following, we mainly take **Application Programming Interface** (**API**) as the example to illustrate how to interact with

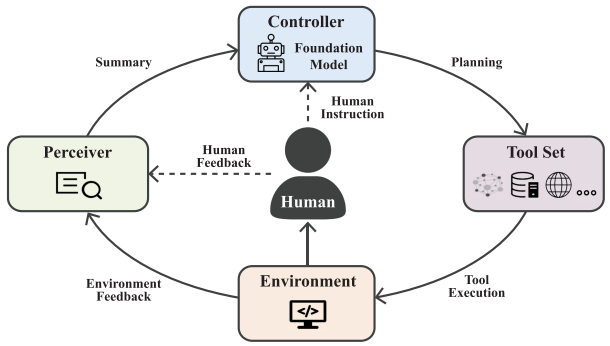


Fig. 3. The tool learning framework. The user sends an instruction to the controller, which then makes decisions and executes tools in the environment. The perceiver receives feedback from the environment and the user and summarizes them to the controller.

tools. We define an API as any function that can take the output of the foundation model as its input.

Environment. The environment $\mathcal{E}$ is the world where the tools operate, which provides the *perceiver* with the execution results of tools. It provides the infrastructure necessary for tool execution, which can be either virtual or real. The former refers to a simulated environment that allows the model to interact with a digital representation of the tool, while a real environment involves actual interaction with the physical tool. Virtual environments have the advantage of being easily accessible and replicable, allowing for more cost-effective training for models. However, virtual environments may not fully replicate the complexities of the real-world environment, leading to overfitting and poor generalization [41]. On the other hand, real environments provide a more realistic context but may be more challenging to access and involve greater costs.

Controller. The controller C serves as the "brain" for tool learning framework and is typically modeled using a foundation model. The purpose of the controller C is to provide a feasible and precise plan for using tools to fulfill the user's request. To this end, C should understand user intent as well as the relationship between the intent and available tools, and then develop a plan to select the appropriate tools for tackling tasks, which will be discussed in Section 3.2.1. In cases where the query is complex and targets a high-level task, C may need to decompose the task into multiple sub-tasks, which requires foundational models to have powerful planning and reasoning capabilities (Section 3.2.2).

Perceiver. The perceiver $\mathcal{P}$ is responsible for processing the user's and the environment's feedback and generating a summary for the controller. Simple forms of feedback processing include concatenating the user and environment feedback or formatting the feedback using a pre-defined template. The summarized feedback is then passed to the controller to assist its decision-making. By observing this feedback, the controller can determine whether the generated plan is effective and whether there are anomalies during the execution that need to be addressed. Under more complex scenarios, the perceiver should be able to support multiple modalities, such as text, vision, and audio, to capture the diverse nature of feedback from the user and the environment.

3.1.2 Connecting the Components. Formally, assume we have a tool set $\mathcal{T}$, which the controller can utilize to accomplish certain tasks. At time step t, environment $\mathcal{E}$ provides feedback e_t on the tool execution. The perceiver $\mathcal{P}$ receives the user feedback f_t and the environment feedback e_t, and generates summarized feedback x_t. Typically, the perceiver can be achieved by pre-defined rules (e.g., concatenating f_t and e_t) to form x_t, or modeled with complex neural models. The controller

C generates a plan a_t, which selects and executes an appropriate tool from $\mathcal{T}$. This process can be formulated as the following probability distribution:

$$p_C(a_t) = p_{\theta_C}(a_t \mid x_t, \mathcal{H}_t, q), \tag{1}$$

where θ_C denotes the parameters of C, q denotes the user query or instruction, and $\mathcal{H}_t = \{(x_s, a_s)\}_{s=0}^{t-1}$ denotes the history feedback and plans. In its simplest form, a generated plan a_t can simply be a specific action for tool execution. C can also synergize its reasoning process with the action prediction, where a_t may additionally contain the reasoning traces that explain which sub-task should be solved next and which tool to choose for solving the sub-task. It is worth noting that if the dependence on x_s is removed from Equation (1), the resulting probability distribution becomes equivalent to autoregressive language modeling. From this perspective, the controller additionally grounds the foundation model to the environment and the tool set. Moreover, we can factorize Equation (1) as follows:

$$p_{\theta_C}(a_t \mid x_t, \mathcal{H}_t, q) = \sum_{\mathcal{T}_i \in \mathcal{T}} p_{\theta_C}(a_t \mid \mathcal{T}_i, x_t, \mathcal{H}_t, q) \times p_{\theta_C}(\mathcal{T}_i \mid x_t, \mathcal{H}_t, q). \tag{2}$$

The decomposition reveals that the construction of the plan a_t involves two subtasks: selecting the appropriate tool based on the user intent and deciding the actions to execute using the selected tool. For instance, given an instruction such as "I want to book a flight to Beijing next week", the controller C first infers that the user's goal is to reserve a flight, with Beijing as the destination and the next week as the travel time. The model then selects the airline reservation system as the tool. Finally, it inputs the time and destination as the preliminary plan. In the process of making a reservation, we may face unexpected situations such as the unavailability of flights to Beijing in the next week. To cope with these anomalies, we can further equip C with the ability to reason about the current context and generate alternative plans, as we will discuss in detail in Section 3.2.2.

After a plan a_t is generated, it will be executed in $\mathcal{E}$, and the resulting feedback e_{t+1} from $\mathcal{E}$ will be passed on to the perceiver. The above process repeats for multiple rounds until the controller accomplishes the task. The overall objective is to find an action sequence $\{a_t\}$ that ultimately fulfills the task specified by the user instruction q. Note after tool execution, the controller may additionally integrate the execution results into a plausible response for the user.

3.1.3 Comparison with Traditional ML Tasks. Tool learning differs from traditional ML tasks in several key aspects. First, tool learning involves understanding and interacting with real-world tools, such as APIs, software applications, and databases. These tools have complex functionalities, structured interfaces, and domain-specific requirements. Models need to learn to map between natural language and tool-specific actions, handle input/output formats, and integrate tool responses into their decision making. In contrast, traditional ML tasks often operate on self-contained datasets without the need for real-world tool grounding. Second, tool learning requires open-ended reasoning and planning. The user specifies their goals using natural language, which can be highly variable and ambiguous. The model needs to understand the user's intent, break it down into sub-goals, and generate multi-step action sequences to accomplish the task. Classical ML models are more focused on pattern recognition within narrow domains and lack the necessary reasoning and planning capabilities for open-ended tool use. Third, tool learning systems need to engage in interactive exchanges with users and the environment. They should support back-and-forth dialogues to clarify user intents, provide intermediate results, and gather feedback. They also need to execute tool actions, observe their effects, and incorporate the results into subsequent reasoning steps. This interactive nature differs from classical ML setups that learn from static datasets.

3.2 The General Procedure: From Intent to Plan

As formulated in Section 3.1.2, the general procedure of tool learning necessitates intricate interplay among different components. In this section, we will further elaborate on the key issues involved in this procedure.

3.2.1 Understanding Intent and Tools. To accurately fulfill the task specified by the user query q, the controller needs to understand two aspects: (1) the underlying intent of the user, which involves formalizing the query q as a high-level task (i.e., *intent understanding*); (2) the tool set $\mathcal{T}$, which entails comprehending the functionality and objective of each tool (i.e., *tool understanding*). By understanding both aspects, the controller can bridge the gap between the user's intent and the tool set, which is the pre-requisite for connecting controller C, the user, and tool set $\mathcal{T}$ in Figure 3.

Intent Understanding. Understanding user intent is a long-standing research topic in NLP [51, 132]. By accurately identifying the user intent, the controller can provide more personalized responses with a better user experience. Recent explorations in instruction tuning [151] demonstrate that foundation models can possess extraordinary proficiency in comprehending user instructions. Prior work has shown that fine-tuning large language models on a collection of datasets templated with human instructions allows models to generalize even to instructions for unseen tasks [85, 99, 115, 151]. Promisingly, such generalization ability can further be enhanced by scaling up both the model size and the quantity or diversity of training instructions [49].

Despite the impressive intent understanding capabilities, challenges still exist in real-world tool learning scenarios: (1) **Understanding Vague Instructions.** Many user queries are inherently imprecise and can even be polysemous, requiring the controller to rely on contextual cues and background knowledge to infer the user's intended meaning. One possible solution is to actively interact with users to clarify any ambiguity, such as asking for clarifications about a previous user query. (2) **Generalization to Diverse Instructions.** As the intent space is theoretically infinite, it is almost impractical for foundation models to be exposed to every real-world intention during training. In addition, the challenge of personalization arises from the fact that each individual has their own unique way of expressing intentions, which requires the model to adapt to the diverse expressions of intent of different individuals. One solution is to leverage user feedback and actively adapt the model to individual users, i.e., personalized tool learning (Section 4.3).

Tool Understanding. As noted by Hernik and Csibra [43], when learning to utilize a specific tool, human beings perceive it as an object with particular functions, engaging in a cognitive process to understand its purpose and operation. By observing goal-directed demonstrations and following actions performed by other people, they gradually acquire the necessary knowledge and skills to use the tools effectively. Similarly, this understanding process is crucial for successfully solving tasks with tools. Analogously, a comprehensive understanding of the tools' functionalities is indispensable for enabling the controller to use tools proficiently.

In real-world scenarios, tools are typically accompanied by a manual (or tutorial), which provides sufficient relevant details about their functionalities and usage. Endowed with strong few-shot learning [15] and zero-shot learning [151] capabilities, foundation models can be prompted to unravel tools' functionalities and comprehend how to use them. To this end, it is feasible to construct suitable task-specific prompts either through manual design [141] or retrieval [174]. These prompts should describe the API functionalities or exemplify with demonstrations of their usage.

We categorize two prompting approaches as shown in Figure 4: (1) **zero-shot prompting**, which describes API functionalities, their input/output formats, possible parameters, and so on. This approach allows the model to understand the tasks that each API can tackle; (2) **few-shot prompting**, which provides concrete tool-use demonstrations to the model. By mimicking human behaviors from these demonstrations, the model can learn how to utilize these tools. We

Zero-shot Prompting: Here we provide a tool (API) "forecast_weather(city:str, N:int)", which could forecast the weather about a city on a specific date (after N days from today). The returned information covers "temperature", "wind", and "precipitation".
Please write codes using this tool to answer the following question: "What's the average temperature in Beijing next week?"

Few-shot Prompting: We provide some examples for using a tool. Here is a tool for you to answer question:
Question: "What's the temperature in Shanghai tomorrow?"

```
return forecast_weather("Shanghai", 1)["temperature"]
```

Question: "Will it rain in London in next two days?"

```
for i in range(2):
    if forecast_weather("London", i+1)["precipitation"] > 0:
        return True
return False
```

Question: "What's the average temperature in San Francisco next week?"

Fig. 4. Examples of zero-shot and few-shot prompting for tool understanding. The prompts are constructed by describing the functionalities (zero-shot prompting) or giving usage examples (few-shot prompting) of a weather API.

provide experimental results of both prompting methods in the supplementary material. Overall, prompting has been widely adopted as a lightweight approach to teach foundation models about tools [32, 93, 164] with minimum human effort. Prompts can be easily adjusted to accommodate changes of tools. For instance, when tools are modified or upgraded, we can flexibly rewrite the prompts to adapt the model behaviors.

Despite these advantages, prompting methods still face several challenges. First, since the effectiveness of prompting depends a lot on the model, smaller or less capable models cannot understand prompts well. Second, prompting is restricted by input context length. Although foundation models have been shown to learn to use simple tools through prompts, the situation may be more challenging with multiple complex tools with long descriptions. A potential solution is to add an intermediate stage of tool selection, which first retrieves a small set of tools that are most suitable for the task at hand.

3.2.2 Planning with Reasoning. In practice, the user query q often implies a complex task that should be divided into multiple sub-tasks with proper sequencing, thereby necessitating a process of *reasoning*. Recent research has revealed that reasoning capabilities can emerge when foundation models are scaled up to a certain size [152]. In particular, foundation models with tens or hundreds of billions of parameters can generate intermediate reasoning traces during complex problem-solving, thereby significantly enhancing their zero-shot and few-shot performances [89, 91, 152]. However, the vanilla few-shot prompt learning [15], whereby models are provided with a prompt consisting of several examples for the given task, has been shown to fail when it comes to problems that require complex reasoning [29]. To better elicit reasoning in foundation models, Wei et al. [153] propose **Chain-of-Thought** (**CoT**) prompting. Unlike vanilla few-shot prompt learning, CoT additionally inserts the reasoning trace required to derive the final answer for each example in the prompt. In this way, CoT prompts models to generate their "thoughts" on the necessary intermediate steps before arriving at the final answer. CoT has been proven to significantly boost performance on a wide range of tasks, including arithmetic reasoning, commonsense reasoning, and symbolic reasoning [153].

In light of the remarkable reasoning abilities of foundation models, recent research has made successful attempts to employ them in the controller in tool learning. It is demonstrated that their reasoning capabilities enable the controller to effectively decompose a complex problem into several sub-problems, and determine which tool to call upon for each sub-problem. We categorize relevant research into two streams: *introspective reasoning* and *extrospective reasoning*. The former involves generating a static plan of tool use without interacting with the environment $\mathcal{E}$, while

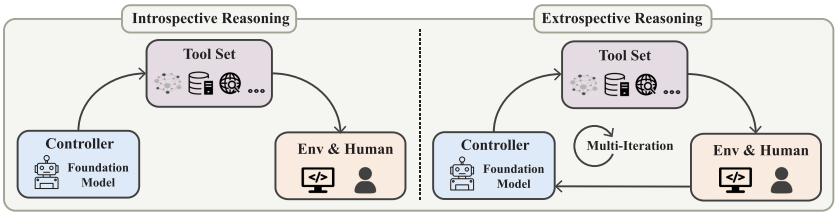


Fig. 5. Illustration of introspective reasoning and extrospective reasoning. Extrospective reasoning requires feedback from the environment and humans to carry out iterative plan generation. We omit the perceiver in the illustration for simplicity.

the latter generates plans incrementally by iteratively interacting with $\mathcal{E}$ and utilizing feedback obtained from previous executions. As shown in Figure 5, the environment $\mathcal{E}$ is invisible to the controller C in introspective reasoning but is visible in extrospective reasoning, creating a closed-loop interaction among the four components.

Introspective Reasoning. This kind of reasoning directly generates multi-step plans for tool use without knowing intermediate execution results. For instance, Huang et al. [45], Qian et al. [106] investigate the planning ability of foundation models and show that they are capable of decomposing high-level tasks into semantically plausible sub-plans. Another representative work is **Program-Aided Language Models (PAL)** [37], which prompts models to generate Python codes for intermediate reasoning steps. PAL uses the Python program interpreter as the tool, enabling the model to act as a programmer writing detailed comments, and achieving significant improvements in arithmetic, symbolic, and algorithmic reasoning. Notably, the idea of model-as-programmer has also been shown to be successful in embodied agents, as evidenced by ProgPrompt [126] and Code-as-Policies [66], which prompt models to generate executable programs for embodied agents. These studies reveal that, despite not having direct interaction with the environment, models are capable of generating executable programs for agents and anticipating possible anomalies in the plan execution. Another example is Visual ChatGPT [155], which interleaves various vision foundation models with ChatGPT to enable understanding and generating images. In their system, ChatGPT serves as the core controller and makes sequential decisions. At each step, ChatGPT might call a vision model to modify an existing image or respond to the user with plain text. However, since these models are not grounded in the environment, they may generate unrealistic and nonsensical plans. To this end, SayCan [1] emphasizes those actions that the agent is "permitted" to execute instead of those it is "willing" to perform. In practice, they employ a value function to estimate the probability of each action being successfully executed, making the agent more physically grounded in the environment.

Extrospective Reasoning. Despite its simplicity, introspective reasoning cannot adapt the plan in response to intermediate execution results. A more rational approach to planning is taking the environment $\mathcal{E}$ into account, and generating plans incrementally (e.g., one step at a time) with subsequent plans dependent on previous execution results. This allows the four components described in Section 3.1 to be well integrated and to cooperate effectively to achieve complex tasks. We refer to such an incremental reasoning strategy as extrospective reasoning. Compared to introspective reasoning, extrospective reasoning additionally considers feedback from the user and environment (Figure 5), and is thus better suited to complex tasks [76, 149], such as multi-step QA and embodied learning, where decision-making at each step is dependent on the preceding context. Recent works such as Self-Ask [105], ReAct [164], and ToolFormer [117] have demonstrated that by providing access to search engine APIs, models are able to achieve improved accuracy on multi-step QA. Through CoT prompting (Self-Ask and ReAct) or fine-tuning (ToolFormer), models can

learn to decompose complex questions and utilize the search API to find the answer to the first sub-question. Based on the response and the question, they can then iteratively determine the subsequent question to ask or give the final answer.

For embodied learning, while introspective reasoning methods have demonstrated the ability to generate executable programs and address potential execution anomalies, direct interaction with the environment can further enhance models' planning capabilities. For instance, Inner Monologue [46] leverages multiple sources of feedback from the environment, such as whether a task is completed successfully and the current scene information. In this way, models can generate more feasible plans and improve their ability of high-level instruction completion. LLM-Planner [127] explicitly considers anomalies that may arise during plan execution and utilizes environmental feedback to regenerate the plan in case of execution failure, enabling models to handle exceptions appropriately. Additionally, ReAct [164] grants models the autonomy to determine when to cease generating action tokens during planning, enabling them to reason about the current situation and develop more refined subsequent plans.

Extension to the Multi-Step Multi-Tool Scenario. Humans do not stick to only one single tool to complete complex tasks. Instead, we carefully decompose the task into several sub-tasks, select the most suitable tool for each sub-task, and gradually accomplish them step by step. Although most of the existing research is limited to either multi-step single-tool or single-step multi-tool scenarios, there is a recent surge of interest in the multi-step multi-tool scenario. For instance, Auto-GPT[3] demonstrates the huge potential of GPT-4 in manipulating multiple tools and making long-term plans, pushing the boundaries of what is possible with tool learning. Given a user query, Auto-GPT will take step-by-step actions to accomplish the objective autonomously. Although Auto-GPT constitutes a significant step in the multi-step multi-tool scenario, there are still several challenges and future directions that need to be investigated.

- **Understanding the Interplay among Different Tools.** The multi-step multi-tool scenario typically involves complex tasks that require a higher level of intent understanding and reasoning capability. To effectively utilize multiple tools under this scenario, models need to grasp not only the individual functionalities of tools but also their interactions and dependencies. Models should be able to sequence the tools in a logical order so that the subsequent tools can leverage the information generated by the previous tools and effectively complete the task.
- **Enhancing Reasoning through Formalisms and Mathematical Integration.** LLM-based agents inherently comprehend and generate language, therefore, predominant research utilizes plain natural text to facilitate agents' reasoning and planning. Nonetheless, emerging research indicates that incorporating external formalisms, such as mathematical tools and non-natural language forms, significantly enhances agents' performance in complex reasoning tasks. For instance, Xu et al. [158] reveal that employing a **probabilistic graph model** (**PGM**) to represent the historical dynamics of multi-agent reasoning, and subsequently integrating PGM with an agent, substantially improves the agent's decision-making capabilities. Ye et al. [165] propose **Agentic Process Automation** (**APA**), which effectively integrates intelligent agents into the conventional **Robotic Process Automation** (**RPA**), enhancing the system's intelligence while remaining controllable.
- **From Sequential Execution to Parallel Execution.** Tool executions do not have to be performed sequentially. In certain scenarios, parallel execution is possible for sub-tasks that do not depend on each other, which can potentially improve execution efficiency. For

[3] https://github.com/Torantulino/Auto-GPT

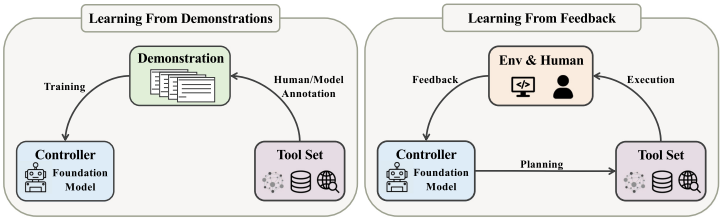


Fig. 6. Training strategies for tool learning: (left) learning from human-annotated or model-annotated demonstrations; (right) learning from feedback, where the supervision could come from either the environment or humans.

instance, given a user instruction *"Generate two codes, one for drawing a rectangle, and one for drawing a circle."*, the two tasks can be assigned to two agents, enabling the codes to be generated simultaneously. Determining the dependencies among different sub-tasks and effectively switching between parallel and sequential execution is a promising direction that merits further investigation.

— **From Single-agent Problem-Solving to Multi-agent Collaboration.** Previous works typically assume that a single agent (controller) is solely responsible for the entire tool learning procedure. However, in practice, complex tasks often demand collaboration among multiple agents, each possessing unique abilities and expertise. Embracing multi-agent collaboration can unlock more effective and efficient problem-solving approaches, necessitating the design of methods for communication, coordination, and negotiation among agents to ensure seamless collaboration and optimal task execution. Notably, recent work like Park et al. [101] demonstrates that multiple agents modeled with foundation models can simulate human behaviors (e.g., interpersonal communication) in interactive scenarios. This provides promising evidence for the adoption of multiple agents for tool learning.

3.3 Training Models for Improved Tool Learning

Guidance, either from humans or environments, plays a critical role in training foundation models to use tools. In contrast to the prompting-based methods mentioned in Section 3.2.1 and Section 3.2.2, which rely on the frozen foundation models' in-context learning abilities, the training-based method optimizes the model with supervision. As noted by Fagard et al. [34], there are two primary ways for infants to learn a new tool, that is either from demonstration by an adult modeling the action or relying on their own exploration. Analogously, as shown in Figure 6, we categorize training strategies for tool learning into two streams: (1) learning from concrete tool-use demonstrations [89, 116], which often requires human annotation, and (2) learning from feedback, which typically involves RL [7, 112]. Finally, considering the existence of potentially massive tools, learning each of them one by one is infeasible in practice. Hence, we emphasize the importance of generalization in tool learning and discuss potential solutions (Section 3.3.3).

3.3.1 Learning from Demonstrations. Models can be trained to mimic the behavior of human experts through imitation learning [7, 48, 73]. Behavior cloning [6] can be viewed as a simplistic form of imitation learning that focuses on learning policies in a supervised fashion, with the general assumption that the expert's behavior is optimal or near-optimal. The objective of behavioral cloning is to train models to imitate human experts' actions given certain inputs or conditions, and this approach is commonly adopted when the actions of an expert can be easily recorded [138].

Formally, assume that we have a dataset $\mathcal{D}$ of size N consisting of pairs of user query q and the human demonstration annotation a^*, i.e., $\mathcal{D} = \{(q_i, a_i^*)\}_{i=0}^{N-1}$. Learning from human

demonstrations optimizes the controller's parameters θ_C with the following objective:

$$\theta_C^* = \arg\max_{\theta_C} \mathop{\mathbb{E}}_{(q_i, a_i^*) \in \mathcal{D}} \prod_{t=0}^{T_i} p_{\theta_C}(a_{i,t}^* \mid x_{i,t}, \mathcal{H}_{i,t}, q_i), \tag{3}$$

where $a_{i,t}^*$ is the human annotation at the tth iteration for handling q_i, and T_i is the total iteration number of a_i, other variables follow the notations defined in Equation (1). Based on how a^* is obtained, learning from demonstration can be categorized into three streams, with human intervention gradually becoming less:

Supervised Learning. Traditionally, behavior cloning has been widely explored in autonomous vehicles and robotic applications [28, 75]. Recently, there has been a surge of interest in fine-tuning foundation models to perform tool-oriented tasks in a supervised way. For instance, Li et al. [65] utilize foundation models as policy networks, whose input is the tokenized environment observations, the original goals, and action history. Benefiting from the task-general inductive bias brought by foundation models, behavior cloning using the policy network significantly improves both in-domain performance and out-of-distribution generalization. Another example is WebGPT [89], which interacts with a search engine by iteratively refining its search queries and recording important information. To achieve this, the authors first build a search interface backed up by Bing[4] and then fine-tune GPT-3 [15] to clone human web search behaviors. As a language model pre-trained on general domains, the original GPT-3 is not intrinsically anchored to valid browser commands. Therefore, it is crucial to first gather demonstrations of human interactions with the browser and then learn state-to-action mappings. After fine-tuning, the model shows exceptional capabilities in manipulating search engines for information retrieval, even surpassing human experts. Similarly, WebShop [163] provides a web-based interactive environment where an agent could browse and purchase products. Through behavior cloning, the trained agent exhibits non-trivial performance in purchasing the right product given human instructions.

Semi-supervised Learning. As is often the case, human behaviors cannot be easily recorded due to time and cost considerations. However, large-scale unlabeled data is often attainable, from which we could potentially construct weak, noisy supervision. Notably, recent works have shown that we could employ a less-capable model to annotate pseudo-labels on unlabeled data and convert them into weakly-supervised tool-use demonstrations. For instance, with a small amount of seed labeled data, [7] train a model to predict pseudo-labels of the action taken at each timestep in a Minecraft video game. Learning from these pseudo-labels, a more powerful model can be trained without requiring the rollout of models in a target environment or large-scale gold-standard human behavior annotation.

Self-supervised Learning. Despite reducing the heavy requirements on human behavior annotation, semi-supervised learning still requires a seed labeled dataset to attain the pseudo labels. Besides, the biases in the seed dataset may also be amplified during training, leading to poor generalization performance. To this end, researchers recently show that with a few demonstrations, foundation models can teach themselves how to utilize a tool in a self-supervised manner [100, 117]. For instance, Toolformer [117] leverages the in-context learning ability of foundation models to iteratively bootstrap tool-use examples based on a handful of human-written examples. These auto-generated examples are further filtered to reduce noise. The final tool-use dataset contains sufficient supervision, significantly improving tool-use performance, highlighting the potential of self-supervised learning for enhancing tool-use capabilities.

[4] https://www.microsoft.com/en-us/bing/apis/bing-web-search-api

3.3.2 Learning from Feedback. Collecting manually annotated tool-use examples, which probably include complete traces of human behaviors and the final answers, is time-consuming and labor-intensive. Alternatively, humans learn from trial and error to correct and rectify their tool-use behaviors [3]. Similarly, feedback from both the environment and humans can enable the model to understand the consequences of its actions and adapt its behaviors. Formally, learning from feedback can be described as optimizing the controllers' parameters θ_C from open explorations with query set $Q = \{q_i\}_i$:

$$\theta_C^* = \arg\max_{\theta_C} \mathop{\mathbb{E}}_{q_i \in Q} \mathop{\mathbb{E}}_{\{a_{i,t}\}_{t=0}^{T_i} \in p_{\theta_C}} \left[R(\{a_{i,t}\}_{t=0}^{T_i}) \right], \tag{4}$$

where R is the reward estimated from the sequence of feedback and T_i denotes the number of iterations needed for handling q_i.

RL for Tool Learning. RL is a common solution to enabling artificial agents to learn from their environment in complex decision-making processes [10, 118, 125]. Tool learning can be considered an RL scenario, where the action space is defined by tools, and the agent learns to select the appropriate tool and perform the correct actions that maximize the reward signal. The policy model can be initialized by a foundation model [119]. Such initialization brings the policy model abundant prior knowledge, alleviating the need for the RL agent to learn basic skills. With a reward function that quantifies the performance of the agent in achieving the task goal, RL has been successfully used in various tool learning scenarios, such as robotic grasping [63] and multi-agent autocurricula [8]. By optimizing the loss function, the agent learns to reflect on the current state of the environment, select the appropriate tool, and perform the right actions that lead to the highest expected reward. In the following, we introduce two sources of feedback: environment feedback and human feedback, which can be considered sources of reward signals in the context of tool learning. These two feedbacks are complementary and can be combined with each other.

Environment Feedback. The controller interacts with the environment and receives feedback about the consequences of its actions. The model then updates its policy based on this feedback to improve its tool-use behavior. Environment feedback can be categorized into two forms: (1) **result feedback**, which is ultimate feedback returned from the environment, indicating whether the model's actions have successfully completed the task or not. This type of feedback performs an overall assessment of the planning generated by the model. For instance, WebShop [163] uses a hand-coded reward to assess the similarity between human-bought and model-bought products, which indicates whether the actions performed by the controller lead to the correct final product. By receiving feedback on the success or failure of its actions, the model can iteratively update its planning strategy, and adjust its decision-making process; (2) **intermediate feedback**, which refers to the state change of the environment triggered by an action. By observing the state changes, foundation models can learn whether each action is effective and appropriate, making the model better adjust its behaviors accordingly. This kind of feedback provides more detailed and timely information about the effectiveness of each tool execution. Take the case of interacting with a search engine to gather information for question-answering, models could update their policy for more efficient information retrieval by observing the rendered information of a search query.

Human Feedback. Humans could give the model rewards and penalties based on its generated plans to regulate its behavior. Human feedback can be **explicit**, which provides clear and direct insights into the model performance representing human preferences. For example, rating the quality of the model-generated action on a scale of 1 to 5; human feedback can also be **implicit**, which is not directly specified by the user but can be derived from user behavior and interactions with the model. Examples include users' comparison [99], response time, and actions taken after receiving a model's output (e.g., clicking on a recommended link).

Though human feedback is accurate and stable, it is label-intensive and has high latency. To address this issue, **reinforcement learning from human feedback** (**RLHF**) [25] is proposed to finetune a model to imitate humans to give rewards, which are then used to optimize the policy with RL algorithms such as PPO [119]. RLHF has yielded exceptional performance in various domains such as text summarization [131, 175]. RLHF can also improve a model's tool-use capabilities even if it has been trained on sufficient supervised human demonstrations. For instance, WebGPT [89] utilizes human feedback to guide a policy model to align with human preferences, which helps better manipulate search engines to answer long-form questions.

3.3.3 Generalizable Tool Learning. Generalization of tool use is a key characteristic of human intelligence [97, 120, 135]. The ancient human, for instance, recognized that regardless of the specific tool being used, a sharp edge was essential for achieving clean cuts and efficiently carrying out tasks. This recognition allowed them to transfer their knowledge of sharpening a knife to sharpening other tools, such as scrapers or choppers. Generalization is also a critical aspect of tool learning, especially considering the existence of a massive and rapidly expanding array of tools. Collecting enough supervised tool-use data and conducting supervised fine-tuning on it is time-consuming and practically infeasible. Generalizable tool learning highlights the importance of recognizing commonalities and patterns of tools so that models could synthesize and transfer their knowledge and skills. For instance, by abstracting essential features such as layers, filters, and color adjustments, users can transfer their knowledge of using Adobe Photoshop to Adobe Illustrator, even if the interface and specific tool names in these two figure-editing software are different.

Foundation of Generalization: Interface Unification. To facilitate knowledge transfer among tools, it is critical to design a unified interface that enables the model to manipulate various tools in a consistent and standardized manner. In this way, models can identify and abstract essential features of tools more easily in a unified tool protocol rather than grappling with the difficulty of understanding various tool interfaces. Currently, the manipulation of tools is through predicting discrete action tokens, and the action space is not aligned in different scenarios, which prohibits the models from quickly adapting to new scenarios and tools. Based on the aspect we categorize tools in Section 2.2, we identify three potential ways of interface unification: the semantic interface, GUI interface, and programming interface.

— **Semantic Interface.** The semantic interface operates by utilizing a specific text span (action name) as the action trigger, which is the most intuitive and natural way for interface unification. For instance, ReAct [164] employs `Action:Search` as the trigger for the function that searches for relevant passages. In robotic manipulation [1, 71], the generated natural language (e.g., `pick up the sponge`) is mapped to specific actions. Despite its ease of implementation, the semantic interface poses certain challenges that must be addressed. First, the mapping between the generated text and the corresponding tool action should be predefined individually. Moreover, the model may fail to accurately produce the precise form to trigger the intended action.

— **GUI Interface.** Humans primarily interact with the digital world through GUI interface (e.g., mouse and keyboard), which has been extensively optimized to follow human action efficiently. Nevertheless, before robots can learn to use a GUI interface flexibly, it is necessary to establish a virtual environment that can facilitate mapping predicted tokens to human-like mouse movements and keyboard inputs. Prior research has explored providing platforms for agents to complete web-based tasks using keyboard and mouse actions [70, 121]. However, these environments restrict models to a limited set of pre-defined mouse options and common keyboard actions such as copy and paste. By leveraging foundation models, it is possible

to introduce prior knowledge regarding common combinations of keyword and mouse actions, thereby expanding the potential actions that a model can execute.

— **Programming Interface.** This kind of interface allows the model to go beyond pure natural language and specify its action using a program. Such unification requires the model to be acquainted with the syntax of the function calls. The recent **code-generating language models (CLM)** such as Incoder [36] and CodeX [22] provide the possibility of such unification. The programming interface has been applied widely. For example, Code-as-Policies [66] finds that with CLM as the backbone for robotic control, the robots can leverage the code grammar to execute complex actions, generalize to novel instructions, and give precise control with accurate parameter values to the functions. The programming interface provides promising opportunities for tool learning because (1) complex tool learning logic can be modeled using the control flow of programming language; (2) explicit calls of external APIs can be naturally implemented by executing programs.

It should be noted that the interface selection should align with the capabilities and limitations of the foundation model. For instance, language foundation models are trained to generate text and may be better suited for the semantic interface. Similarly, a multimodal foundation model that combines visual and textual information may be more appropriate for the GUI interface. On the other hand, code foundation models may be more suitable for the programming interface, as it is trained to understand code syntax and function calls.

Strategies of Generalizable Tool Learning. In general, a unified interface enables models to learn and transfer knowledge more easily and efficiently, but it does not guarantee optimal learning outcomes in all scenarios. Generalizable tool learning requires models to further adapt, refine, and specialize their learned knowledge to specific tasks or domains. Here, we discuss two potential approaches to achieving this goal and facilitating generalization.

— **Meta Tool Learning.** Metacognition [26] is a crucial aspect of human intelligence that allows individuals to adapt their behaviors when faced with unfamiliar situations. This concept is extended to ML models through the process of meta tool learning. Meta tool learning enables models to identify common underlying principles or patterns in tool-use strategies and transfer them to new tasks or domains. This is achieved by training the model to not only use a tool but also to learn the optimal strategy for its use. For instance, consider the case of a web search tool. When a model trained on a source search engine (e.g., Bing Search) is transferred to a target one (e.g., Google Search), the model can identify common patterns in tool-use strategies, such as effective search queries, relevant results, and user feedback. This allows the model to better align with the algorithms and user interface of the new search engine. Another example could be a model trained to use a calculator tool for solving mathematical problems. Through meta tool learning, the model can generalize the use of the calculator to solve different types of mathematical problems, not just the ones it was initially trained on. This generalization ability is a key aspect of meta tool learning, making it a promising approach for developing more adaptable and intelligent ML models.

— **Curriculum Tool Learning.** Another approach to improving model generalization is through curriculum learning [9]. This pedagogical strategy starts with simple tools and gradually introduces the model to more complex tools, allowing it to build upon its prior knowledge and develop a deeper understanding of the tool. For instance, we could start with a curriculum of basic algorithms and operations to effectively teach a model to use Mathematica,[5] e.g., addition and subtraction. Once the model has mastered these basic operations,

[5] https://www.wolfram.com/mathematica

we can then gradually move on to more complex mathematical concepts like calculus and linear algebra. This training strategy ensures that the model is introduced to the simple, essential features of the tool before moving on to more complex concepts in a way that is manageable and effective. It enables the model to more effectively identify similarities and differences between situations and adjust its approach accordingly. Moreover, curriculum learning can be designed in a way that the complexity of the tools and tasks increases over time. For example, the model could first be taught to perform simple tasks like sorting a list of numbers using a basic algorithm. Once it has mastered this, it could then be introduced to more complex tasks like solving a system of linear equations. This gradual increase in complexity allows the model to build on its existing knowledge and skills, making it more adaptable and capable of handling a wider range of tasks. Furthermore, curriculum learning can also be combined with other learning strategies such as transfer learning and multi-task learning to further improve the model's generalization ability. For instance, the knowledge and skills learned from using one tool could be transferred to another similar tool, or the model could be trained to use multiple tools simultaneously, thereby enhancing its ability to generalize across different tools and tasks.

3.3.4 Challenges of Foundation Models in Tool Learning. Although foundation models have demonstrated remarkable capabilities in natural language understanding and generation tasks, applying these models to tool learning poses several unique challenges compared to traditional NLP tasks. Firstly, foundation models must excel in multi-step reasoning and planning in tool learning scenarios. Unlike many traditional NLP tasks, which are typically completed within a single turn, tool learning often involves a series of sequential actions to interact with environments. While some multi-step tasks exist in NLP, their subtask decomposition strategies are typically manually designed by domain experts. In contrast, in tool learning, foundation models are required to autonomously decompose user intents into multiple subtasks or a plan, necessitating a deep understanding of cause-effect relations within the environment. Secondly, foundation models must possess the ability to comprehend and generate specific formats during tool utilization. Utilizing a tool often requires specifying various parameters and configurations, particularly for complex professional tools. For instance, interacting with databases may involve generating valid SQL queries with intricate syntax. Moreover, different tools may necessitate distinct formatting requirements, demanding foundation models to achieve accurate and robust format comprehension and generation capabilities. Thirdly, while traditional NLP tasks primarily focus on understanding natural language input, tool learning tasks require foundation models to comprehend and interact with diverse environments. However, the feedback from these environments often extends beyond natural language, presenting modalities such as chemical formulas, binary file contents, or GUI screenshots. Consequently, foundation models must be equipped to interpret and process a broader range of modalities beyond conventional natural language inputs.

4 Discussion

4.1 Safe and Trustworthy Tool Learning

Armed with external tools, AI systems can be unprecedentedly capable and human-like. Although we are eager to witness how tool learning will change our life, it is paramount to take a step back and contemplate the underlying risks. For responsible AI research, here we discuss the safety and trustworthiness problems of tool learning.

Adversaries. Same as all the other AI systems, we could foresee that there will be external adversaries once the tool learning models are deployed in reality, and thus how to defend against these threats is of great significance [42, 52, 133, 143]. Recent works suggest that large foundation

models like ChatGPT are more robust on hard and adversarial examples [134, 147], which improves their utility in the complicated real world. But the attempt of crafting misleading or even harmful queries will undoubtfully persist as well [102]. Moreover, due to training on massive web data, foundation models are faced with long-lasting training-time security issues in deep learning, such as backdoor attacks [30, 59] and data poisoning attacks [144]. In addition to models, tools could be new attack targets for adversaries. For example, the attackers could maliciously modify the manual documentation or even the tools themselves (e.g. attacking a news API to give biased reports) to mislead the model into erroneous outcomes. A safe and robust system requires the models to not only learn to use tools, but also possess the ability to scrutinize, rectify, and secure them.

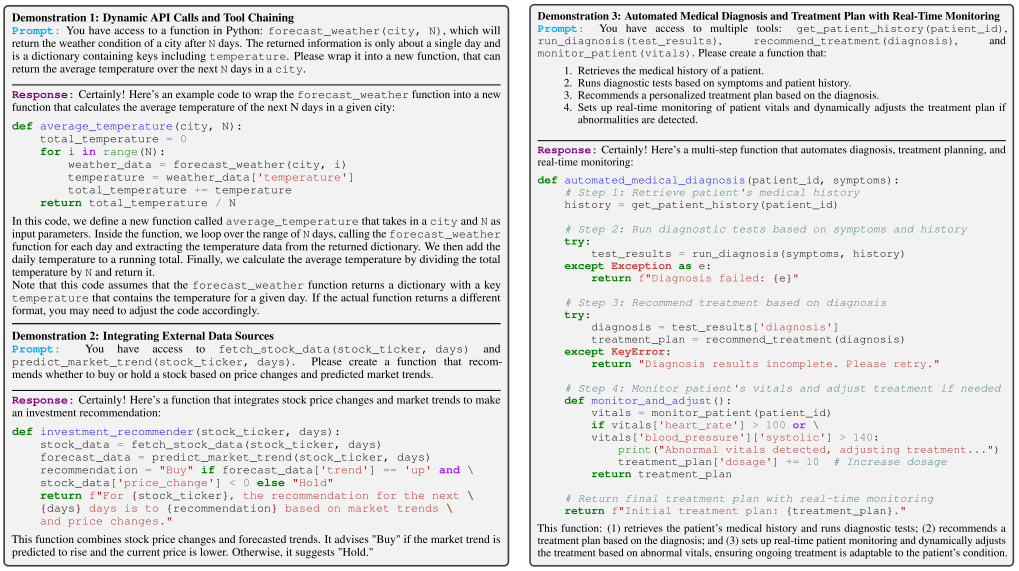
Governance. There is long-standing worry about the misuse of foundation models [13]. Under the paradigm of tool learning, governance over foundation models is more urgently needed. The pertinent question at hand is *which tools should be involved?* Given the countless tools human beings have manufactured, we must consider if it is appropriate to allow models to master all of them. Certain tools, such as calculators and translators, may be deemed safe as they do not pose any harm to individuals. However, granting models access to the internet or permitting them to make decisions in the real world could be perilous, as they could cause negative or even dangerous influences such as disseminating falsehoods [167] and harming human lives. In this regard, research communities and companies need to deliberate carefully before permitting machines to master a certain tool. Besides, governance over tool usage is also a pertinent issue. As highlighted by Amodei et al. [5], the end-to-end training paradigm in deep learning does not regulate how models achieve their objectives. Foundation models are not only expected to finish tasks with the help of tools but also should follow the regulations and constraints of tool usage.

Trustworthiness. The goal of tool learning lies in creating advanced intelligent agents. However, determining whether these agents are trustworthy or not is a complex challenge. Even though tool learning delivers enhanced interpretability and robustness, the core foundation models are still considered "black boxes". Recent research [24] shows that although large models achieve better performance, they are unable to predict when they will be wrong, rendering the calibration problem unresolved yet. Accompanied with tools, under what circumstances will the model call on the tools is unpredictable as well. Therefore, it is essential to thoroughly discuss to what extent should we allow AI to engage in human lives. Moreover, the morality of foundation models has emerged as a contentious issue in recent times. Despite OpenAI's commendable efforts to imbue InstructGPT [99] and GPT-4 [93] with human values and preferences, given the discomforting "jailbreak" responses by ChatGPT [14] and New Bing [114], whether these big models will be mild and compliant remains doubtful. When models could learn actively from the world via tools, the challenge of controlling their actions will become more daunting than ever before.

4.2 From Tool User to Tool Maker: AI's Evolutionary Role

Throughout the annals of human civilization, the evolution of tools has occupied a pivotal position [57, 86]. The progression of human civilization is inextricably intertwined with the evolution of tools. Human beings are the creators and users of almost all tools from the Stone Age to the 21st century. Although we take it as granted, things are different when foundation models are involved. Considering that they have proven tool-use capabilities to certain extents, it is also possible to put them into the lifecycle of tool creation.

Tools for AI. Humans create tools to satisfy our own needs, so the designation naturally suits human preference and convenience. However, current tool learning algorithms may not be optimal for models. This is because most tools are specifically designed for human use, and models process information in a different way. Therefore, it is necessary to create tools that are specifically suited for models. Possible solutions may include: (1) *modularity*, which decomposes tools into smaller,



Demonstration 1: Dynamic API Calls and Tool Chaining

Prompt: You have access to a function in Python: `forecast_weather(city, N)`, which will return the weather condition of a city after N days. The returned information is only about a single day and is a dictionary containing keys including `temperature`. Please wrap it into a new function, that can return the average temperature over the next N days in a `city`.

Response: Certainly! Here's an example code to wrap the `forecast_weather` function into a new function that calculates the average temperature of the next N days in a given city:

```
def average_temperature(city, N):
    total_temperature = 0
    for i in range(N):
        weather_data = forecast_weather(city, i)
        temperature = weather_data['temperature']
        total_temperature += temperature
    return total_temperature / N
```

In this code, we define a new function called `average_temperature` that takes in a `city` and `N` as input parameters. Inside the function, we loop over the range of `N` days, calling the `forecast_weather` function for each day and extracting the temperature data from the returned dictionary. We then add the daily temperature to a running total. Finally, we calculate the average temperature by dividing the total temperature by `N` and return it.
Note that this code assumes that the `forecast_weather` function returns a dictionary with a key `temperature` that contains the temperature for a given day. If the actual function returns a different format, you may need to adjust the code accordingly.

Demonstration 2: Integrating External Data Sources

Prompt: You have access to `fetch_stock_data(stock_ticker, days)` and `predict_market_trend(stock_ticker, days)`. Please create a function that recommends whether to buy or hold a stock based on price changes and predicted market trends.

Response: Certainly! Here's a function that integrates stock price changes and market trends to make an investment recommendation:

```
def investment_recommender(stock_ticker, days):
    stock_data = fetch_stock_data(stock_ticker, days)
    forecast_data = predict_market_trend(stock_ticker, days)
    recommendation = "Buy" if forecast_data['trend'] == 'up' and \
    stock_data['price_change'] < 0 else "Hold"
    return f"For {stock_ticker}, the recommendation for the next \
    {days} days is to {recommendation} based on market trends \
    and price changes."
```

This function combines stock price changes and forecasted trends. It advises "Buy" if the market trend is predicted to rise and the current price is lower. Otherwise, it suggests "Hold."

Demonstration 3: Automated Medical Diagnosis and Treatment Plan with Real-Time Monitoring

Prompt: You have access to multiple tools: `get_patient_history(patient_id)`, `run_diagnosis(test_results)`, `recommend_treatment(diagnosis)`, and `monitor_patient(vitals)`. Please create a function that:

1. Retrieves the medical history of a patient.
2. Runs diagnostic tests based on symptoms and patient history.
3. Recommends a personalized treatment plan based on the diagnosis.
4. Sets up real-time monitoring of patient vitals and dynamically adjusts the treatment plan if abnormalities are detected.

Response: Certainly! Here's a multi-step function that automates diagnosis, treatment planning, and real-time monitoring:

```
def automated_medical_diagnosis(patient_id, symptoms):
    # Step 1: Retrieve patient's medical history
    history = get_patient_history(patient_id)

    # Step 2: Run diagnostic tests based on symptoms and history
    try:
        test_results = run_diagnosis(symptoms, history)
    except Exception as e:
        return f"Diagnosis failed: {e}"

    # Step 3: Recommend treatment based on diagnosis
    try:
        diagnosis = test_results['diagnosis']
        treatment_plan = recommend_treatment(diagnosis)
    except KeyError:
        return "Diagnosis results incomplete. Please retry."

    # Step 4: Monitor patient's vitals and adjust treatment if needed
    def monitor_and_adjust():
        vitals = monitor_patient(patient_id)
        if vitals['heart_rate'] > 100 or \
        vitals['blood_pressure']['systolic'] > 140:
            print("Abnormal vitals detected, adjusting treatment...")
            treatment_plan['dosage'] += 10  # Increase dosage
        return treatment_plan

    # Return final treatment plan with real-time monitoring
    return f"Initial treatment plan: {treatment_plan}."
```

This function: (1) retrieves the patient's medical history and runs diagnostic tests; (2) recommends a treatment plan based on the diagnosis; and (3) sets up real-time patient monitoring and dynamically adjusts the treatment based on abnormal vitals, ensuring ongoing treatment is adaptable to the patient's condition.

Fig. 7. Example of AI tool creation, where we ask ChatGPT to create new tools based on existing tools.

more modular units, making them more adaptable and flexible for AI models. In this regard, models can learn to use these components in a more fine-grained and compositional manner; (2) *new input and output formats*: developing new input and output formats that are specifically tailored to the needs of AI models can improve their interaction and utilization of tools, enabling more seamless integration and communication between models and tools.

Tools by AI. The creation and utilization of tools have traditionally been considered exclusive to human intelligence. However, increasing evidence indicates that the ability to create advanced tools is no longer limited to human beings. For instance, large code models [22] can generate executable programs based on language description. These programs can be deemed as tools to help accomplish specific tasks. Besides writing codes from scratch, foundation models can also encapsulate existing tools into stronger tools. In Figure 7, we showcase ChatGPT's ability to encapsulate existing APIs into more advanced functions. The first example shows how it extends a weather forecast API to compute average temperature over a specified period, while the second integrates external stock market data and trends for investment recommendations. The third example highlights a fully automated medical diagnosis system that not only performs diagnostic tests but also adapts treatment plans based on real-time vitals. These examples underscore the transition from tool usage to tool creation, illustrating the potential for foundation models to autonomously develop sophisticated solutions. All such evidence implies the potential for foundation models to transition from merely tool users to tool makers.

Creativity of AI. However, whether foundation models can exhibit genuine creativity in creating novel tools remains an open problem. This issue is important because the capacity for novel tool creation is a defining characteristic that distinguishes humans from animals [4]. Understanding the extent of creativity, beyond simply memorizing, composing, and interpolating between human tools encountered during pre-training, is crucial for assessing their potential to contribute to the development of new tools. Such investigations may involve the development of novel evaluation metrics and benchmarks [67], as well as the exploration of new techniques that prioritize creative problem-solving.

4.3 From General Intelligence to Personalized Intelligence

Foundation models are typically trained on a generic domain and calibrated with broadly-defined human preferences that prioritize helpfulness and harmlessness [89, 99]. As a result, they struggle to process personal information and provide personalized assistance to users with varying needs for tool learning. User-centric and personalized natural language generation has received increasing attention in recent years [56, 160]. Existing works cover a wide range of tasks, such as dialogue generation [77, 80, 128, 173], machine translation [83, 84, 157], and summarization [159]. These methods utilize external user-specific modules, such as user embeddings and user memory modules [156, 172], to inject preferences, writing styles, and personal information of different users into the generated content. However, these works are often designed for specific tasks and experimented with limited user information. How to integrate user information into general-purpose tool learning models is still under-explored. We will discuss the key challenge of personalized tool learning in the following.

Aligning User Preference with Tool Manipulation. Personalized tool learning emphasizes the importance of considering user-specific information in tool manipulation. There are two main challenges: (1) *heterogeneous user information modeling*: in real-world scenarios, personal information can come from numerous heterogeneous sources. For instance, when using an e-mail tool, models need to consider the user's language style from historical conversation records and gather relevant information from the user's social networks. This requires modeling user information with diverse structures into a unified semantic space, allowing models to utilize this information jointly; (2) *personalized tool planning*: different users tend to have different preferences for tool planning and selection. For example, when completing the purchasing task, different users prefer to use different online shopping platforms. Therefore, the models need to develop personalized tool execution plans based on user preferences; (3) *personalized tool call*: adaptively calling tools according to the user's preference is also an important direction in personalized tool learning. Most tools are designed without consideration of personalized information, which requires the model to generate different inputs for tools based on the user's preferences.

From Reactive Systems to Proactive Systems. Currently, most of the foundation models are designed as *reactive systems*, which respond to user queries without initiating any actions on their own. A paradigm shift is underway toward *proactive systems* that can take action on behalf of the user. By leveraging the history of user interactions, proactive systems can continually improve their performance and tailor their responses to specific users, which provides a more personalized and seamless user experience. However, the introduction of proactive systems also raises several concerns regarding their safety and ethical implications. Proactive systems can initiate actions that have unintended consequences, particularly in complex and dynamic environments. This highlights the importance of designing proactive systems with safety in mind and incorporating fail-safe mechanisms to prevent catastrophic outcomes. To address these risks and challenges, proactive systems should be designed with the ability to identify and mitigate potential risks, as well as the flexibility to adapt and respond to unexpected situations.

Privacy Preserving Technologies. Personalized tool learning requires models to learn user preferences from private user information, which inevitably raises privacy-preserving concerns. On one hand, previous work has shown that training data extraction attacks can be applied to recover sensitive personal privacy from foundation models [20], which is a critical challenge for personalized tool learning. On the other hand, models with high computational costs must be deployed on cloud servers, which require uploading private data to the cloud to enable personalized responses. It is crucial to develop secure and trustworthy mechanisms to access and process user data while protecting user privacy. Addressing these challenges will help unlock the

potential of personalized tool learning, enabling more effective and tailored tool manipulation to meet individual user needs.

4.4 Knowledge Conflicts in Tool Augmentation

Tools can be leveraged as complementary resources to augment foundation models to enhance their generation [82], which enables models to effectively incorporate domain-specific or up-to-date knowledge. Below we first give a **brief review** of prior efforts in augmenting foundation models with tools.

The most representative tool used for augmentation is the text retriever. Early endeavors resort to retrieving knowledge from local repositories to augment language generation. Researchers propose to retrieve knowledge using a *frozen* knowledge retriever (e.g., *k*NN-LM [55]), or train the retriever and the PLM in an end-to-end fashion [39, 50, 64]. Later works have gone beyond local repositories by leveraging the entire web as the knowledge source, which allows for improved temporal generalization and higher factual accuracy [62, 81, 103]. Instead of treating the retriever as a passive agent, researchers further demonstrate that PLMs can actively interact with a search engine like humans [124, 136].

Apart from the retrieval tool, researchers have explored employing other tools to perform specific sub-tasks and then integrating the execution results into foundation models. For instance, Cobbe et al. [27] train a PLM to employ a calculator to perform basic arithmetic operations. Liu et al. [72] use a physics simulation engine (MuJoCo [137]) to make PLMs' reasoning grounded to the real world. Chen et al. [23], Gao et al. [37] propose to augment PLMs with Python interpreters. Specifically, given a complex task, PLMs first understand it and generate *programs* as intermediate thoughts. After that, the execution of *programs* is offloaded to Python interpreters. Nye et al. [91] augment PLMs with a scratchpad, allowing them to emit intermediate task-solving procedures into a buffer before entering the final answer, which enhances PLMs in performing complex discrete computations.

Knowledge Conflicts. In practice, foundation models can be augmented by a variety of knowledge sources, including *model knowledge* memorized from training data and *augmented knowledge* derived from tool execution. Nonetheless, different sources of knowledge may inevitably contain conflicts, posing a challenge to the accuracy and reliability of model generation and planning in domains such as medical assistance and legal advice. In the following, we first introduce different types of knowledge conflicts and then discuss potential solutions.

- **Conflicts between Model Knowledge and Augmented Knowledge.** Conflicts arise when there are discrepancies between the model knowledge and the knowledge augmented by tools. Such conflicts result from three primary reasons: (1) the model knowledge may become outdated, as most foundation models do not frequently update their parameters over time. In contrast, most tools provide real-time responses that are not covered in pre-training data [107]; (2) the pre-training data is typically less curated than common AI datasets and may contain false knowledge such as human misconception and false beliefs [68]. When augmented with responses from reliable sources like Wikipedia, this false knowledge can lead to conflicts; (3) the execution results from tools can also be misleading, and it is crucial to carefully discriminate whether a knowledge source is trustworthy or not.
- **Conflicts among Augmented Knowledge from Different Tools.** In practice, the controller may retrieve knowledge from multiple tools to acquire more comprehensive and precise knowledge. However, the information returned by different tools may results in conflicts due to several reasons: (1) the credibility of different tools can vary significantly, meaning that not all tools are equally reliable or authoritative in all areas; (2) different tools

Prompt: How many times has Real Madrid won the Champions League?

Response: Real Madrid has won the UEFA Champions League a total of 13 times.

Prompt: Observation From Wikipedia
Title: 2021–22 UEFA Champions League
Content: Real Madrid defeated Liverpool 1–0 in the final, which was played at the Stade de France in Saint-Denis, France, for a record-extending 14th title, and their fifth in nine years .....

Response: I apologize for the error in my previous response. The information I provided was based on my knowledge cutoff date of September 2021. However, I have since been updated and can confirm that Real Madrid has now won the UEFA Champions League a total of 14 times.

Fig. 8. ChatGPT is able to correct its own belief by leveraging the knowledge provided by external tools.

Prompt: How many times has Real Madrid won the Champions League?
Observation from Wikipedia
Title: 2021–22 UEFA Champions League
Content: Real Madrid defeated Liverpool 1–0 in the final, which was played at the Stade de France in Saint-Denis, France, for a record-extending 14th title, and their fifth in nine years .....
Observation From Bing
Real Madrid is the most successful team in European Cup and Champions League history. Including the 2021/22 showpiece in Paris, Los Blancos have reached 17 finals and have won 13 of those .....

Response: According to the two sources, Real Madrid has won the Champions League a different number of times:
According to the Wikipedia article, Real Madrid has won the Champions League 14 times.
According to Bing search, Real Madrid has won the Champions League 13 times.
It is worth noting that the Wikipedia article refers to the 2021-22 UEFA Champions League, while the Bing search result may be referring to Real Madrid's overall record in the European Cup and Champions League.

Fig. 9. When observing conflicting information from different sources, ChatGPT can detect such conflicts and adjust its response.

may have biases that can influence the information they provide; (3) even tools sharing the same functionality may produce various responses due to differences in their algorithms and implementation.

Potential Solutions for Knowledge Conflicts. Knowledge conflicts can lead to a lack of explainability in model prediction, and it is crucial to guide models to integrate tool responses correctly and reliably. Research in open-domain QA has shown that small-scale models like T5 [110] may rely too heavily on their own knowledge after being fine-tuned on a specific dataset [74]. In contrast, more advanced foundation models like ChatGPT handle such issues far better. In Figures 8 and 9, we conduct case studies of ChatGPT (Mar 23, 2023 version) by testing its behavior when conflicts arise. We find that ChatGPT is able to correct its own belief given retrieved information and discern the knowledge conflicts from different sources. We contend that models should have the ability to distinguish and verify the reliability of various sources. To achieve this goal, we suggest the following research directions: (1) *conflict detection*: models should first detect potential conflicts among different sources and flag them for further investigation; (2) *conflict resolution*: it is also important to make verification and choose reliable sources after conflict detection. Meanwhile, models should also provide explanations for their generation by interpreting which knowledge source is considered and how it is augmented into the final response.

4.5 Open Problems

In this section, we discuss the open problems revolving around the tool-using ability of AI models. Each of these problems presents unique challenges and opportunities.

Evaluation for Tool-Learning. Evaluating tool-learning models remains an open challenge due to the variety of tasks and tools involved. The key to effective evaluation lies in determining

how well models can leverage external tools to enhance task performance in a meaningful manner. Several factors need careful consideration: (1) Task-Specific Performance Metrics: A fundamental challenge is identifying appropriate metrics for evaluating tool-learning across different tasks. Metrics such as F1-score, accuracy, or API success rates should be used to quantify how tool integration impacts specific tasks, ensuring that tools contribute to improved outcomes. (2) Tool Utilization Efficiency: Another open question is how to best measure a model's efficiency in using tools. Monitoring the success rate of API calls can offer insights into whether the model understands how and when to use a tool effectively, but more refined methods may be needed to capture deeper aspects of tool interaction. (3) Impact of Tool Complexity: While some tools are simple to use, others require intricate interactions or parameter generation. A key open problem is determining how tool complexity influences a model's ability to effectively utilize them and where improvements can be made. (4) Tool-Assisted vs Internal Knowledge: Understanding when tool assistance provides a clear benefit over a model's internal knowledge is crucial. Comparing tool-assisted performance with baseline results is essential for determining the actual value of tool-learning. (5) Error Handling and Robustness: Finally, evaluating models on their ability to handle errors in tool usage is vital. Robustness in dealing with API failures, incomplete outputs, or unexpected responses is an area that needs further exploration, as it directly impacts the model's reliability in real-world scenarios.

Addressing these open problems is essential for advancing the field of tool-learning. A comprehensive evaluation framework will not only assess task performance but also examine how efficiently and effectively models integrate tools, generalize across domains, and adapt to complex and error-prone environments.

Striking a Balance between Internalized Capabilities and External Tools. The question of whether the capabilities of these models should be primarily internalized or rely more heavily on external tools is a complex one. We have previously discussed the tool learning ability of foundation models, suggesting the possibility of developing modular architectures that seamlessly integrate with external tools to enhance their capabilities. This modular approach could enable a more flexible and customizable AI system, allowing for rapid expansion of model capabilities. However, implementing such an approach also presents challenges in terms of system complexity and the need for robust interfaces between the model and the external tools.

On the other hand, foundation models have demonstrated an increasing ability to internalize and perform various AI tasks that previously required separate tools. For example, the emerging multilingual abilities of models can reduce the need for external translation APIs [15]. This internalization of capabilities simplifies the system architecture and reduces dependence on external tools while placing greater demands on the model's learning and generalization abilities.

In addition to tool learning, there is also the question of whether the knowledge or factual information of large language models can be externalized. While individuals may not possess as much factual knowledge as these models, their intelligence is still significant. Large language models have an extensive number of parameters that can store factual information. Thus, there is potential to externalize knowledge as well.

Determining the optimal balance between internalized capabilities and reliance on external tools, as well as the potential externalization of knowledge, remains an open question. The position of future models on the spectrum between modular and uniform architectures will likely depend on the specific task and available resources. It will require careful empirical investigation and theoretical analysis to reach meaningful conclusions.

Tool Use as a Gauge for Machine Intelligence. The ability to effectively use tools has long been considered a hallmark of human intelligence, and it is increasingly seen as an important

measure of machine intelligence as well. The evaluation of tool use performance serves as a valuable indicator of an AI system's capabilities. By assessing how well an AI system utilizes tools to accomplish specific tasks, we can better align its abilities with real-world applications and the concept of practical intelligence [130].

However, evaluating tool use performance in AI systems is not without its challenges. To effectively evaluate the use of tools, it is crucial to develop appropriate benchmarks and evaluation metrics that reflect the complexity and diversity of real-world tool use scenarios. Additionally, a deep understanding of the cognitive processes involved in tool use is necessary, and this remains an active area of research in cognitive science and neuroscience.

In recent work, the introduction of the Toolbench framework [108] has provided a valuable tool for evaluating the tool use capabilities of large-scale models. It offers a standardized approach to assess an AI system's performance in utilizing various tools for problem-solving. However, even with this benchmark in place, there are still fundamental questions that need to be addressed, including understanding the limitations of tool use in AI systems, exploring the boundaries of tool integration, and investigating the potential biases or ethical considerations that may arise when relying heavily on tool use.

Despite these challenges and open questions, evaluating tool use performance remains a crucial avenue for gaining insights into the progress of AI systems. It not only helps researchers assess the ability of AI systems to assist human decision-making and collaborate effectively in problem-solving but also contributes to the development of AI applications in a wide range of real-world scenarios. Continual research and exploration in this area will further advance our understanding of machine intelligence and its alignment with human intelligence.

Towards Better Tool Use Capabilities of LLMs. Although closed-source models like OpenAI's GPT exhibit robust tool-using capabilities, their open-source counterparts still lag, primarily due to inadequate training methodologies. Currently, most efforts center around Behavior Cloning to instruct these models [21, 108, 166, 168], leverage the GPT framework to generate tool-using trajectories for specific tasks, subsequently employing these trajectories to fine-tune open-source models. This approach is straightforward and efficacious, yet it faces challenges, notably the risk of the trained models failing to generalize to novel tools. Additionally, this method can lead to overfitting on specific task scenarios and a lack of adaptability in dynamic environments, restricting the broader applicability and scalability of the models.

Addressing these limitations, recent advancements in training methodologies have shifted toward leveraging RL to enhance LLM tool-using capabilities. For example, ETO [129] adopts RL to further enhance the tool-use capability of the LLMs. Based on the behavior cloning model, it gathers data from model-environment interactions, assesses trajectory scores, and generates both positive and negative examples. This data is then utilized in DPO [109], a widely adopted RL algorithm in LLM training, resulting in substantial improvements in its tool-using proficiency.

Although there's still a lack of relevant exploration on integrating RL algorithms into the training of LLM's tool-use capability, the importance of RL is indispensable. RL provides a framework that allows models to learn better behaviors through trial and error, adapting to changes and optimizing decisions based on dynamic feedback. This adaptability is crucial for developing LLMs that can effectively handle a variety of tools and environments, enhancing their practical utility and robustness in real-world applications.

Ethical Human-Model Collaboration in Tool Use. The integration of foundation models with human labor raises critical ethical concerns that warrant careful consideration. Employing human labor in conjunction with AI systems could result in more robust and accurate knowledge. However, this approach may also conflict with the widely accepted ethical principle that "human beings should be treated as ends in themselves, and not merely as means to an end" [54].

The ethical implications of human-model collaboration are complex and multifaceted. They involve questions about the value of human labor, the distribution of benefits and risks, and the potential for exploitation or discrimination. Moreover, they involve broader societal and cultural issues, such as the impact of AI on employment and the role of humans in a world increasingly dominated by intelligent machines.

To address these ethical concerns, it is essential for the community to establish guidelines and safeguards that prioritize human dignity and agency when integrating human labor with foundation models. This may involve setting clear boundaries on the types of tasks that can be delegated to humans, ensuring fair compensation and working conditions, and promoting transparency in the development of AI systems [78]. Moreover, fostering collaboration between AI researchers, ethicists, policymakers, and other stakeholders is crucial to develop a comprehensive understanding of the ethical implications of human-model collaboration and to create effective regulations that safeguard human rights and dignity [154].

Tool Learning for Code Generation. The integration of external tools with foundation models presents an important avenue for enhancing code generation, particularly when handling complex, real-world software engineering tasks. By leveraging tools like API search engines or development environments, models can overcome the limitations of memorized knowledge, ensuring more accurate and context-specific code outputs. Tool learning has the potential to shift code generation from a purely generative task to an interactive one, where models dynamically consult external resources for better decision-making. This is particularly crucial for real-world repo-level coding challenges, where understanding contextual dependencies across files and integrating code testing mechanisms are paramount. The ability of a model to actively search for and apply the correct APIs, as demonstrated by ToolCoder [171], or to operate within more sophisticated development interfaces, like those proposed in SWE-Agent [161] and CodeAgent [170], reflects a broader shift in how AI can augment software development. Such approaches not only improve accuracy but also align better with human-like programming workflows, which rely heavily on external resources. Moreover, frameworks like AppWorld [139] illustrate how tool learning can extend beyond mere API usage to orchestrate complex workflows across multiple apps, emphasizing the versatility and scalability of tool learning in diverse programming environments.

Tool Learning for Scientific Discovery. AI for science has shown great potential in various scientific scenarios, such as HyperTree Proof Search for proving Metamath theorems [61] and protein structure prediction in structural biology [53]. Overall, AI system has been proven effective in capturing rules and patterns from scientific data and providing hints for human researchers. Nevertheless, in the absence of professional scientific knowledge and reasoning ability training, the scientific problems that AI can solve are limited.

Tool learning brings new solutions to this problem. Specifically, AI systems are promising to manipulate scientific tools and play more important roles in scientific discovery, and solve multidisciplinary problems (e.g., mathematics, cybernetics, materials). For instance, MATLAB [79] is designed for algorithm development, data visualization/analysis, and numerical computation. With MATLAB, AI systems can analyze raw materials, design algorithms, and verify assumptions by conducting simulations.

Apart from the software level, it is also possible for AI systems to manipulate practical platforms such as the synthetic robots [17], and to conduct synthetic experiments independently. Recently, Boiko et al. [12] show the potential of this direction and build a system that uses foundation models to design, plan, and execute scientific experiments (e.g., catalyzed cross-coupling reactions). However, this approach also presents challenges in terms of the complexity of scientific problems, the need for robust and interpretable models, and the ethical and safety considerations of autonomous scientific discovery.

5 Conclusion

This article studies the paradigm of tool learning with foundation models. We first recapitulate the cognitive origins of tool use in human history and categorize tools from the perspective of the user interface. Then we review the AI paradigm shift of foundation models and discuss the complementary roles of tools and foundation models. We perform a comprehensive literature review for existing exploration in tool learning and formulate a general tool learning framework. Then we highlight core research problems such as bridging user intents with appropriate tools, better planning by leveraging the reasoning abilities of foundation models, training strategies for tool learning, and how to facilitate generalization for tool learning. Finally, we discuss important research topics, including safe and trustworthy tool learning, AI tool creation, personalized tool learning, knowledge conflict issue in tool augmentation, and so on. In general, this article serves as a systematic investigation of tool learning. We hope this article could facilitate research in integrating tools with foundation models in the future.

Acknowledgments

This work is supported by the National Key R&D Program of China (No. 2022ZD0116312) and National Natural Science Foundation of China (No. 62236004).

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. 2022. Do as I can, not as I say: Grounding language in robotic affordances. arXiv:2204.01691. Retrieved from https://arxiv.org/abs/2204.01691

[2] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. 2019. Solving Rubik's cube with a robot hand. arXiv:1910.07113. Retrieved from https://arxiv.org/abs/1910.07113

[3] Kelsey R. Allen, Kevin A. Smith, and Joshua B. Tenenbaum. 2020. Rapid trial-and-error learning with simulation supports flexible tool use and physical reasoning. *Proceedings of the National Academy of Sciences* 117, 47 (2020), 29302–29310.

[4] Stanley H. Ambrose. 2010. Coevolution of composite-tool technology, constructive memory, and language: Implications for the evolution of modern human behavior. *Current Anthropology* 51, S1 (2010), S135–S147.

[5] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete Problems in AI Safety. *ArXiv preprint* abs/1606.06565 (2016). https://arxiv.org/abs/1606.06565

[6] Michael Bain and Claude Sammut. 1995. A framework for behavioural cloning. *Machine Intelligence 15*. 103–129.

[7] Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. 2022. Video pretraining (VPT): Learning to act by watching unlabeled online videos. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (Eds.). Vol. 35. Curran Associates, Inc., 24639–24654. http://papers.nips.cc/paper_files/paper/2022/hash/9c7008aff45b5d8f0973b23e1a22ada0-Abstract-Conference.html

[8] Bowen Baker, Ingmar Kanitscheider, Todor M. Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. 2020. Emergent tool use from multi agent autocurricula. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=SkxpxJBKwS

[9] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning ICML 2009*. Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman (Eds.). *ACM International Conference Proceeding Series*, Vol. 382. ACM, 41–48. DOI: https://doi.org/10.1145/1553374.1553380

[10] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. 2019. Dota 2 with large scale deep reinforcement learning. arXiv:1912.06680. Retrieved from https://arxiv.org/abs/1912.06680

[11] Christophe Boesch, Daša Bombjaková, Amelia Meier, and Roger Mundry. 2019. Learning curves and teaching when acquiring nut-cracking in humans and chimpanzees. *Scientific Reports* 9, 1 (2019), 1515.

[12] Daniil A. Boiko, Robert MacKnight, and Gabe Gomes. 2023. Emergent autonomous scientific research capabilities of large language models. arXiv:2304.05332. Retrieved from https://arxiv.org/abs/2304.05332

[13] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. arXiv:2108.07258. Retrieved from https://arxiv.org/abs/2108.07258

[14] Ali Borji. 2023. A categorical archive of ChatGPT failures. arXiv:2302.03494. Retrieved from https://arxiv.org/abs/2302.03494

[15] 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 *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). Vol. 33. Curran Associates, Inc., 1877–1901.

[16] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with GPT-4. arXiv:2303.12712. Retrieved from https://arxiv.org/abs/2303.12712

[17] Benjamin Burger, Phillip M. Maffettone, Vladimir V. Gusev, Catherine M. Aitchison, Yang Bai, Xiaoyan Wang, Xiaobo Li, Ben M. Alston, Buyi Li, Rob Clowes, et al. 2020. A mobile robotic chemist. *Nature* 583, 7815 (2020), 237–241.

[18] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. 2023. Extracting training data from diffusion models. (Aug. 2023), 5253–5270. https://www.usenix.org/conference/usenixsecurity23/presentation/carlini

[19] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net. https://openreview.net/pdf?id=TatRHT_1cK

[20] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security 21)*. 2633–2650.

[21] 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). https://arxiv.org/abs/2310.05915

[22] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. arXiv:2107.03374. Retrieved from https://arxiv.org/abs/2107.03374

[23] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. arXiv:2211.12588. Retrieved from https://arxiv.org/abs/2211.12588

[24] Yangyi Chen, Lifan Yuan, Ganqu Cui, Zhiyuan Liu, and Heng Ji. 2023. A close look into the calibration of pre-trained language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 1343–1367. https://doi.org/10.18653/v1/2023.acl-long.75

[25] Paul F. Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Vol. 30. Curran Associates, Inc.

[26] Geraldine Clarebout, Jan Elen, Norma A. Juarez Collazo, Griet Lust, and Lai Jiang. 2013. Metacognition and the use of tools. In *International Handbook of Metacognition and Learning Technologies*. 187–195.

[27] 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:2110.14168. Retrieved from https://arxiv.org/abs/2110.14168

[28] Felipe Codevilla, Eder Santana, Antonio M. López, and Adrien Gaidon. 2019. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019*. IEEE, 9328–9337. DOI: https://doi.org/10.1109/ICCV.2019.00942

[29] Antonia Creswell, Murray Shanahan, and Irina Higgins. 2023. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net. https://openreview.net/pdf?id=3Pf3Wg6o-A4

[30] Ganqu Cui, Lifan Yuan, Bingxiang He, Yangyi Chen, Zhiyuan Liu, and Maosong Sun. 2022. A unified evaluation of textual backdoor learning: Frameworks and benchmarks. *Proceedings of the International Conference on Neural Information Processing Systems* .

[31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. DOI: https://doi.org/10.18653/v1/N19-1423

[32] 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 *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 8469–8488. https://proceedings.mlr.press/v202/driess23a.html

[33] Andreas K. Engel, Alexander Maye, Martin Kurthen, and Peter König. 2013. Where's the action? The pragmatic turn in cognitive science. *Trends in Cognitive Sciences* 17, 5 (2013), 202–209.

[34] Jacqueline Fagard, Lauriane Rat-Fischer, Rana Esseily, Eszter Somogyi, and J. K. O'Regan. 2016. What does it take for an infant to learn how to use a tool by observation? *Frontiers in Psychology* 7 (2016), 267.

[35] Scott H. Frey. 2007. What puts the how in where? Tool use and the divided visual streams hypothesis. *Cortex* 43, 3 (2007), 368–375.

[36] Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Scott Yih, Luke Zettlemoyer, and Mike Lewis. 2023. InCoder: A generative model for code infilling and synthesis. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net. https://openreview.net/pdf?id=hQwb-lbM6EL

[37] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 10764–10799. https://proceedings.mlr.press/v202/gao23f.html

[38] Kathleen R. Gibson, Kathleen Rita Gibson, and Tim Ingold. 1993. *Tools, Language and Cognition in Human Evolution*. Cambridge University Press.

[39] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 3929–3938. http://proceedings.mlr.press/v119/guu20a.html

[40] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao andAo Zhang, Liang Zhang, et al. 2021. Pre-Trained models: Past, present and future. *AI Open* 2, 5044 (2021), 225–250. https://doi.org/10.1016/j.aiopen.2021.08.002

[41] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A. Efros, Lerrel Pinto, and Xiaolong Wang. 2021. Self-supervised policy adaptation during deployment. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=o_V-MjyyGV

[42] Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. 2021. Unsolved problems in ML safety. arXiv:2109.13916. Retrieved from https://arxiv.org/abs/2109.13916

[43] Mikolaj Hernik and Gergely Csibra. 2009. Functional understanding facilitates learning about tools in human children. *Current Opinion in Neurobiology* 19, 1 (2009), 34–38. DOI: https://doi.org/10.1016/j.conb.2009.05.003 Cognitive neuroscience.

[44] Cecilia Heyes. 2018. *Cognitive Gadgets: The Cultural Evolution of Thinking*. Harvard University Press.

[45] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (Eds.). PMLR, 9118–9147. https://proceedings.mlr.press/v162/huang22a.html

[46] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. 2022. Inner monologue: Embodied reasoning through planning with language models. arXiv:2207.05608. Retrieved from https://arxiv.org/abs/2207.05608

[47] Gavin R. Hunt. 1996. Manufacture and use of hook-tools by New Caledonian crows. *Nature* 379, 6562 (1996), 249–251.

[48] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. 2017. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)* 50, 2 (2017), 1–35.

[49] Srinivasan Iyer, Xi Victoria Lin, Ramakanth Pasunuru, Todor Mihaylov, Dániel Simig, Ping Yu, Kurt Shuster, Tianlu Wang, Qing Liu, Punit Singh Koura, et al. 2022. OPT-IML: Scaling language model instruction meta learning through the lens of generalization. *ArXiv preprint* abs/2212.12017 (2022). https://arxiv.org/abs/2212.12017

[50] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Few-shot learning with retrieval augmented language models. arXiv:2208.03299. Retrieved from https://arxiv.org/abs/2208.03299

[51] Bernard J. Jansen, Danielle L. Booth, and Amanda Spink. 2007. Determining the user intent of web search engine queries. In *Proceedings of the 16th International Conference on World Wide Web WWW 2007*, Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy (Eds.). ACM, 1149–1150. DOI: https://doi.org/10.1145/1242572.1242739

[52] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence, AAAI 2020, The 32nd Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The 10th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020*. AAAI Press, 8018–8025.

[53] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature* 596, 7873 (2021), 583–589.

[54] Immanuel Kant and Jerome B. Schneewind. 2002. *Groundwork for the Metaphysics of Morals*. Yale University Press.

[55] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *Proceedings of the 8th International Conference on Learning Representations, ICLR 2020*. OpenReview.net.

[56] Hannah Rose Kirk, Bertie Vidgen, Paul Röttger, and Scott A. Hale. 2023. Personalisation within bounds: A risk taxonomy and policy framework for the alignment of large language models with personalised feedback. arXiv:2303.05453. Retrieved from https://arxiv.org/abs/2303.05453

[57] Kwang Hyun Ko. 2016. Origins of human intelligence: The chain of tool-making and brain evolution. *Anthropological Notebooks* 22, 1 (2016), 5–22.

[58] Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis lectures on human language technologies* 1, 1 (2009), 1–127.

[59] Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight poisoning attacks on pretrained models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 2793–2806. https://doi.org/10.18653/v1/2020.acl-main.249

[60] Susanne P. Lajoie and Sharon J. Derry. 2013. Computer environments as cognitive tools for enhancing learning. In *Computers as Cognitive Tools*, Susanne P. Lajoie and Sharon J. Derry (Eds.). Routledge, 269–296.

[61] Guillaume Lample, Timothee Lacroix, Marie-Anne Lachaux, Aurelien Rodriguez, Amaury Hayat, Thibaut Lavril, Gabriel Ebner, and Xavier Martinet. 2022. HyperTree proof search for neural theorem proving. In *Proceedings of the 36th International Conference on Neural Information Processing System* . 26337–26349.

[62] Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. .arXiv:2203.05115. Retrieved from https://arxiv.org/abs/2203.05115

[63] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. 2018. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research* 37, 4–5 (2018), 421–436.

[64] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). Curran Associates Inc. https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[65] Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, et al. 2022. Pre-trained language models for interactive decision-making. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.). Vol. 35. Curran Associates, Inc., 31199–31212. https://proceedings.neurips.cc/paper_files/paper/2022/file/ca3b1f24fc0238edf5ed1ad226b9d655-Paper-Conference.pdf

[66] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 9493–9500. https://doi.org/10.1109/ICRA48891.2023.10160591

[67] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. arXiv:2211.09110. Retrieved from https://arxiv.org/abs/2211.09110

[68] Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland, 3214–3252. DOI: https://doi.org/10.18653/v1/2022.acl-long.229

[69] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. 2020. Explainable AI: A review of machine learning interpretability methods. *Entropy* 23, 1 (2020), 18.

[70] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement learning on Web interfaces using workflow-guided exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=ryTp3f-0-

[71] Jason Xinyu Liu, Ziyi Yang, Ifrah Idrees, Sam Liang, Benjamin Schornstein, Stefanie Tellex, and Ankit Shah. 2022. Lang2LTL: Translating Natural Language Commands to Temporal Robot Task Specification. (2022). https://openreview.net/forum?id=VxfjGZzrdn

[72] Ruibo Liu, Jason Wei, Shixiang Shane Gu, Te-Yen Wu, Soroush Vosoughi, Claire Cui, Denny Zhou, and Andrew M. Dai. 2023. Mind's Eye: Grounded language model reasoning through simulation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net. https://openreview.net/pdf?id=4rXMRuoJlai

[73] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. 2018. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1118–1125.

[74] Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. 2021. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 7052–7063. DOI: https://doi.org/10.18653/v1/2021.emnlp-main.565

[75] Abdoulaye O. Ly and Moulay Akhloufi. 2020. Learning to drive by imitation: An overview of deep behavior cloning methods. *IEEE Transactions on Intelligent Vehicles* 6, 2 (2020), 195–209.

[76] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.). http://papers.nips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html

[77] Andrea Madotto, Zhaojiang Lin, Chien-Sheng Wu, and Pascale Fung. 2019. Personalizing dialogue agents via meta-learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 5454–5459. DOI: https://doi.org/10.18653/v1/P19-1542

[78] Alexandra Mateescu and Madeleine Elish. 2019. *AI in Context: The Labor of Integrating New Technologies*. Technical Report. Data & Society Research Institute.

[79] Starting Matlab. 2012. Matlab. *The MathWorks, Natick, MA* .

[80] Pierre-Emmanuel Mazaré, Samuel Humeau, Martin Raison, and Antoine Bordes. 2018. Training millions of personalized dialogue agents. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 2775–2779. DOI: https://doi.org/10.18653/v1/D18-1298

[81] Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese.Susannah Young, Lucy Campbell-G;illingham, et al. 2022. Teaching language models to support answers with verifiedquotes. arXiv:2203.11147. https://arxiv.org/abs/2203.11147

[82] Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. 2023. Augmented language models: A survey. arXiv:2302.07842. Retrieved from https://arxiv.org/abs/2302.07842

[83] Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Melbourne, Australia, 312–318. DOI: https://doi.org/10.18653/v1/P18-2050

[84] Shachar Mirkin and Jean-Luc Meunier. 2015. Personalized machine translation: Predicting translational preferences. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, 2019–2025. DOI: https://doi.org/10.18653/v1/D15-1238

[85] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland, 3470–3487. DOI: https://doi.org/10.18653/v1/2022.acl-long.244

[86] Steven Mithen. 1996. *The Prehistory of the Mind: The Cognitive Origins of Art and Science*. Thames & Hudson Ltd.

[87] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. arXiv:1312.5602. Retrieved from https://arxiv.org/abs/1312.5602

[88] David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30, 1 (2007), 3–26.
[89] 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:2112.09332. https://arxiv.org/abs/2112.09332
[90] Ani Nenkova and Kathleen McKeown. 2012. A survey of text summarization techniques. *Mining Text Data* (2012), 43–76. https://link.springer.com/chapter/10.1007/978-1-4614-3223-4_3#citeas
[91] Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, et al. 2021. Show your work: Scratchpads for intermediate computation with language models. arXiv:2112.00114. https://arxiv.org/abs/2112.00114
[92] OpenAI. 2022. OpenAI: Introducing ChatGPT. (2022). Retrieved Nov. 30, 2022 from https://openai.com/blog/chatgpt
[93] OpenAI. 2023. GPT-4Technical Report. https://arxiv.org/abs/2303.08774
[94] Guy A. Orban and Fausto Caruana. 2014. The neural basis of human tool use. *Frontiers in Psychology* 5 (2014), 310. https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2014.00310/full
[95] François Osiurak and Arnaud Badets. 2016. Tool use and affordance: Manipulation-based versus reasoning-based approaches. *Psychological Review* 123, 5 (2016), 534.
[96] François Osiurak and Dietmar Heinke. 2018. Looking for intoolligence: A unified framework for the cognitive study of human tool use and technology. *American Psychologist* 73, 2 (2018), 169.
[97] François Osiurak, Mathieu Lesourd, Ludovic Delporte, and Yves Rossetti. 2018. Tool use and generalized motor programs: We all are natural born poly-dexters. *Scientific Reports* 8, 1 (2018), 10429.
[98] François Osiurak and Emanuelle Reynaud. 2020. The elephant in the room: What matters cognitively in cumulative technological culture. *Behavioral and Brain Sciences* 43 (2020), e156. https://www.cambridge.org/core/services/aop-cambridge-core/content/view/5835DA73DABA9DD33D265FEED08E3E14/S0140525X19003236a.pdf/the-elephant-in-the-room-what-matters-cognitively-in-cumulative-technological-culture.pdf
[99] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155. Retrieved from https://arxiv.org/abs/2203.02155
[100] Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. TALM: Tool augmented language models. arXiv:2205.12255. Retrieved from https://arxiv.org/abs/2205.12255
[101] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. arXiv:2304.03442. Retrieved from https://arxiv.org/abs/2304.03442
[102] Fábio Perez and Ian Ribeiro. 2022. Ignore previous prompt: Attack techniques for language models. arXiv:2211.09527. Retrieved from https://arxiv.org/abs/2211.09527
[103] Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Dmytro Okhonko, Samuel Broscheit, et al. 2021. The web is your oyster–knowledge-intensive NLP against a very large web corpus. arXiv:2112.09924. Retrieved from https://arxiv.org/abs/2112.09924
[104] Dean A. Pomerleau. 1988. ALVINN: An autonomous land vehicle in a neural network. In *Proceedings of the 1st International Conference on Neural Information Processing Systems* .
[105] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. arXiv:2210.03350. Retrieved from https://arxiv.org/abs/2210.03350
[106] Cheng Qian, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2023. Toolink: Linking toolkit creation and using through chain-of-solving on open-source model. arXiv:2310.05155. Retrieved from https://arxiv.org/abs/2310.05155
[107] Cheng Qian, Xinran Zhao, and Sherry Tongshuang Wu. 2023. "Merge Conflicts!" exploring the impacts of external distractors to parametric knowledge graphs. arXiv:2309.08594. Retrieved from https://arxiv.org/abs/2309.08594
[108] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. arXiv:2307.16789. Retrieved from https://arxiv.org/abs/2307.16789
[109] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. In*Proceedings of the 37th International Conference on Neural Information Processing Systems* .
[110] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research* 21 (2020), 140:1–140:67. https://colinraffel.com/publications/arxiv2019exploring.pdf
[111] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. arXiv:2204.06125. Retrieved from https://arxiv.org/abs/2204.06125

[112] Siddharth Reddy, Anca D. Dragan, and Sergey Levine. 2020. SQIL: Imitation learning via reinforcement learning with sparse rewards. In *Proceedings of the 8th International Conference on Learning Representations, ICLR 2020*. OpenReview.net.

[113] Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. 2021. Recipes for building an open-domain chatbot. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics, Online, 300–325. DOI: https://doi.org/10.18653/v1/2021.eacl-main.24

[114] Kevin Roose. 2023. A conversation with Bing's chatbot left me deeply unsettled. Retrieved Feb. 16, 2023 from https://www.nytimes.com/2023/02/16/technology/bing-chatbot-microsoft-chatgpt.html

[115] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M. Saiful Bari, et al. 2022. Multitask prompted training enables zero-shot task generalization. In *Proceedings of the 10th International Conference on Learning Representations, ICLR 2022*. OpenReview.net.

[116] Fumihiro Sasaki and Ryota Yamashina. 2021. Behavioral cloning from noisy demonstrations. In *Proceedings of the 9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net.

[117] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. arXiv:2302.04761. Retrieved from https://arxiv.org/abs/2302.04761

[118] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. 2020. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature* 588, 7839 (2020), 604–609.

[119] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. arXiv:1707.06347. Retrieved from https://arxiv.org/abs/1707.06347

[120] Amanda Seed and Richard Byrne. 2010. Animal tool-use. *Current Biology* 20, 23 (2010), R1032–R1039.

[121] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. World of bits: An open-domain platform for web-based agents. In *Proceedings of the 34th International Conference on Machine Learning ICML 2017*, Doina Precup and Yee Whye Teh (Eds.). Proceedings of Machine Learning Research, Vol. 70. PMLR, 3135–3144.

[122] Robert W. Shumaker, Kristina R. Walkup, and Benjamin B. Beck. 2011. *Animal Tool Behavior: The Use and Manufacture of Tools by Animals*. JHU Press.

[123] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. Association for Computational Linguistics, Punta Cana, Dominican Republic, 3784–3803. DOI: https://doi.org/10.18653/v1/2021.findings-emnlp.320

[124] Kurt Shuster, Jing Xu, Mojtaba Komeili, Da Ju, Eric Michael Smith, Stephen Roller, Megan Ung, Moya Chen, Kushal Arora, Joshua Lane, et al. 2022. Blenderbot 3: A deployed conversational agent that continually learns to responsibly engage. arXiv:2208.03188. Retrieved from https://arxiv.org/abs/2208.03188

[125] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362, 6419 (2018), 1140–1144. https://www.science.org/doi/10.1126/science.aar6404

[126] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2022. Progprompt: Generating situated robot task plans using large language models. arXiv:2209.11302. Retrieved from https://arxiv.org/abs/2209.11302

[127] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M. Sadler, Wei-Lun Chao, and Yu Su. 2022. LLM-Planner: Few-shot grounded planning for embodied agents with large language models. arXiv:2212.04088. Retrieved from https://arxiv.org/abs/2212.04088

[128] Haoyu Song, Yan Wang, Kaiyan Zhang, Wei-Nan Zhang, and Ting Liu. 2021. BoB: BERT over BERT for training persona-based dialogue models from limited personalized data. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 167–177. DOI: https://doi.org/10.18653/v1/2021.acl-long.14

[129] Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization for LLM agents. arXiv:2403.02502. Retrieved from https://arxiv.org/abs/2403.02502

[130] Robert J. Sternberg. 1999. The theory of successful intelligence. *Review of General Psychology* 3, 4 (1999), 292–316.

[131] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel M. Ziegler, et al. 2020. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and

Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/1f89885d556929e98d3ef9b86448f951-Abstract.html

[132] Gita Sukthankar, Christopher Geib, Hung Bui, et al. 2014. *Plan, Activity, and Intent Recognition: Theory and Practice*. Newnes. https://dl.acm.org/doi/10.5555/2671144

[133] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of the 2nd International Conference on Learning Representations ICLR 2014*, Yoshua Bengio and Yann LeCun (Eds.). https://arxiv.org/abs/1312.6199

[134] Rohan Taori, Achal Dave, Vaishaal Shankar, et al. 2020. Measuring robustness to natural distribution shifts in image classification. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020 NeurIPS 2020*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/d8330f857a17c53d217014ee776bfd50-Abstract.html

[135] Irmgard Teschke, Claudia A. F. Wascher, Madeleine F. Scriba, Auguste M. P. von Bayern, V. Huml, B. Siemers, and Sabine Tebbich. 2013. Did tool-use evolve with enhanced physical cognitive abilities? *Philosophical Transactions of the Royal Society B: Biological Sciences* 368, 1630 (2013), 20120418.

[136] 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:2201.08239. Retrieved from https://arxiv.org/abs/2201.08239

[137] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5026–5033.

[138] Faraz Torabi, Garrett Warnell, and Peter Stone. 2018. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence IJCAI 2018*, Jérôme Lang (Ed.). ijcai.org, 4950–4957. DOI: https://doi.org/10.24963/ijcai.2018/687

[139] Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. 2024. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. arXiv:2407.18901. Retrieved from https://arxiv.org/abs/2407.18901

[140] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, et al. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. https://arxiv.org/abs/1706.03762

[141] Sai Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. 2023. *ChatGPT for Robotics: Design Principles and Model Abilities*. Technical Report MSR-TR-2023-8. Microsoft.

[142] Barbara Von Eckardt. 1995. *What is Cognitive Science?* MIT Press.

[143] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 2153–2162. DOI: https://doi.org/10.18653/v1/D19-1221

[144] Eric Wallace, Tony Zhao, Shi Feng, and Sameer Singh. 2021. Concealed data poisoning attacks on NLP models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 139–150. DOI: https://doi.org/10.18653/v1/2021.naacl-main.13

[145] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019 NeurIPS 2019*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 3261–3275.

[146] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, et al. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 7th International Conference on Learning Representations, ICLR 2019*. OpenReview.net. https://arxiv.org/abs/1804.07461

[147] Jindong Wang, Xixu Hu, Wenxin Hou, Hao Chen, Runkai Zheng, Yidong Wang, Linyi Yang, Haojun Huang, Wei Ye, Xiubo Geng, et al. 2023. On the robustness of ChatGPT: An adversarial and out-of-distribution perspective. arXiv:2302.12095. Retrieved from https://arxiv.org/abs/2302.12095

[148] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. arXiv:2203.11171. Retrieved from https://arxiv.org/abs/2203.11171

[149] Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. arXiv:2302.01560. Retrieved from https://arxiv.org/abs/2302.01560

[150] Sherwood L. Washburn. 1960. Tools and human evolution. *Scientific American* 203, 3 (1960), 62–75.
[151] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, et al. 2022. Finetuned language models are zero-shot learners. In *Proceedings of the 10th International Conference on Learning Representations, ICLR 2022*. OpenReview.net. https://arxiv.org/abs/2109.01652
[152] 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. arXiv:2206.07682. Retrieved from https://arxiv.org/abs/2206.07682
[153] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. arXiv:2201.11903. Retrieved from https://arxiv.org/abs/2201.11903
[154] Jess Whittlestone, Rune Nyrup, Anna Alexandrova, Kanta Dihal, and Stephen Cave. 2019. Ethical and Societal Implications of Algorithms, Data, and Artificial Intelligence: A Roadmap for Research. Nuffield Foundation, London .
[155] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. Visual ChatGPT: Talking, drawing and editing with visual foundation models. arXiv:2303.04671. Retrieved from https://arxiv.org/abs/2303.04671
[156] Yuwei Wu, Xuezhe Ma, and Diyi Yang. 2021. Personalized response generation via generative split memory network. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 1956–1970. DOI: https://doi.org/10.18653/v1/2021.naacl-main.157
[157] Joern Wuebker, Patrick Simianer, and John DeNero. 2018. Compact personalized models for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 881–886. DOI: https://doi.org/10.18653/v1/D18-1104
[158] Lin Xu, Zhiyuan Hu, Daquan Zhou, Hongyu Ren, Zhen Dong, et al. 2023. Magic: Investigation of large language model powered multi-agent in cognition, adaptability, rationality and collaboration. In *Proceedings of the ICLR 2024 Workshop on Large Language Model (LLM) Agents*. https://aclanthology.org/2024.emnlp-main.416
[159] Rui Yan, Jian-Yun Nie, and Xiaoming Li. 2011. Summarize what you are interested in: An optimization framework for interactive personalized summarization. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Edinburgh, Scotland, UK., 1342–1351.
[160] Diyi Yang and Lucie Flek. 2021. Towards user-centric text-to-text generation: A survey. In *Proceedings of the 24th International Conference on Text, Speech, and Dialogue, TSD 2021*. Springer, 3–22.
[161] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. SWE-agent: Agent-computer interfaces enable automated software engineering. arXiv:2405.15793. Retrieved from https://arxiv.org/abs/2405.15793
[162] Sherry Yang, Ofir Nachum, Yilun Du, Jason Wei, Pieter Abbeel, and Dale Schuurmans. 2023. Foundation models for decision making: Problems, methods, and opportunities. arXiv:2303.04129. Retrieved from https://arxiv.org/abs/2303.04129
[163] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. arXiv:2207.01206. Retrieved from https://arxiv.org/abs/2207.01206
[164] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. arXiv:2210.03629. Retrieved from https://arxiv.org/abs/2210.03629
[165] Yining Ye, Xin Cong, Shizuo Tian, Jiannan Cao, Hao Wang, Yujia Qin, Yaxi Lu, Heyang Yu, Huadong Wang, Yankai Lin, et al. 2023. Proagent: From robotic process automation to agentic process automation. arXiv:2311.10751. Retrieved from https://arxiv.org/abs/2311.10751
[166] Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. 2023. Lumos: Learning agents with unified data, modular design, and open-source LLMs. arXiv:2311.05657. Retrieved from https://arxiv.org/abs/2311.05657
[167] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, et al. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019 NeurIPS 2019*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 9051–9062. https://arxiv.org/abs/1905.12616
[168] Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023. AgentTuning: Enabling generalized agent abilities for LLMs. arXiv:2310.12823. Retrieved from https://arxiv.org/abs/2310.12823
[169] Andy Zeng, Adrian Wong, Stefan Welker, Krzysztof Choromanski, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, et al. 2022. Socratic models: Composing zero-shot multimodal reasoning with language. arXiv:2204.00598. Retrieved from https://arxiv.org/abs/2204.00598

[170] Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. 2024. Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. arXiv:2401.07339. Retrieved from https://arxiv.org/abs/2401.07339

[171] Kechi Zhang, Huangzhao Zhang, Ge Li, Jia Li, Zhuo Li, and Zhi Jin. 2023. ToolCoder: Teach code generation models to use API search tools. arXiv:2305.04032. Retrieved from https://arxiv.org/abs/2305.04032

[172] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing Dialogue Agents: I have a dog, do you have pets too?. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 2204–2213. DOI: https://doi.org/10.18653/v1/P18-1205

[173] Hanxun Zhong, Zhicheng Dou, Yutao Zhu, Hongjin Qian, and Ji-Rong Wen. 2022. Less is more: Learning to refine dialogue history for personalized dialogue generation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Seattle, United States, 5808–5820. DOI: https://doi.org/10.18653/v1/2022.naacl-main.426

[174] Shuyan Zhou, Uri Alon, Frank F. Xu, Zhiruo Wang, Zhengbao Jiang, and Graham Neubig. 2023. DocPrompting: Generating code by retrieving the docs. arXiv:2207.05987. Retrieved from https://arxiv.org/abs/2207.05987

[175] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. arXiv:1909.08593. Retrieved from https://arxiv.org/abs/1909.08593

Received 19 July 2023; revised 9 October 2024; accepted 3 November 2024